The Discontinuous Galerkin Method on Cartesian Grids with Embedded Geometries: Spectrum Analysis and Implementation for Euler Equations

by

Ruibin Qin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, we analyze theoretical properties of the discontinuous Galerkin method (DGM) and propose novel approaches to implementation with the aim to increase its efficiency. First, we derive explicit expressions for the eigenvalues (spectrum) of the discontinuous Galerkin spatial discretization applied to the linear advection equation. We show that the eigenvalues are related to the subdiagonal $[p/p+1]$ Padé approximation of e^{-z} when the p-th degree basis functions are used. We derive an upper bound on the eigenvalue with the largest magnitude as $(p+1)(p+2)$. We demonstrate that this bound is not tight and prove that the asymptotic growth rate of the spectral radius is slower than quadratic in p. We also analyze the behavior of the spectrum near the imaginary axis to demonstrate that the spectral curves approach the imaginary axis although there are no purely imaginary eigenvalues.

Then, we extend the analysis to nonuniform meshes where both the size of elements and the composition of the mesh influence the spectrum. We show that the spectrum depends on the ratio of the size of the largest to the smallest cell as well as the number of cells of different types. We find that the spectrum grows linearly as a function of the proportion of small cells present in the mesh when the size of small cells is greater than some critical value. When the smallest cells are smaller than this critical value, the corresponding eigenvalues lie outside of the main spectral curve. We provide an easily computable estimate of the upper bound on the spectrum. Numerical examples on nonuniform meshes are presented to show the improvement on the time step restriction. In particular, this result can be used to improve the time step restriction on Cartesian grids.

Finally, we present a discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries. Cartesian grids are an alternative to a more popular unstructured triangular/tetrahedral approach. Their advantage is a simplified mesh generation and a computational efficiency resulting from the structure of a Cartesian grid. However, cutting an embedded geometry out of the grid creates cut cells, which are difficult to deal with for two reasons. One is the restrictive CFL number and the other is the integration on irregularly shaped cells. Both of these issues are more involved for the DGM than for finite volume methods, which most Cartesian grid techniques have been developed for. We use explicit time integration employing cell merging to avoid restrictively small time steps. We provide an algorithm for splitting complex cells into triangles and use standard quadrature rules on these for numerical integration. To avoid the loss of accuracy due to straight sided grids, we employ the curvature boundary conditions. We show that the proposed method is robust and high-order accurate.

Acknowledgements

I would like to acknowledge my supervisor Lilia Krivodonova for her patience and support. She patiently helped to improve my writing, and provided valuable guidance and discussion in solving problems. I would not complete this thesis without her help.

I would also like to thank my colleagues for reading and providing helpful comments on the draft. Special thanks are due to Noel Chalmers, Dale Connor and Martin Fuhry. Thanks are also due to Cong Wang for his efforts on implementing the mesh generator.

I would like to acknowledge the examining committee, Hans De Sterck, Michael Waite, Fue-Sang Lien and Sigal Gottlieb for reviewing this thesis. Their various insights are much appreciated.

Finally, I would like to express my appreciation to my parents. Their encouragement and love supported me during the years I was studying in Waterloo.

Dedication

To my parents

Table of Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

The discontinuous Galerkin method (DGM) has become a popular approach to numerical solution of hyperbolic conservation laws in the last twenty years. Such equations are used in modeling of compressible flows [47], electromagnetics [53], acoustics [26], and other problems involving wave propagation. The method has many attractive properties, including the potential for achieving arbitrary high-order accuracy. The DGM has excellent parallel efficiency and is suitable for handling complicated geometries. This method can also easily handle adaptive strategies since it does not require conforming meshes. However, it requires evaluation of integrals over elemental volumes and surfaces and its CFL condition depends on the order of approximation. These make it more expensive than finite volume and finite difference schemes. In this thesis, we analyze the theoretical properties of the method and propose novel approaches to implementation with the aim to increase its efficiency.

We start with analysis of the spectrum of the DGM. In particular, we derive an explicit formula for the eigenvalues of the DG spatial discretization operator for one dimensional linear advection equation. These are important for two reasons. Firstly, it is interesting from a theoretical point of view, as it helps us to understand the restriction on the time step with the increase in the order of spatial approximation. Secondly, it can be used to relax this restriction as we show in [7, 39]. We show that the eigenvalues are related to the subdiagonal $[p/p+1]$ Padé approximants of e^{-z} when p-th degree basis functions are used. We derive an upper bound on the eigenvalue with the largest magnitude as $(p+1)(p+2)$. We demonstrate that this bound is not tight, and prove that the asymptotic growth rate of the spectral radius is slower than quadratic in p. We also analyze the behavior of the spectrum near the imaginary axis to demonstrate that the spec-

tral curves approach the imaginary axis as p increases, although there are no purely imaginary eigenvalues.

Cartesian grids, in practice, often contain cells of different sizes, which is a result of mesh refinement and the presence of irregular cells. On nonuniform meshes, the global CFL restriction is scaled according to the size of the smallest cell, which can be very inefficient [11]. More sophisticated approaches group cells of similar sizes and advance each group with appropriate locally defined fractional time steps [36, 17, 18]. This reduces the total number of times the DG spatial approximation needs to be evaluated which leads to computational savings. Numerical tests indicate that the restriction can be relaxed in a mesh containing a low number of small cells. This motivates us to look at the spectrum of the DG spatial discretization on nonuniform meshes. In chapter 3, we follow the derivations of chapter 2 and find that the eigenvalues also depend on the subdiagonal Padé approximants of e^{-z}. We show that not only the order of the approximation but also the structure of the mesh affect the distribution of eigenvalues. We find that presence of half-size cells in a mesh does not cause the spectrum to grow twice larger; instead, the spectrum grows approximately linearly depending on the proportion of half-size cells. We also find that presence of a few tiny cells in a mesh results in the eigenvalues of the largest magnitude to be scaled according to the size of the smallest cell size. In both cases, the CFL restriction could be relaxed.

In chapter 4, we seek a practical technique for solution of two dimensional problems. Cartesian grid methods can provide considerable computational savings for computationally intensive schemes like the DGM. Cartesian mesh generation is also simplified compared to the body fitted meshes. However, small cut cells may appear near the boundary of embedded geometries. Analysis on nonuniform meshes reveals that the extremely small element still restricts the stable time step. Thus, we develop an algorithm to eliminate small cut cells by merging them with their neighbors. Beside the sizes of cut cells, the irregular shapes of cut cells also introduce difficulties to the volume integrals. We provide an algorithm to split cut cells into triangles and use standard quadrature rules on these for numerical integration [45].

1.2 Conservation laws

Hyperbolic systems of conservation laws are time-dependent systems of partial differential equations. A system of conservation laws can be written in a general form as follows

$$\partial_t \boldsymbol{u}(\vec{x}, t) + \boldsymbol{div}\vec{\boldsymbol{F}}(\boldsymbol{u}(\vec{x}, t)) = \mathbf{0}. \tag{1.1}$$

Here, the variables with a superimposed arrow denote vectors in $\mathbb{R}^d$, where $d = 1, 2, 3$ is the dimension of the space. The variable in bold type denote vectors in $\mathbb{R}^m$, where m is the number of equations in the system. Then, $\boldsymbol{u}$ is an m-dimensional vector of conserved quantities or state variables. $\vec{\boldsymbol{F}}$ is an $m \times d$ matrix called the flux matrix, which allows us to determine the rate of flow of each state variable at $(\vec{x}, t)$. $\boldsymbol{div}$ is a vector valued divergence operator, $\boldsymbol{div} = [div, div, \ldots, div]^T$. Thus, the i^{th} equation of this system is

$$\partial_t u_i + div\vec{F}_i(\boldsymbol{u}) = 0,$$

where $\vec{F}_i(\boldsymbol{u})$ is the i^{th} row of the flux matrix $\vec{\boldsymbol{F}}$. If we write the flux matrix as $\vec{\boldsymbol{F}} = [\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_d]$, where $\boldsymbol{F}_i,\ i = 1, \ldots, d$ are the columns of the flux matrix, then system (1.1) is hyperbolic when the matrix $A \coloneqq \sum_{i=1}^{d} \alpha_i \frac{D\boldsymbol{F}_i}{D\boldsymbol{x}}$ has only real eigenvalues and is diagonalizable for any $\alpha_1, \ldots, \alpha_d \in \mathbb{R}$.

The two-dimensional Euler equations describing compressible, inviscid flows are an example of a system of hyperbolic conservation laws. In this case, $\boldsymbol{u} = [\rho, \rho v_x, \rho v_y, E]^T$, $\vec{\boldsymbol{F}} = [\rho\vec{v}, \rho v_x\vec{v} + p\vec{i}_x, \rho v_y\vec{v} + p\vec{i}_y, \vec{v}(E + p)]^T$, and the system can be written as

$$\partial_t \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix} + \boldsymbol{div} \begin{pmatrix} \rho v_x & \rho v_y \\ \rho v_x^2 + p & \rho v_x v_y \\ \rho v_x v_y & \rho v_y^2 + p \\ v_x(E + p) & v_y(E + p) \end{pmatrix} = \boldsymbol{0}. \tag{1.2}$$

Here, ρ is the density, $\vec{v} = (v_x, v_y)$ is the velocity with v_x and v_y being the components of the velocity vector in x and y direction respectively, $\vec{i}_x$ is the unit vector in the x-direction and $\vec{i}_y$ is the unit vector in the y-direction. E is the total internal energy and p is the pressure. System (1.2) has four equations and five unknowns. One more equation is needed to close this system. An equation of state, which connects pressure, density and energy, serves this purpose. For the ideal gas, it has the form

$$p = (\gamma - 1)\left[E - \rho\frac{\|\vec{v}\|^2}{2}\right], \tag{1.3}$$

where γ is the heat capacity ratio. For air, we take $\gamma = 1.4$.

1.3 DGM for one-dimensional scalar problems

First, we derive the DGM for one-dimensional scalar problems as a simple introduction to the method. A general formulation for multidimensional systems is given in the next section.

Let us consider the equation

$$\partial_t u + \partial_x f(u) = 0 \tag{1.4}$$

on the interval $[a, b]$ with the initial condition $u(x, 0) = u_0(x)$ and periodic boundary conditions.

The interval $[a, b]$ is divided into N subintervals $[x_{j-1}, x_j]$, $j = 1, 2, \ldots, N$, where $a = x_0 < x_1 < \cdots < x_N = b$ and $\Delta x_j = x_j - x_{j-1}$. On each interval $[x_{j-1}, x_j]$, equation (1.4) is multiplied by a test function $v(x) \in H^1(x_{j-1}, x_j)$, and integrated on this interval to yield

$$\int_{x_{j-1}}^{x_j} u_t v \mathrm{d}x + \int_{x_{j-1}}^{x_j} f(u)_x v \mathrm{d}x = 0. \tag{1.5}$$

Then, using integration by parts, we obtain

$$\int_{x_{j-1}}^{x_j} u_t v \mathrm{d}x + f(u)v\Big|_{x_{j-1}}^{x_j} - \int_{x_{j-1}}^{x_j} f(u) v_x \mathrm{d}x = 0. \tag{1.6}$$

Now, we use $\xi = \frac{2}{\Delta x_j}(x - \frac{x_{j-1}+x_j}{2})$ to map the interval $[x_{j-1}, x_j]$ to the canonical element $[-1, 1]$. Changing the variable x in (1.6) gives

$$\frac{\Delta x_j}{2} \int_{-1}^{1} u_t v \mathrm{d}\xi + f(u)v\Big|_{-1}^{1} - \int_{-1}^{1} f(u) v_\xi \mathrm{d}\xi = 0. \tag{1.7}$$

Notice that we use $u(\xi, t)$ in (1.7) to denote the function $u(x(\xi), t)$.

The exact solution $u(\xi, t)$ is approximated on element j by a polynomial of degree up to p, and can be written as

$$u(\xi, t) \approx U_j = \sum_{k=0}^{p} c_{jk}(t)\varphi_k(\xi), \tag{1.8}$$

where $\{\varphi_k(\xi)\}$ are the basis functions in the finite dimensional polynomial space. We use the Legendre polynomials as a basis for one dimensional problems. The Legendre polynomials are given by [1]

$$P_k(\xi) = \frac{1}{2^k k!} \frac{d^k}{d\xi^k}(\xi^2 - 1)^k, \; k = 0, 1, \ldots. \tag{1.9}$$

They form an orthogonal set on $[-1, 1]$ and satisfy

$$\int_{-1}^{1} P_k(\xi) P_i(\xi) \mathrm{d}\xi = \frac{2}{2\mathrm{k} + 1} \delta_{\mathrm{ki}}. \tag{1.10}$$

With the normalization (1.10), the Legendre polynomials have the following properties

$$P_k(1) = 1,\ P_k(-1) = (-1)^k. \tag{1.11}$$

Here are some examples of the Legendre polynomials

$$\begin{aligned} P_0(\xi) &= 1, \\ P_1(\xi) &= \xi, \\ P_2(\xi) &= \frac{3}{2}\xi^2 - \frac{1}{2}, \\ P_3(\xi) &= \frac{5}{2}\xi^3 - \frac{3}{2}\xi. \end{aligned}$$

Substituting (1.8) into (1.7) gives

$$\frac{\Delta x_j}{2}\frac{\mathrm{d}}{\mathrm{d}t}\int_{-1}^{1}\sum_{k=0}^{p} c_{jk}P_kP_i\mathrm{d}\xi + F_n(U_j)P_i\Big|_{-1}^{1} - \int_{-1}^{1} f(U_j)P_i'\mathrm{d}\xi = 0,\ i = 0, 1, \ldots, p. \tag{1.12}$$

Using the orthogonal property of Legendre polynomials (1.10) and normalization (1.11), we obtain

$$\frac{\Delta x_j}{2i+1}\frac{\mathrm{d}}{\mathrm{d}t}c_{ji} + F_n(U_j)\Big|_{\xi=1} - (-1)^i F_n(U_j)\Big|_{\xi=-1} - \int_{-1}^{1} f(U_j)P_i'\mathrm{d}\xi = 0. \tag{1.13}$$

Notice that in (1.12) and (1.13) we use F_n to denote the numerical flux at the interfaces. This is due to the discontinuity of approximate solutions at the interfaces, e.g. at $x = x_{j-1}$, $U_{j-1}(1,t)$ and $U_j(-1,t)$ might take different values. Thus, a numerical flux function resolving the ambiguity is needed. For example, the local Lax-Friedrichs flux that is easy to compute and works well for relatively simple problems is given by

$$F_n(U_{j-1}, U_j) = \frac{1}{2}[f(U_{j-1}(1)) + f(U_j(-1)) - \lambda_{max}(U_j(-1) - U_{j-1}(1))], \tag{1.14}$$

where λ_{max} is the maximum absolute value of $f'(u)$, for u varying between $U_{j-1}(1)$ and $U_j(-1)$. Here time t is omitted since $U_{j-1}(1)$ and $U_j(-1)$ are taken at the same time t. When f' is constant, (1.14) is equivalent to the upwind flux

$$F_n(U_{j-1}, U_j) = \begin{cases} f(U_{j-1}(1)), & f' > 0 \\ f(U_j(-1)), & f' < 0. \end{cases} \tag{1.15}$$

The integral $\int_{-1}^{1} f(U_j)P_i'\mathrm{d}\xi$ in (1.13) is calculated using a numerical quadrature rule. For one dimensional problems, this can be done using a Gauss-Legendre quadrature

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{w_i f}(\xi_\mathrm{i}), \tag{1.16}$$

which is exact for polynomials of degree not higher than $2n-1$. Some low order rules are listed below:

$$\begin{aligned}
n=1,&\quad \int_{-1}^{1} f(\xi)\mathrm{d}\xi = 2\mathrm{f}(0);\\
n=2,&\quad \int_{-1}^{1} f(\xi)\mathrm{d}\xi = \mathrm{f}\left(-\frac{\sqrt{3}}{3}\right)+\mathrm{f}\left(\frac{\sqrt{3}}{3}\right);\\
n=3,&\quad \int_{-1}^{1} f(\xi)\mathrm{d}\xi = \frac{5}{9}\mathrm{f}\left(-\sqrt{\frac{3}{5}}\right)+\frac{8}{9}\mathrm{f}(0)+\frac{5}{9}\mathrm{f}\left(\sqrt{\frac{3}{5}}\right).
\end{aligned}$$

Higher order quadratures can be found in [1]. For p-th order approximation, the quadrature rule is chosen to be exact for polynomials of degree at least $2p$ for nonlinear problems, and $2p-1$ for linear problems.

Moving the numerical flux terms and the volume integral to the righthand side, (1.13) can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{c}(t) = L_h(\boldsymbol{c},t), \tag{1.17}$$

where $\boldsymbol{c}$ is the vector of the degrees of freedom, which consists of c_{ji}, $j=1,2,\ldots,N$, $i=0,1,\ldots,p$.

Equation (1.17) is a system of ODEs that can be solved by an explicit or implicit ODE solver. We employ explicit Runge-Kutta schemes to advance (1.17) in time. An advantage of using an explicit scheme is that computations can be performed element-wise, i.e. do not require construction of a global matrix and solution of a large linear algebra problem. Also, explicit schemes are easier to implement than implicit schemes. Usually, the p-th order DG approximation is coupled with a $p+1$-th order Runge-Kutta scheme. The time step is chosen to satisfy the CFL condition

$$\Delta t \leqslant \min_j \frac{\Delta x_j}{(2p+1)\lambda_j}, \tag{1.18}$$

where λ is the absolute value of the wave speed $f'(u)$ [11].

The initial condition $u(x(\xi),0) = u_0(x(\xi))$ is projected onto the basis functions on element j

by the $\mathcal{L}^2$ projection

$$\int_{-1}^{1} u_0(x(\xi))P_i(\xi)\mathrm{d}\xi = \int_{-1}^{1} \sum_{k=0}^{p} c_{jk}(\xi)P_k(\xi)P_i(\xi)\mathrm{d}\xi,\ i = 0, 1, \ldots, p. \tag{1.19}$$

Using the orthogonal property of the basis functions (1.10), we can compute the initial solution coefficients as

$$c_{ji}(0) = \frac{2i+1}{2} \int_{-1}^{1} u_0(x(\xi))P_i(\xi)\mathrm{d}\xi,\ i = 0, 1, \ldots, p. \tag{1.20}$$

In our test problems, we use periodic boundary conditions. To implement these, a ghost cell is created before the first and after the last cell. In each stage of the computation, solutions on the ghost cell next to the first cell are updated using the solutions on the last cell, and similarly to the other ghost cell. When other boundary conditions are used, numerical flux at the boundaries needs to be adjusted using the specified boundary conditions. Limiters are needed when the solution has discontinuities.

1.4 DGM for multi-dimensional problems

In the previous section, we described the DGM applied to a one-dimensional scalar problem. Now, we will extend this method to multi-dimensional system of equations. In order to construct a discontinuous Galerkin method, the computational domain Ω is divided into a collection of elements

$$\Omega = \bigcup_{j=1}^{N_h} \Omega_j.$$

Then, a Galerkin problem on an element Ω_j is constructed by multiplying (1.1) by a test function $v \in \mathcal{H}^1(\Omega_j)$, integrating the result on Ω_j, and using the Divergence theorem to obtain

$$\int_{\Omega_j} v\partial_t \boldsymbol{u}\mathrm{d}\tau - \int_{\Omega_j} \vec{\boldsymbol{F}}(\boldsymbol{u}) \cdot \boldsymbol{grad} v\mathrm{d}\tau + \int_{\partial\Omega_j} v\boldsymbol{F}_n\mathrm{d}\sigma = 0, \tag{1.21}$$

where $\partial\Omega_j$ is the boundary of Ω_j, $\boldsymbol{F}_n = \vec{\boldsymbol{F}}(\boldsymbol{u}) \cdot \vec{n}$, and $\vec{n}$ is the outward facing normal vector. Several issues must be resolved before the above form can be used as a numerical method.

Basis functions

In the two-dimensional case, we use coordinate variables (x, y) to describe a point in the physical space and (ξ, η) in the computational space. Triangular elements are usually mapped

onto a canonical triangle $\Omega_0 = \{(\xi, \eta) | 0 \leqslant \xi, \eta \leqslant 1 \text{ and } \xi + \eta \leqslant 1\}$. The mapping between Ω_j and Ω_0 is given by

$$\xi = \frac{\det \begin{bmatrix} 1 & x & y \\ 1 & x_1 & y_1 \\ 1 & x_3 & y_3 \end{bmatrix}}{\det \begin{bmatrix} 1 & x_2 & y_2 \\ 1 & x_1 & y_1 \\ 1 & x_3 & y_3 \end{bmatrix}}, \quad \eta = \frac{\det \begin{bmatrix} 1 & x & y \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix}}{\det \begin{bmatrix} 1 & x_3 & y_3 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix}}, \tag{1.22}$$

where (x_i, y_i), $i = 1, 2, 3$ are the coordinates of the vertices of the triangle Ω_j. We use this mapping on triangular cut cells in Cartesian grids. Rectangular elements are usually mapped onto a unit square $\Omega_0 = \{(\xi, \eta) | -1 \leqslant \xi, \eta \leqslant 1\}$ by a simple linear mapping.

Without the requirement of continuity, there is a great flexibility in the choice of basis functions φ_k, $k = 0, 1, \ldots, N_p$. A convenient choice is to make them orthogonal in $\mathcal{L}^2(\Omega_0)$, which will produce a diagonal mass matrix. Here the mass matrix refers to $M = (\int_{\Omega_0} \varphi_i \varphi_k \mathrm{d}\tau)$, $i, k = 0, 1, \ldots, N_p$, which is introduced by the first term in (1.21). The diagonal mass matrix can significantly simplify computations since it can be easily inverted and does not need memory space to store.

On the canonical square $[-1, 1] \times [-1, 1]$, the tensor product of Legendre polynomials forms an orthogonal basis,

$$\{P_i(\xi) P_k(\eta) | i, k = 0, 1, \ldots, p, \ (\xi, \eta) \in [-1, 1] \times [-1, 1]\},$$

where $P_k(\xi), k = 0, 1, \ldots$ are the Legendre polynomials (1.9). The tensor product basis is a convenient choice since integration over Ω_0 can be replaced by two sequential one-dimensional integrals. Hence, a tensor product of one-dimensional quadrature rules can be used. However, the tensor product basis is not optimal in terms of the number of basis functions for a given order of approximation and not orthogonal on non-rectangular elements. Indeed, the number of functions in a monomial basis $1, x, y, x^2, xy, y^2, \ldots, y^p$ is $(p + 1)(p + 2)/2$, while there are $(p+1)^2$ functions in the tensor product basis.

The Gram-Schmidt orthogonalization provides us with a method to construct an orthogonal set of polynomials on elements of arbitrary shape. Given a set of basis functions $M = \{m_i, i = 1, 2, \ldots, N_p\}$ of degree up to p, we can form an orthogonal basis $\Psi = \{\psi_i\}_{i=1}^{N_p}$ in $\mathcal{L}^2(\Omega_j)$ as

$$\psi_i = \frac{m_i - \sum_{k=1}^{i-1} (m_i, \psi_k) \psi_k}{\sqrt{\left(m_i, m_i - \sum_{k=1}^{i-1} (m_i, \psi_k) \psi_k\right)}}, \ i = 1, 2, \ldots, N_p, \tag{1.23}$$

where (f, g) is the inner product $\int_{\Omega_j} fg\mathrm{d}x$ on Ω_j and the summations are zero when their upper limit is less than their lower limit. This approach can be used to construct an orthogonal basis on each cut cell, but we do not employ it. We will discuss basis functions on cut cells in Section 4.5.2.

Quadrature rules

The integrals in (1.21) need to be evaluated by numerical quadrature rules. Similar to one-dimensional cases, quadrature rules for the volume integrals should be exact for polynomials of degree $2p$ if the approximation and test functions are of degree p. The boundary integrals should be evaluated using quadrature rules that are exact for polynomials of degree $2p + 1$ in multi-dimensional space [11]. On rectangular elements, we use the tensor product of one-dimensional quadrature rules as we discussed above. For triangular elements, we use the quadrature rules described in [20]. However, these quadrature rules can not be directly applied to irregular elements, e.g. cut cells in Cartesian grids. We will propose a technique to overcome this difficulty in Chapter 4.

Numerical flux

For linear problems, we use the upwind numerical flux similarly to (1.15). For nonlinear problems, we use the Roe's flux [51, 32].

Time integration

After evaluating the numerical flux, computing integrals using a quadrature rule and inverting the mass matrix, equation (1.21) can be rewritten in an ODE form (1.17). Notice that the basis functions are discontinuous, the mass matrix is block-diagonal and the blocks can be easily inverted. If a local orthogonal basis is chosen, the mass matrix is diagonal.

The equation (1.17) is solved using an ODE solver. To preserve stability of the scheme, we use the strong stability preserving (SSP) explicit Runge-Kutta time discretization [49]. Let $\{t^n\}_{n=0}^M$ be a partition of $[0, T]$, where T is the final time, and $\Delta t^n = t^{n+1} - t^n, n = 0, \ldots, M - 1$. The time stepping algorithm is given by:

- Initialize $\boldsymbol{c}_0$ by projecting the initial condition $u_0(x)$ onto the basis functions $\{\varphi_i(x)\}$ on cell Ω_j using the $\mathcal{L}^2$ projection (1.19).
- For $n = 0, \ldots, M - 1$ compute $\boldsymbol{c}_{n+1}$ as follows:
    1. Set $K_1 = L_h(t_n, \boldsymbol{c}_n)$;

1. For $i = 2, \ldots, s$ compute the intermediate stages

$$K_i = L_h(\boldsymbol{c}_n + \Delta t_n \sum_{l=1}^{i-1} b_{il} K_l, t_n + a_i \Delta t_n); \tag{1.24}$$

2. Update $\boldsymbol{c}_{n+1} = \boldsymbol{c}_n + \Delta t_n (d_1 K_1 + \cdots + d_s K_s)$.

We use the second and third order Runge-Kutta time discretizations listed in the Butcher tableau (Table 1.1) for the piecewise linear and piecewise quadratic approximations respectively.

$$\begin{array}{c|ccccc} 0 & & & & & \\ a_2 & b_{21} & & & & \\ a_3 & b_{31} & b_{32} & & & \\ \vdots & \vdots & \vdots & & & \\ a_s & b_{s1} & b_{s2} & \cdots & b_{s(s-1)} & \\ \hline & d_1 & d_2 & \cdots & d_{s-1} & d_s \end{array}$$

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 0.5 & 0.5 \end{array}$$

$$\begin{array}{c|ccc} 0 & & & \\ 0.5 & 0.5 & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \end{array}$$

Table 1.1: Butcher tableau for examples with $s = 2, 3$.

The size of a stable time step depends on the order of a discontinuous Galerkin approximation and the stability region of the chosen time integration scheme. For $p = 1, 2, 3$ and an s stage s order Runge-Kutta method ($s = 2, 3$), the time step is given by (1.18). When solving nonlinear problems or using an adaptive scheme, the time step needs to be recalculated after each iteration as the wave speed might change.

Slope limiting

All linear numerical schemes of order higher than one can become unstable [24], so a procedure to stabilize the solution and remove spurious oscillations is needed. Such a procedure is

nonlinear even for linear problems. One approach to stabilization is to use limiters, which we describe below.

In one dimension, a linear approximation can be written as

$$U_j(x) = c_j \frac{(x - x_{j+1/2})}{\Delta x_j/2} + \bar{u}_j, \tag{1.25}$$

where $x_{j+1/2}$ is the middle point of interval $[x_j, x_{j+1}]$, $\Delta x_j = x_{j+1} - x_j$, $\bar{u}_j$ is the mean value of $U_j(x)$ on the same interval and c_j is the slope. The limited slope c_j^{lim} is determined as

$$c_j^{lim} = minmod(c_j, \bar{u}_{j+1} - \bar{u}_j, \bar{u}_j - \bar{u}_{j-1}), \tag{1.26}$$

where the function *minmod* is defined by

$$minmod(a_1, a_2, \ldots, a_m) = \begin{cases} s \min_i |a_i|, & \text{if } s = sign(a_1) = \cdots = sign(a_m), \\ 0, & \text{otherwise.} \end{cases} \tag{1.27}$$

In a rectangular element $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$, a linear approximating function can be written as

$$U(x, y, t) = \bar{c}(t) + c_{1x}(t)\varphi_i(x) + c_{1y}(t)\psi_j(y), \tag{1.28}$$

where

$$\varphi_i(x) = \frac{x - x_{i+1/2}}{\Delta x_i/2}, \quad \psi_j(y) = \frac{y - y_{j+1/2}}{\Delta y_j/2}, \tag{1.29}$$

and

$$\Delta x_i = x_{i+1} - x_i, \quad \Delta y_j = y_{j+1} - y_j.$$

The limiting is performed on the 'slope' c_{1x} and c_{1y} in (1.28), using the differences of the means. For a scalar equation, c_{1x} will be limited by

$$minmod(c_{1x}, \bar{c}_{i+1,j} - \bar{c}_{i,j}, \bar{c}_{i,j} - \bar{c}_{i-1,j}). \tag{1.30}$$

Similarly, c_{1y} is replaced by $minmod(c_{1y}, \bar{c}_{i,j+1} - \bar{c}_{i,j}, \bar{c}_{i,j} - \bar{c}_{i,j-1})$.

For a system of equations, the limiting is performed in the local characteristic variables. For example, the vector $\boldsymbol{c}_{1x}$ in element (i, j) is limited as follows:

1. Compute the matrix R such that

$$R^{-1} \frac{\partial \boldsymbol{F}_1(\bar{\boldsymbol{u}}_{ij})}{\partial \boldsymbol{u}} R = \Lambda,$$

where Λ is a diagonal matrix containing the eigenvalues of the Jacobian and the Jacobian in x-direction $(\partial \boldsymbol{F}_1(\bar{\boldsymbol{u}}_{ij}))/(\partial \boldsymbol{u})$ is evaluated at the mean $\bar{\boldsymbol{u}}_{ij}$ in the element (i, j).

2. Transform $\boldsymbol{c}_{1x}, \bar{\boldsymbol{c}}_{i+1,j} - \bar{\boldsymbol{c}}_{i,j}$ and $\bar{\boldsymbol{c}}_{i,j} - \bar{\boldsymbol{c}}_{i-1,j}$ to the characteristic fields by left-multiplying them by R^{-1}.

3. Apply the scalar limiter (1.27) to each of the components of the transformed vectors.

4. Transform the result back to the original space by left-multiplying by R.

Limiting for higher orders of approximation can be performed using the moment limiter [35].

Interpolation Error

The accuracy of the scheme is influenced by a number of factors, such as the formal order of approximation, smoothness of the solution and, for multidimensional problems, the quality of the mesh. Thin or distorted elements are usually detrimental for the quality of the numerical solution. This is an important issue as Cartesian grids with embedded geometries might produce such thin cut cells. We list here two important theoretical results related to interpolation of an arbitrary function u by a piecewise polynomial U on a given mesh [22].

Theorem 1. *Let Ω be a polygonal domain that has been discretized into a set of triangles Ω_j, $j = 1, 2, \ldots, N$. Let Δx and α denote the largest element edge and smallest angle in the mesh respectively. Let U interpolate u such that no error results when u is any polynomial of degree p or less. Then, there exists a constant $C > 0$, independent of $u \in \mathcal{H}^{p+1}$ and the mesh, such that*

$$|u - U|_q \leqslant \frac{C h^{p+1-q}}{(\sin \alpha)^q} |u|_{p+1}, \quad \forall u \in \mathcal{H}^{p+1}(\Omega), \quad q = 0, 1. \tag{1.31}$$

A similar result applies to rectangles.

Theorem 2. *Let the rectangular domain Ω be discretized into a mesh of rectangular elements Ω_j, $j = 1, 2, \ldots, N$. Let Δx and β denote the largest element edge and smallest edge ratio in the mesh, respectively. Let U interpolate u such that no error results when u is any polynomial of degree p or less. Then, there exists a constant $C > 0$, independent of $u \in \mathcal{H}^{p+1}$ and the mesh, such that*

$$|u - U|_q \leqslant \frac{C h^{p+1-q}}{\beta^q} |u|_{p+1}, \quad \forall u \in \mathcal{H}^{p+1}(\Omega), \quad q = 0, 1. \tag{1.32}$$

These two theorems mean that the error between numerical solution and the exact solution has an upper bound depending on the quality of a mesh no matter which numerical method

we use. It can be shown that the upper bound can be achieved for certain types of u. Thus, reducing upper bounds of error should make the solution more accurate. We conclude that a mesh containing thin triangles or thin rectangles might result in a less accurate solution than one obtained on a mesh of better quality as small α in (1.31) or small β in (1.32) will result in a higher upper bound. We further address this issue in Chapter 4.

Chapter 2

Spectrum Analysis of the DGM on Uniform Grids

2.1 Introduction

In this chapter we derive explicit expressions for the eigenvalues (spectrum) of the semi-discrete discontinuous Galerkin method applied to the one-dimensional linear advection equation. Recall from Section 1.3 the DG spatial discretization results in a linear system of ODEs

$$\frac{d}{dt}\mathbf{c} = \frac{a}{\Delta x} L\mathbf{c} \tag{2.1}$$

for $(p+1)N$ degrees of freedom $\mathbf{c}$ on an N element uniform mesh with p-th degree approximation in space. Here, a is the wave speed and Δx is the cell size. We show that for a discretization with the upwind flux (1.15 and periodic boundary conditions, the eigenvalues of L are given by $f_{p+1}(\lambda) = exp(\frac{2\pi i}{N} j)$, $j = 0, 1, \ldots, N-1$, where $f_{p+1}(z)$ is the subdiagonal $[p/p+1]$ Padé approximant of $exp(-z)$.

A direct application of the eigenvalue analysis is to the linear stability of the fully discrete scheme. Equation (2.1) is usually integrated in time using a suitable ODE solver. Thus, the necessary condition for the stability of the method is to require the time step Δt to be small enough so that the full spectrum of $\frac{a\Delta t}{\Delta x} L$ fits inside the absolute stability region of the chosen time integration scheme. The eigenvalues of L can be computed using a linear algebra software which has been done for a variety of combinations of spatial orders and time integration schemes [14, 32]. However, the analytical form of the eigenvalues has not been previously known. It is interesting from a purely theoretical point of view and can also be used to get further insight into

the DG method. We use it to improve the CFL number by manipulating the scheme so that the spectrum of L is shrunk [7]. This is achieved by constructing a different rational approximant of $exp(-z)$ which seeks to preserve the order of accuracy in the L^2 norm.

A linear stability analysis of (2.1) arising from a low order DG spatial discretization and Runge-Kutta time integration was previously performed in [12, 8]. It was shown that the DGM with $p > 0$ is not stable with a fixed CFL number when the forward Euler time integration is used [8]. This is caused by the eigenvalues of (2.1) being located very close to the imaginary axis which is not included in the stability region of the forward Euler method. It was proven in [12] that the DG method with $p = 1$ and the second order Runge-Kutta scheme is L^2 stable with the CFL number equal to $1/3$. It was further hypothesized there that a coupling of a pth degree DG scheme with a $(p + 1)$st order RK scheme is stable under a CFL condition $1/(2p + 1)$. In recent years, the DGM has been used with a variety of explicit time integration schemes, such as Adams-Bashforth [23], strong-stability preserving schemes [25], low storage RK schemes [32]. In this view, the universal CFL number seems to be of less importance.

Using the obtained expressions for the eigenvalues, we analyze the asymptotic behavior of the spectrum as the order of approximation p goes to infinity. The real eigenvalue, which is conjectured to be the largest in magnitude, and the real component of complex eigenvalues is shown to be bounded from below by $-(p+1)(p+2)$ for any p. However, we prove that the actual growth rate of the size of the largest eigenvalue is slower than quadratic. Numerical experiments indicate that $-1.5(p + 1)^{1.75}$ is an upper bound on the eigenvalues. The least square fitting gives a growth rate of about $1.4(p + 1)^{1.78}$ for $p < 100$. We also demonstrate that although the curves $|f_{p+1}(z)| = 1$ move closer to the imaginary axis as p increases there are no purely imaginary eigenvalues for any p.

A connection between the DG method and the Padé approximants has been observed previously. In [48], Le Saint and Raviart showed that the absolute stability region of the discontinuous Galerkin method used to solve an ODE is given by $|R(\lambda\Delta x)| \leq 1$, where $R(z)$ is the $[p/p + 1]$ Padé approximant of $exp(z)$. In [33], Hu and Atkins studied the dispersion properties of the DG scheme applied to the scalar advection equation in one dimension. They showed that for the physical mode, the numerical dispersion relation is accurate to $(k\Delta x)^{2p+2}$, where k is the wavenumber and $k\Delta x$ is small. Their reasoning was founded on the conjecture that certain polynomials involved in the analysis are related to $[p + 1/p]$ Padé approximation of $exp(z)$. An extended analysis of the dispersion and dissipation errors were given by Ainsworth in [3]. It was demonstrated there that the numerical wave speed $\tilde{k}$ satisfies the relation $f_{p+1}(-i\Delta x k) = exp(i\Delta x\tilde{k})$. The proof is based on a demonstration that DG solutions satisfy a certain eigenvalue problem conjectured in [33]. In Theorem 1 we show how this eigenvalue problem arises from the characteristic polynomial of L.

The $[p/p+1]$ and $[p+1/p]$ Padé approximants of $exp(z)$ are $O(z^{2p+2})$ accurate for small z. This explains the excellent dispersion and dissipation properties of the DGM which were called "superconvergent" in [33, 3]. This makes the scheme very suitable for wave propagation problems especially ones requiring long time integration. However, from our analysis it follows that the same approximants are involved in defining the spectrum of the semi-discrete method and, in this sense, are responsible for a severe time step restriction associated with the DGM. The small CFL number is frequently quoted as an disadvantage of the DGM. It makes the method, especially for low p and nonlinear problems, more expensive when compared to schemes that are able to maintain the CFL close to unity, e.g. finite volume schemes.

The rest of this chapter is organized as follows. We begin by deriving the discontinuous Galerkin formulation of the model problem with the aim to obtain a general form of the resulting systems of ODEs. In Section 2.4, we derive the equations that describe the eigenvalues and eigenvectors of the spatial discretization and prove our main result, i.e the relation between the characteristic polynomial of L and Padé approximants. Section 2.5 contains an analysis of the distribution of eigenvalues and the growth speed of the eigenvalue of the largest modulus. Finally, the dispersion relation of the DGM is discussed in Section 2.6.

2.2 Derivation of the characteristic polynomial of L

We consider the one-dimensional linear advection equation

$$u_t + au_x = 0 \tag{2.2}$$

subject to appropriate initial and periodic boundary conditions on interval I, $a > 0$. The domain is discretized uniformly into mesh elements $I_j = [x_{j-1}, x_j]$ of size Δx, $j = 1, 2, ..., N$. Following the derivation in Section 1.3, substituting (1.8) into (1.13) with the Legendre polynomials as the basis and test functions, and using the upwind numerical flux (1.15), we obtain that

$$\frac{\Delta x}{2k+1}\dot{c}_{jk} = -a\left(\sum_{i=0}^{p} c_{ji} - (-1)^k \sum_{i=0}^{p} c_{j-1,i}\right) + a\int_{-1}^{1}\left(\sum_{i=0}^{p} c_{ji}P_i\right)P_k'\,d\xi, \quad k = 0, 1, \ldots, p, \tag{2.3}$$

where the dot in $\dot{c}_{jk}$ represents differentiation with respect to t. Collecting common terms of c_{ji} results in

$$\dot{c}_{jk} = a\frac{2k+1}{\Delta x}\left[(-1)^k \sum_{i=0}^{p} c_{j-1,i} + \sum_{i=0}^{p}\left(\int_{-1}^{1} P_iP_k'\,d\xi - 1\right)c_{ji}\right], \quad k = 0, 1, \ldots, p. \tag{2.4}$$

This can be written in a vector form as

$$\dot{c}_{jk} = a\frac{2k+1}{\Delta x}\left((-1)^k[1,1,...,1]\mathbf{c}_{j-1} + [\int_{-1}^{1} P_0 P_k' d\xi - 1, ..., \int_{-1}^{1} P_p P_k' d\xi - 1]\mathbf{c}_j\right), \tag{2.5}$$

where $\mathbf{c}_j = [c_{j0}, c_{j1}, \ldots, c_{jp}]^T$ and $\mathbf{c}_{j-1}$ is defined similarly. Combining cell solution-coefficient vectors into a global vector $\mathbf{c} = [\mathbf{c}_0^T, \mathbf{c}_1^T, \ldots, \mathbf{c}_p^T]^T$, equation (2.5) can be written as a system of ODEs (2.1). With periodic boundary conditions, L is a block matrix of the form

$$L = \begin{bmatrix} A_n & 0 & 0 & \ldots & 0 & 0 & D_n \\ D_n & A_n & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & D_n & A_n \end{bmatrix}, \tag{2.6}$$

where D_n and A_n are $n \times n$ matrices, $n = p + 1$. For approximation of order p, there are $p + 1$ basis functions, so the size of each block is $(p + 1) \times (p + 1)$. In the following discussion this notation of n is consistent and n can always be replaced by $p + 1$. In the matrix L,

$$D_n = \begin{bmatrix} 1 & \ldots & 1 \\ -3 & \ldots & -3 \\ \vdots & & \vdots \\ (-1)^{n-1}(2n-1) & \ldots & (-1)^{n-1}(2n-1) \end{bmatrix}, \tag{2.7}$$

$$A_n = \begin{bmatrix} \int_{-1}^{1} P_0 P_0' d\xi - 1 & \ldots & \int_{-1}^{1} P_{n-1} P_0' d\xi - 1 \\ 3\left(\int_{-1}^{1} P_0 P_1' d\xi - 1\right) & \ldots & 3\left(\int_{-1}^{1} P_{n-1} P_1' d\xi - 1\right) \\ \vdots & & \vdots \\ (2n-1)\left(\int_{-1}^{1} P_0 P_{n-1}' d\xi - 1\right) & \ldots & (2n-1)\left(\int_{-1}^{1} P_{n-1} P_{n-1}' d\xi - 1\right) \end{bmatrix}, \tag{2.8}$$

or

$$A_n = (a_{ij}) = \left((2i-1)(\int_{-1}^{1} P_{j-1} P_{i-1}' d\xi - 1)\right). \tag{2.9}$$

Noticing that the derivatives of the Legendre polynomials satisfy [1]

$$(2k+1)P_k = P_{k+1}' - P_{k-1}', \tag{2.10}$$

we derive

$$P_{k+1}' = (2k+1)P_k + (2(k-2)+1)P_{k-2} + (2(k-4)+1)P_{k-4} + \ldots. \tag{2.11}$$

We use (2.11) with the orthogonality property of the Legendre polynomials (1.10) to simplify the integrals in A_n. We obtain

$$\int_{-1}^{1} P_i P_k' d\xi = \begin{cases} 0, & k \leqslant i, \\ 2, & k > i, \text{ and } (k-i) \equiv 1 \pmod 2, \\ 0, & k > i, \text{ and } (k-i) \equiv 0 \pmod 2. \end{cases} \tag{2.12}$$

Thus, A_n can be simplified as

$$A_n = -\begin{bmatrix} a_1 & a_1 & a_1 & \cdots & a_1 & a_1 \\ -a_2 & a_2 & a_2 & \cdots & a_2 & a_2 \\ a_3 & -a_3 & a_3 & \cdots & a_3 & a_3 \\ -a_4 & a_4 & -a_4 & \cdots & a_4 & a_4 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ (-1)^{n-2}a_{n-1} & (-1)^{n-3}a_{n-1} & (-1)^{n-4}a_{n-1} & \cdots & a_{n-1} & a_{n-1} \\ (-1)^{n-1}a_n & (-1)^{n-2}a_n & (-1)^{n-3}a_n & \cdots & -a_n & a_n \end{bmatrix}, \tag{2.13}$$

where $a_i = 2i - 1,\ i = 1, 2, \ldots, n$. For example,

$$A_3 = -\begin{bmatrix} 1 & 1 & 1 \\ -3 & 3 & 3 \\ 5 & -5 & 5 \end{bmatrix}.$$

Next, we derive an expression for the eigenvalues of L. λ is an eigenvalue of L if it satisfies

$$\begin{bmatrix} A_n & 0 & 0 & \ldots & 0 & 0 & D_n \\ D_n & A_n & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & D_n & A_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix}, \tag{2.14}$$

where $\mathbf{v}^T = [\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_N^T]$ is the corresponding eigenvector and its components $\mathbf{v}_j,\ j = 1, 2, \ldots, N$, are column vectors of length $n = p + 1$. Equivalently, we can write equation (2.14) as

$$D_n \mathbf{v}_{j-1} + A_n \mathbf{v}_j = \lambda \mathbf{v}_j, \quad j = 1, 2, \ldots, N, \tag{2.15}$$

with an understanding that $\mathbf{v}_0 = \mathbf{v}_N$. We express D_n defined by (2.7) as an outer product $D_n = \mathbf{r}_n[1, 1, ..., 1]$, where $\mathbf{r}_n = [1, -3, \ldots, (-1)^{n-1}(2n-1)]^T$. Then (2.15) can be rewritten as

$$\mathbf{r}_n[1, 1, ..., 1]\mathbf{v}_{j-1} = (\lambda I - A_n)\mathbf{v}_j. \tag{2.16}$$

Introducing a new variable $S_j = [1, 1, ..., 1] \cdot \mathbf{v}_j$, we write (2.16) as

$$S_{j-1}\mathbf{r}_n = (\lambda I - A_n)\mathbf{v}_j. \tag{2.17}$$

Multiplying both sides of (2.17) by $[1, 1, ..., 1](\lambda I - A_n)^{-1}$ yields

$$S_j = S_{j-1}[1, 1, ..., 1](\lambda I - A_n)^{-1}\mathbf{r}_n. \tag{2.18}$$

Let

$$f_n(\lambda) = [1, 1, ..., 1](\lambda I - A_n)^{-1}\mathbf{r}_n. \tag{2.19}$$

Then, (2.18) results in a recursive formula

$$S_j = f_n(\lambda)S_{j-1}. \tag{2.20}$$

Expansion of (2.20) starting with $j = N$ gives

$$S_N = f_n^{N-1}(\lambda)S_1. \tag{2.21}$$

Finally, taking into account periodicity of the boundary conditions, we obtain $S_N = f_n^N(\lambda)S_N$. This implies

$$f_n^N(\lambda) = 1. \tag{2.22}$$

Then, the eigenvalues of L are the roots of the equations

$$f_n(\lambda) = \omega_j, \quad \omega_j = e^{\frac{2\pi i}{N}j}, \quad j = 0, 1, 2, \ldots, N-1. \tag{2.23}$$

Eigenvectors. For completeness of this discussion, we derive the eigenvectors of matrix L. Since L is a block circulant matrix, we look for eigenvectors $\mathbf{v}$ in the form $[\tilde{\mathbf{v}}^T, \omega_k\tilde{\mathbf{v}}^T, \ldots, \omega_k^{N-1}\tilde{\mathbf{v}}^T]^T$. Substituting $\mathbf{v}$ into (2.14) gives

$$\omega_k^{j-1}D_n\tilde{\mathbf{v}} + \omega_k^j A_n\tilde{\mathbf{v}} = \lambda_k\omega_k^j\tilde{\mathbf{v}}, \quad 1 \leqslant j \leqslant N, \tag{2.24}$$

or

$$(\omega_k\lambda_k I - \omega_k A_n - D_n)\tilde{\mathbf{v}} = 0, \tag{2.25}$$

where λ_k is one of the roots of $f_n(\lambda) = \omega_k$. $\tilde{\mathbf{v}}$ can be easily obtained by solving the linear system (2.25). The solutions are not particularly illuminating and we do not report them. Figure 2.1 shows the periodic property of the components of the eigenvectors. We plot one of the two eigenvectors corresponding to $k = 4$ (left) and $k = 17$ (right). In figure 2.1 each point corresponds to an entry in $\mathbf{v}$. The entries of the eigenvectors represent sampling of a scaled unit circle at N

or, if N/k is an integer, N/k points. One of the circles in Figure 2.1, left and right, corresponds to the first entry of $\omega_k^j \tilde{\mathbf{v}}$, $j = 1, 2, \ldots, N-1$, and the other to the second entry of $\omega_k^j \tilde{\mathbf{v}}$. The line connecting two points represents two consecutive entries of $\tilde{\mathbf{v}}$ and, thus, the shift in sampling between the two components of each $\omega_k^j \tilde{\mathbf{v}}$, $k = 1, 2, \ldots, N-1$,.

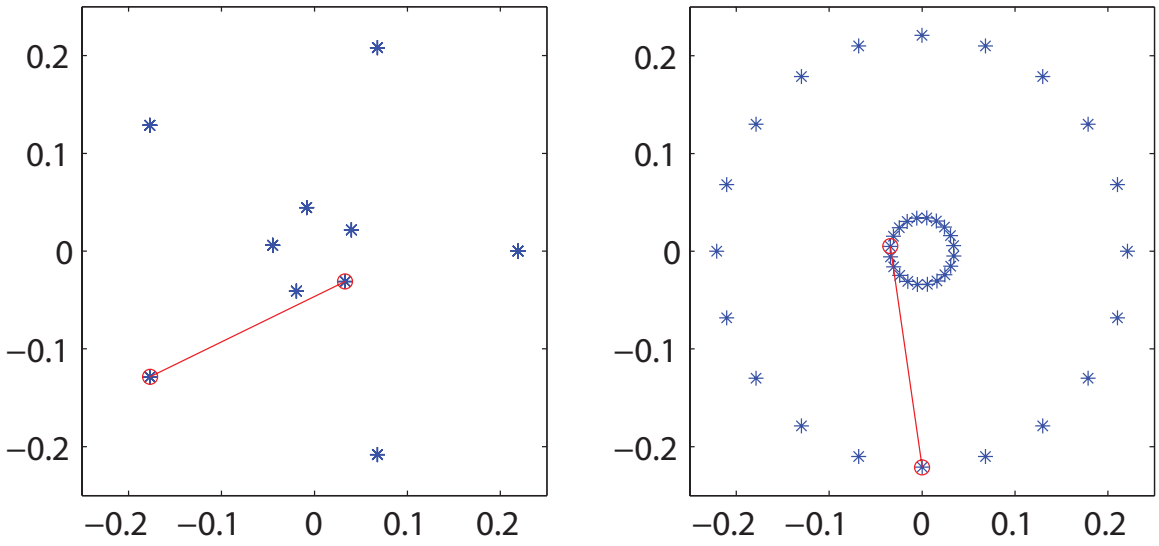

Figure 2.1: Eigenvectors of L with $N = 20$, $p = 1$, and $k = 4$ (left) and $k = 17$ (right). Each point in plots correspond to an entry in an eigenvector. The two points connected by a line show the first two entries of $\tilde{\mathbf{v}}$.

2.3 Padé approximants

Before deriving a general formulation of function $f_n(z)$, we will give a short introduction to Padé approximants. Let us suppose that we are given a power series representing a function $g(z) = \sum_{i=0}^{\infty} c_i z^i$. A Padé approximant is a rational function

$$[L/M] = \frac{a_0 + a_1 z + \cdots + a_L z^L}{b_0 + b_1 z + \cdots + b_M z^M}, \tag{2.26}$$

that satisfies

$$\sum_{i=0}^{\infty} c_i z^i = \frac{a_0 + a_1 z + \cdots + a_L z^L}{b_0 + b_1 z + \cdots + b_M z^M} + O(z^{L+M+1}). \tag{2.27}$$

Coefficients $a_0, a_1, \ldots, a_L$, and $b_0, b_1, \ldots, b_M$ are uniquely defined by $c_0, c_1, \ldots$, if b_0 is fixed. The approximant is usually scaled so that $b_0 = 1$. It is a common practice to display the approximants in a table, which is called the Padé table. We show a part of the Padé table of $g(z) = e^z$ in Table 2.1.

Table 2.1: Part of the Padé table of e^z [4].

	0	1	2	3
0	$\frac{1}{1}$	$\frac{1+z}{1}$	$\frac{2+2z+z^2}{2}$	$\frac{6+6z+3z^2+z^3}{6}$
1	$\frac{1}{1-z}$	$\frac{2+z}{2-z}$	$\frac{6+4z+z^2}{6-2z}$	$\frac{24+18z+6z^2+z^3}{24-6z}$
2	$\frac{2}{2-2z+z^2}$	$\frac{6+2z}{6-4z+z^2}$	$\frac{12+6z+z^2}{12-6z+z^2}$	$\frac{60+36z+9z^2+z^3}{60-24z+3z^2}$
3	$\frac{6}{6-6z+3z^2-z^3}$	$\frac{24+6z}{24-18z+6z^2-z^3}$	$\frac{60+24z+3z^2}{60-36z+9z^2-z^3}$	$\frac{120+60z+12z^2+z^3}{120-60z+12z^2-z^3}$

The Padé approximants of e^z are given by the following formula for non-negative integers p, q [4]

$$[p/q]_{exp(z)} = \frac{{}_1F_1(-p, -p-q, z)}{{}_1F_1(-q, -p-q, -z)}, \tag{2.28}$$

where ${}_1F_1$ denotes the confluent hypergeometric function defined by the series [1]

$${}_1F_1(a, b, z) = 1 + \frac{a}{b}z + \frac{a}{b}\frac{a+1}{b+1}\frac{z^2}{2!} + \frac{a}{b}\frac{a+1}{b+1}\frac{a+2}{b+2}\frac{z^3}{3!} + \cdots. \tag{2.29}$$

When a, b are negative integers and $b \leqslant a$, ${}_1F_1(a, b, z)$ is a finite sum which is a polynomial of degree $|a|$. Using the Pochhammer's symbol, $(a)_k = a(a+1)\cdots(a+k-1)$ and $(a)_0 = 1$, we can rewrite (2.29) in a compact form

$${}_1F_1(a, b, z) = \sum_{k=0}^{\infty} \frac{(a)_k}{(b)_k}\frac{z^k}{k!}. \tag{2.30}$$

2.4 Padé approximants and DG spatial discretization

In the following theorem, we state the main result of this section.

Theorem 3. *If A_n is an $n \times n$ matrix given by (2.13), and $f_n(z) = (1, ..., 1)(zI - A_n)^{-1}\mathbf{r}_n$, where $\mathbf{r}_n = \left[1, -3, \cdots, (-1)^{n-1}(2n-1)\right]^T$, then*

$$f_n(z) = \frac{{}_1F_1(-n+1, -2n+1, -z)}{{}_1F_1(-n, -2n+1, z)}, \tag{2.31}$$

which is the $[n-1/n]$ Padé approximant of e^{-z}.

In order to prove the theorem, we will need to establish three auxiliary results which are proved in the following three lemmas. We start by introducing additional notation.

Definition 1.

- *$\tilde{A}_n$: a matrix defined as $\tilde{A}_n = zI - A_n$.*
- *$M_{n,i}$: the (n, i) minor of $\tilde{A}_n$, i.e. the determinant of the $(n-1) \times (n-1)$ matrix obtained by elimination of the n-th row and i-th column of $\tilde{A}_n$, $i = 1, 2, \ldots, n$.*
- *$\tilde{A}_n^j$: the $n \times n$ matrix obtained by replacing the j-th column of $\tilde{A}_n$ with $\mathbf{r}_n$, $j = 1, 2, ..., n$.*
- *M_n^j: the determinant of $\tilde{A}_n^j$, $j = 1, 2, \ldots, n$.*
- *$M_{n,i}^j$: the (n, i) minor of the matrix $\tilde{A}_n^j$, $i, j = 1, 2, \ldots, n$.*

We also introduce two sequences of polynomials which are essential to our proofs

$$\begin{cases} Q_n(z) &= (a_n + z)Q_{n-1}(z) + a_n R_{n-1}(z), \\ R_n(z) &= a_n Q_{n-1}(z) + (a_n - z)R_{n-1}(z), \end{cases} \tag{2.32}$$

where $Q_1(z) = a_1 + z$, $R_1(z) = a_1$, and $a_n = 2n - 1$. As an example, $Q_n(z)$ and $R_n(z)$ for small n are listed in Table 2.2. Note that while Q_n is a polynomial of degree n, R_n is a polynomial of degree $n-1$. We will show that Q_n and R_n are proportional to the hypergeometric functions appearing in (2.31) and give an alternative expression for $f_n(z)$

$$f_n(z) = \frac{R_n(-z)}{Q_n(z)}. \tag{2.33}$$

Thus, (2.32) is a recursive formula for generating $[p/p+1]$ and $[p+1/p]$ (sub- and superdiagonal) Padé approximants for $exp(-z)$.

Table 2.2: Polynomials $Q_n(z)$ and $R_n(-z)$ defined in (2.32)

n	$R_n(-z)$	$Q_n(z)$
2	$6(1-\frac{1}{3}z)$	$6(1+\frac{2}{3}z+\frac{1}{6}z^2)$
3	$60(1-\frac{2}{5}z+\frac{1}{20}z^2)$	$60(1+\frac{3}{5}z+\frac{3}{20}z^2+\frac{1}{60}z^3)$
4	$840(1-\frac{3}{7}z+\frac{1}{14}z^2-\frac{1}{210}z^3)$	$840(1+\frac{4}{7}z+\frac{1}{7}z^2+\frac{2}{105}z^3+\frac{1}{840}z^4)$
5	$15120(1-\frac{4}{9}z+\frac{1}{12}z^2-\frac{1}{126}z^3+\frac{1}{3024}z^4)$	$15120(1+\frac{5}{9}z+\frac{5}{36}z^2+\frac{5}{252}z^3+\frac{5}{3024}z^4+\frac{1}{15120}z^5)$

We start with Lemma 1 which relates $Q_n(z)$ and $R_n(z)$ to the determinant of $\tilde{A}_n$ and its minors.

Lemma 1. *Let $Q_n(z)$ and $R_n(z)$ be defined by (2.32). Then*

$$Q_n(z) = \det(\tilde{A}_n), \tag{2.34a}$$

$$R_n(z) = \sum_{i=1}^{n} M_{n+1,i}. \tag{2.34b}$$

Proof. We will use an induction argument to prove (2.34). By Definition 1 and (2.32), $Q_1(z) = a_1 + z = \det(\tilde{A}_1)$, and $R_1(z) = a_1 = M_{2,1}$. This establishes the base of the induction. We assume that (2.34) holds for $Q_n(z), R_n(z)$, and we will prove that it is valid for $Q_{n+1}(z), R_{n+1}(z)$.

Applying the cofactor expansion along the $(n+1)$-th row of $\det(\tilde{A}_{n+1})$ while noticing that $M_{n+1,n+1} = \det(\tilde{A}_n)$ yields

$$\begin{aligned} \det(\tilde{A}_{n+1}) &= \sum_{i=1}^{n}(-1)^{n+1-i}a_{n+1}(-1)^{n+1+i}M_{n+1,i} + (a_{n+1}+z)M_{n+1,n+1} \\ &= a_{n+1}\sum_{i=1}^{n} M_{n+1,i} + (a_{n+1}+z)\det(\tilde{A}_n) \\ &= (a_{n+1}+z)Q_n(z) + a_{n+1}R_n(z) = Q_{n+1}(z). \end{aligned} \tag{2.35}$$

This proves the recursion (2.34a) for $Q_n(z)$.

Next, we prove (2.34b) for $R_{n+1}(z) = \sum_{i=1}^{n+1} M_{n+2,i}$. We begin by relating $M_{n+2,i}$ to $M_{n+1,i}$, $i < n+1$. In (2.36), we write an explicit expression for $M_{n+2,i}$, then subtract the $n+1$st column from the nth column and compute the determinant by a cofactor expansion based on the nth column,

$$M_{n+2,i} = \begin{vmatrix} a_1+z & \cdots & a_1 & a_1 \\ -a_2 & \cdots & a_2 & a_2 \\ \vdots & & \vdots & \vdots \\ (-1)^n a_{n+1} & \cdots & a_{n+1}+z & a_{n+1} \end{vmatrix} = \begin{vmatrix} a_1+z & \cdots & 0 & a_1 \\ -a_2 & \cdots & 0 & a_2 \\ \vdots & & \vdots & \vdots \\ (-1)^n a_{n+1} & \cdots & z & a_{n+1} \end{vmatrix} = -zM_{n+1,i}. \quad (2.36)$$

Similarly, for $i = n+1$, a cofactor expansion based on the last row yields

$$\begin{aligned} M_{n+2,n+1} &= \sum_{i=1}^{n} \left[(-1)^{n+1-i}(-1)^{n+1+i} a_{n+1} M_{n+1,i}\right] + a_{n+1} M_{n+1,n+1} \\ &= a_{n+1} \sum_{i=1}^{n} M_{n+1,i} + a_{n+1} M_{n+1,n+1}. \end{aligned} \quad (2.37)$$

Thus,

$$\begin{aligned} \sum_{i=1}^{n+1} M_{n+2,i} &= \sum_{i=1}^{n} (-z) M_{n+1,i} + a_{n+1} \sum_{i=1}^{n} M_{n+1,i} + a_{n+1} M_{n+1,n+1} \\ &= (a_{n+1} - z) \sum_{i=1}^{n} M_{n+1,i} + a_{n+1} M_{n+1,n+1} \\ &= (a_{n+1} - z) R_n(z) + a_{n+1} Q_n(z) \\ &= R_{n+1}(z). \end{aligned} \quad (2.38)$$

This completes the proof. □

Lemma 2 relates $R_n(z)$ and $Q_n(z)$ to the determinant of $\tilde{A}_n^j$ and its minors.

Lemma 2. *Let $R_n(z)$, $Q_n(z)$ be defined by (2.32). Then,*

$$R_n(-z) = \sum_{j=1}^{n} M_n^j, \tag{2.39a}$$

$$Q_n(-z) = \sum_{j=1}^{n+1} \left[\sum_{\substack{i=1 \\ i\neq j}}^{n} M_{n+1,i}^j + (-1)^{j-1} M_{n+1,j}^j \right]. \tag{2.39b}$$

Proof. The case $n = 1$ is satisfied trivially by the involved variables given by Definition 1 and (2.32). We assume that (2.39) is true for n, and we will prove it is also true for $n+1$.

Applying cofactor expansion to M_{n+1}^j, $j = 1, \ldots, n$ along the last row gives

$$\begin{aligned} M_{n+1}^j &= \sum_{\substack{i=1 \\ i\neq j}}^{n} \left[(-1)^{n+1-i} a_{n+1} (-1)^{n+1+i} M_{n+1,i}^j \right] + (-1)^n a_{n+1} (-1)^{n+1+j} M_{n+1,j}^j + (a_{n+1} + z) M_{n+1,n+1}^j \\ &= a_{n+1} \sum_{\substack{i=1 \\ i\neq j}}^{n} M_{n+1,i}^j + (-1)^{j-1} a_{n+1} M_{n+1,j}^j + (a_{n+1} + z) M_{n+1,n+1}^j. \end{aligned} \tag{2.40}$$

Similarly, applying cofactor expansion to M_{n+1}^{n+1} along the last row gives

$$\begin{aligned} M_{n+1}^{n+1} &= \sum_{i=1}^{n} (-1)^{n+1-i} a_{n+1} (-1)^{n+1+i} M_{n+1,i}^{n+1} + (-1)^n a_{n+1} M_{n+1,n+1}^{n+1} \\ &= a_{n+1} \sum_{i=1}^{n} M_{n+1,i}^{n+1} + (-1)^n a_{n+1} M_{n+1,n+1}^{n+1}. \end{aligned} \tag{2.41}$$

Since $M_{n+1,n+1}^j = M_n^j$, we can write

$$\begin{aligned} \sum_{j=1}^{n+1} M_{n+1}^j &= (a_{n+1} + z) \sum_{j=1}^{n} M_n^j + a_{n+1} \sum_{j=1}^{n+1} \left[\sum_{\substack{i=1 \\ i\neq j}}^{n} M_{n+1,i}^j + (-1)^{j-1} M_{n+1,j}^j \right] \\ &= (a_{n+1} + z) R_n(-z) + a_{n+1} Q_n(-z) \\ &= R_{n+1}(-z). \end{aligned} \tag{2.42}$$

This proves (2.39a).

We split the proof of (2.39b) into 2 parts: $j = 1, 2, \ldots, n$ and $j = n+1, n+2$. For $j = 1, \ldots, n$,

using an argument similar to one employed in (2.36), we can derive $M^j_{n+2,i} = (-z)M^j_{n+1,i},\ i = 1, 2, ..., n$. This with a cofactor expansion on the last row of $M^j_{n+2,n+1}$ gives

$$\begin{aligned}\sum_{\substack{i=1\\ i\neq j}}^{n+1} M^j_{n+2,i} + (-1)^{j-1}M^j_{n+2,j} &= \sum_{\substack{i=1\\ i\neq j}}^{n} M^j_{n+2,i} + (-1)^{j-1}M^j_{n+2,j} + M^j_{n+2,n+1}\\ &= (-z)\left[\sum_{\substack{i=1\\ i\neq j}}^{n} M^j_{n+1,i} + (-1)^{j-1}M^j_{n+1,j}\right]\\ &\quad +a_{n+1}\left[\sum_{\substack{i=1\\ i\neq j}}^{n} M^j_{n+1,i} + (-1)^{j-1}M^j_{n+1,j}\right] + a_{n+1}M^j_{n+1,n+1}.\end{aligned} \tag{2.43}$$

For $j = n+1, n+2$, we switch the last two columns of $M^{n+2}_{n+2,i}$ to obtain

$$\begin{aligned}M^{n+2}_{n+2,i} &= \begin{vmatrix} a_1+z & \cdots & a_1 & a_1\\ -a_2 & \cdots & a_2 & -a_2\\ \vdots & & \vdots & \vdots\\ (-1)^n a_{n+1} & \cdots & a_{n+1}+z & (-1)^n a_{n+1}\end{vmatrix}\\ &= -\begin{vmatrix} a_1+z & \cdots & a_1 & a_1\\ -a_2 & \cdots & -a_2 & a_2\\ \vdots & & \vdots & \vdots\\ (-1)^n a_{n+1} & \cdots & (-1)^n a_{n+1} & a_{n+1}+z\end{vmatrix}\end{aligned} \tag{2.44}$$

Comparing $-M^{n+2}_{n+2,i}$ with $M^{n+1}_{n+2,i}$ reveals that the entries in the determinants are identical except for the $(n+1, n+1)$ element, which is $(a_{n+1}+z)$ in $-M^{n+2}_{n+2,i}$ and a_{n+1} in $M^{n+1}_{n+2,i}$. Expanding the determinants along the last rows of $M^{n+1}_{n+2,i}$ and $M^{n+2}_{n+2,i}$ and adding up the results, we have

$$M^{n+1}_{n+2,i} + M^{n+2}_{n+2,i} = (-z)M^{n+1}_{n+1,i},\ i = 1, 2, ..., n. \tag{2.45}$$

A similar observation gives

$$M^{n+1}_{n+2,n+1} + M^{n+2}_{n+2,n+2} = (-z)M^{n+1}_{n+1,n+1}. \tag{2.46}$$

Combining (2.45) and (2.46) and using a cofactor expansion on $M^{n+2}_{n+2,n+1}$ along the last row, we

obtain

$$\begin{aligned}
&\sum_{j=n+1}^{n+2}\sum_{i=1}^{n} M^{j}_{n+2,i} + (-1)^{n} M^{n+1}_{n+2,n+1} + M^{n+2}_{n+2,n+1} + (-1)^{n+1} M^{n+2}_{n+2,n+2} \\
= \;& (-z)\sum_{i=1}^{n} M^{n+1}_{n+1,i} + (-1)^{n}(a_{n+1}-z) M^{n+1}_{n+1,n+1} + a_{n+1}\sum_{i=1}^{n} M^{n+1}_{n+1,i}.
\end{aligned} \tag{2.47}$$

Finally, combining (2.43) and (2.47) yields the result

$$\begin{aligned}
\sum_{j=1}^{n+2}\left[\sum_{\substack{i=1\\ i\neq j}}^{n+1} M^{j}_{n+2,i} + (-1)^{j-1} M^{j}_{n+2,j}\right] &= (a_{n+1}-z)\sum_{j=1}^{n+1}\left[\sum_{\substack{i=1\\ i\neq j}}^{n} M^{j}_{n+1,i} + (-1)^{j-1} M^{j}_{n+1,j}\right] \\
&\quad + a_{n+1}\sum_{j=1}^{n} M^{j}_{n+1,n+1} \\
&= (a_{n+1}-z) Q_n(-z) + a_{n+1} R_n(-z) \\
&= Q_{n+1}(-z),
\end{aligned} \tag{2.48}$$

which completes the proof. □

Lemma 3 relates polynomials $Q_n(z)$ and $R_n(z)$ to the confluent hypergeometric functions.

Lemma 3. *Let $Q_n(z)$ and $R_n(z)$ be polynomials defined by (2.32). Then,*

$$Q_n(z) = \prod_{i=1}^{n} a_i 2^{n-1}\, {}_1F_1(-n, -2n+1, z), \tag{2.49a}$$

$$R_n(z) = \prod_{i=1}^{n} a_i 2^{n-1}\, {}_1F_1(-n+1, -2n+1, z). \tag{2.49b}$$

Proof. When $n = 1$, (2.49) is validated by Definition 1 and (2.32). Assuming that (2.49) is true for $Q_n(z), R_n(z)$, we will show it is also true for $Q_{n+1}(z), R_{n+1}(z)$. We start with (2.49a). By the

recurrence relation (2.32) and the assumption that (2.49) is true for $Q_n(z), R_n(z)$, we have

$$\begin{aligned}
Q_{n+1}(z) &= (a_{n+1}+z)Q_n(z) + a_{n+1}R_n(z) \\
&= (a_{n+1}+z)\prod_{i=1}^{n} a_i 2^{n-1}\,{}_1F_1(-n,-2n+1,z) + a_{n+1}\prod_{i=1}^{n} a_i 2^{n-1}\,{}_1F_1(-n+1,-2n+1,z) \\
&= \prod_{i=1}^{n} a_i 2^{n-1}\left[(a_{n+1}+z)\,{}_1F_1(-n,-2n+1,z) + a_{n+1}\,{}_1F_1(-n+1,-2n+1,z)\right] \\
&= \prod_{i=1}^{n} a_i 2^{n-1}\left[(a_{n+1}+z)\sum_{k=0}^{n}\frac{(-n)_k}{(-2n+1)_k}\frac{z^k}{k!} + a_{n+1}\sum_{k=0}^{n-1}\frac{(-n+1)_k}{(-2n+1)_k}\frac{z^k}{k!}\right].
\end{aligned} \tag{2.50}$$

Next, we collect the terms of the same degree k in (2.50) and simplify the obtained coefficients. The coefficients in front of z^k, $k = 2, 3, ..., n$,

$$\begin{aligned}
& a_{n+1}\frac{(-n)_k}{(-2n+1)_k}\frac{1}{k!} + a_{n+1}\frac{(-n+1)_k}{(-2n+1)_k}\frac{1}{k!} + \frac{(-n)_{k-1}}{(-2n+1)_{k-1}}\frac{1}{(k-1)!} \\
= \;& a_{n+1}(-n+k-1)\frac{(-n+1)_{k-2}}{(-2n+1)_{k-1}}\frac{1}{k!} + (-n)\frac{(-n+1)_{k-2}}{(-2n+1)_{k-1}}\frac{1}{(k-1)!} \\
= \;& [a_{n+1}(-n+k-1) + (-n)(k)]\frac{(-n+1)_{k-2}}{(-2n+1)_{k-1}}\frac{1}{k!} \\
= \;& [(2n+1)(-n+k-1) + (-n)(k)]\frac{(-n+1)_{k-2}}{(-2n+1)_{k-1}}\frac{1}{k!} \\
= \;& (-2n+k-1)(n+1)\frac{(-n+1)_{k-2}}{(-2n+1)_{k-1}}\frac{1}{k!} \\
= \;& (-2n+k-1)(n+1)\frac{(-2n-1)(-2n)}{(-n-1)(-n)(-2n+k-1)}\frac{(-n-1)_k}{(-2n-1)_k}\frac{1}{k!} \\
= \;& 2a_{n+1}\frac{(-n-1)_k}{(-2n-1)_k}\frac{1}{k!}.
\end{aligned} \tag{2.51}$$

For the constant term, $k = 0$, we have

$$a_{n+1}\cdot 1 + a_{n+1}\cdot 1 = 2a_{n+1}\cdot 1. \tag{2.52}$$

For the term of degree 1,

$$a_{n+1}\frac{-n}{-2n+1} + 1 + a_{n+1}\frac{-n+1}{-2n+1} = 2(n+1) = 2a_{n+1}\frac{-n-1}{-2n-1}. \tag{2.53}$$

For the term of degree $n+1$

$$\frac{(-n)_n}{(-2n+1)_n}\frac{1}{n!} = \frac{(-2n-1)(-2n)}{(-n-1)(-n)}\frac{(-n-1)_{n+1}}{(-2n-1)_{n+1}}\frac{1}{n!} = 2a_{n+1}\frac{(-n-1)_{n+1}}{(-2n-1)_{n+1}}\frac{1}{(n+1)!}. \tag{2.54}$$

Inserting (2.51)-(2.54) into (2.50), we obtain

$$Q_{n+1}(z) = \prod_{i=1}^{n+1} a_i 2^n \sum_{k=0}^{n+1} \frac{(-n-1)_k}{(-2n-1)_k}\frac{z^k}{k!} = \prod_{i=1}^{n+1} a_i 2^n {}_1F_1(-n-1,-2n-1,z). \tag{2.55}$$

Statement (2.49b) can be proven using a similar reasoning. To avoid repetition, the proof is omitted. Thus, we proved that (2.49) is valid for any $n \in \mathbb{N}$. □

Now we can complete the proof of Theorem 1.

Proof. Let $\mathbf{w}_n = \tilde{A}_n^{-1}\mathbf{r}_n$, i.e. $\tilde{A}_n\mathbf{w}_n = \mathbf{r}_n$. Using the Cramer's rule, $\mathbf{w}_n = \det(\tilde{A}_n)^{-1}[M_n^1, M_n^2, ..., M_n^n]^T$. Therefore,

$$\begin{aligned}
f_n(z) &= [1,1,...,1]\tilde{A}_n^{-1}\mathbf{r}_n \\
&= [1,1,...,1]\mathbf{w}_n \\
&= \frac{1}{\det(\tilde{A}_n)}\sum_{j=1}^{n} M_n^j \\
&= \frac{R_n(-z)}{Q_n(z)} && \text{(Lemma 1 and Lemma 2)} \\
&= \frac{{}_1\tilde{F}_1(-n+1,-2n+1,-z)}{{}_1F_1(-n,-2n+1,z)} && \text{(Lemma 3)}
\end{aligned}$$

□

2.5 Spectrum of the DG discretization

In the previous section we showed that the eigenvalues of the discontinuous Galerkin spatial discretization matrix L are given by

$$f_n(\lambda) = \omega_j, \quad \omega_j = e^{\frac{2\pi i}{N}j}, \quad j = 0,1,\ldots,N-1. \tag{2.56}$$

Then, using (2.33) the eigenvalues can be computed as n roots of

$$R_n(-\lambda) - \omega_j Q_n(\lambda) = 0 \tag{2.57}$$

for each $0 \leq j \leq N-1$.

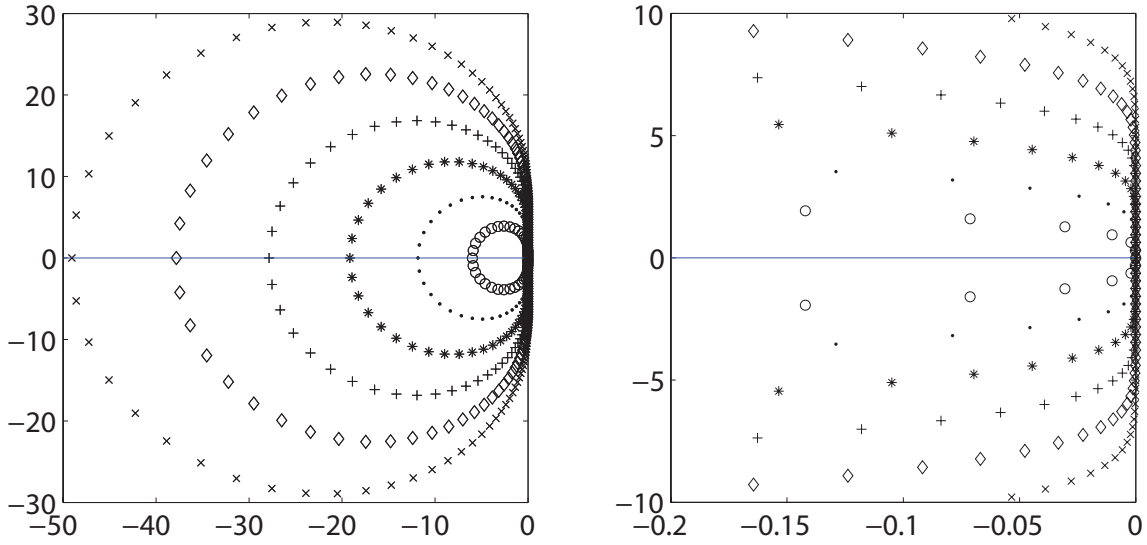

Figure 2.2: Eigenvalues of L for $p = 1, 2, 3, 4, 5, 6$ with $N = 20$. The inner curve corresponds to $p = 1$ and the outer curve corresponds to $p = 6$. The right figure is a zoom of the left one.

The computed eigenvalues with $p = 1, 2, \ldots, 6$ on a twenty cell mesh are shown in Figure 2.2, left. We observe that the size of the spectrum grows with p. At the same time the curves $|f_{p+1}(z)| = 1$ (of which (2.56) is a discrete approximation) seem to approach and flatten near the imaginary axis (Fig. 2.2, right) as p increases.

The leftmost eigenvalues are mainly responsible for the decrease of the CFL number with increasing order of approximation and can be used as a proxy for the size of the spectrum. We investigate the growth rate of the eigenvalue with the largest magnitude later in this section. Flattening of the spectral curves near the imaginary axis means that for the fully discrete method to be stable with a constant (mesh size independent) CFL number, the absolute stability region of the time integration scheme should include a sufficiently large part of the imaginary axis. For example, the DG with $p > 0$ is not stable with a fixed CFL number with the forward Euler time stepping or $p > 1$ and a two stage second order Runge-Kutta schemes [14]. However, despite approaching the imaginary axis, the eigenvalues are never purely imaginary when the upwind flux is used in the DG discretization. This is proven in the following lemma and theorem.

Lemma 4. *Let $Q_n(z)$ and $R_n(z)$ be polynomials defined by (2.32). Then,*

$$Q_n(\beta i)Q_n(-\beta i) = R_n(\beta i)R_n(-\beta i) + \beta^{2n}, \quad \beta \in \mathbb{R}, \quad n = 1, 2, \ldots. \tag{2.58}$$

Proof. When $\beta = 0$, it follows from (2.49) and the definition (2.29) that $Q_n(0) = R_n(0)$. Then, (2.58) is trivially true.

When $\beta \neq 0$, we will use the mathematical induction on n to prove (2.58). For $n = 1$, from (2.32), we obtain

$$Q_1(\beta i)Q_1(-\beta i) = (1 + \beta i)(1 - \beta i) = 1 + \beta^2 = R_1(\beta i)R_1(-\beta i) + \beta^2, \tag{2.59}$$

which establishes the basis of induction. We assume (2.58) is valid for n, and we will prove it consequently holds for $n + 1$. Using (2.32) yields

$$\begin{aligned} Q_{n+1}(\beta i)Q_{n+1}(-\beta i) &= [(a_{n+1} + \beta i)Q_n(\beta i) + a_{n+1}R_n(\beta i)]\,[(a_{n+1} - \beta i)Q_n(-\beta i) + a_{n+1}R_n(-\beta i)] \\ &= (a_{n+1}^2 + \beta^2)Q_n(\beta i)Q_n(-\beta i) + a_{n+1}^2 R_n(\beta i)R_n(-\beta i) \\ &\quad + (a_{n+1}^2 + a_{n+1}\beta i)Q_n(\beta i)R_n(-\beta i) + (a_{n+1}^2 - a_{n+1}\beta i)Q_n(-\beta i)R_n(\beta i). \end{aligned} \tag{2.60}$$

$$\begin{aligned} R_{n+1}(\beta i)R_{n+1}(-\beta i) &= [a_{n+1}Q_n(\beta i) + (a_{n+1} - \beta i)R_n(\beta i)][a_{n+1}Q_n(-\beta i) + (a_{n+1} + \beta i)R_n(-\beta i)] \\ &= a_{n+1}^2 Q_n(\beta i)Q_n(-\beta i) + (a_{n+1}^2 + \beta^2)R_n(\beta i)R_n(-\beta i) \\ &\quad + (a_{n+1}^2 + a_{n+1}\beta i)Q_n(\beta i)R_n(-\beta i) + (a_{n+1}^2 - a_{n+1}\beta i)Q_n(-\beta i)R_n(\beta i). \end{aligned} \tag{2.61}$$

Thus,

$$\begin{aligned} Q_{n+1}(\beta i)Q_{n+1}(-\beta i) - R_{n+1}(\beta i)R_{n+1}(-\beta i) &= \beta^2[Q_n(\beta i)Q_n(-\beta i) - R_n(\beta i)R_n(-\beta i)] \\ &= \beta^{2(n+1)}, \end{aligned} \tag{2.62}$$

which completes the proof. □

Theorem 4. *Equation (2.56) has no pure imaginary roots.*

Proof. We start by observing that for any polynomial $p(z)$ with real coefficients the following holds

$$\overline{p(\beta i)} = p(-\beta i). \tag{2.63}$$

Next, let us assume that $z = \beta i$ is a pure imaginary root of (2.56), where $\beta \neq 0$ is a real number.

Substitute $z = \beta i$ into (2.33) and take the modulus to obtain

$$\begin{aligned} |f_n(\beta i)|^2 &= f_n(\beta i)\overline{f_n(\beta i)} \\ &= f_n(\beta i) f_n(-\beta i) \\ &= \frac{R_n(\beta i) R_n(-\beta i)}{Q_n(\beta i) Q_n(-\beta i)} \\ &= \frac{R_n(\beta i) R_n(-\beta i)}{R_n(\beta i) R_n(-\beta i) + \beta^{2n}} \quad \text{(Lemma 4)} \\ &= \frac{|R_n(\beta i)|^2}{|R_n(\beta i)|^2 + \beta^{2n}} < 1. \end{aligned} \tag{2.64}$$

Since $|f_n(z)| = 1$ is a necessary condition for z being a root of (2.56), $|f_n(\beta i)| < 1$ implies that βi is not a root of (2.56). □

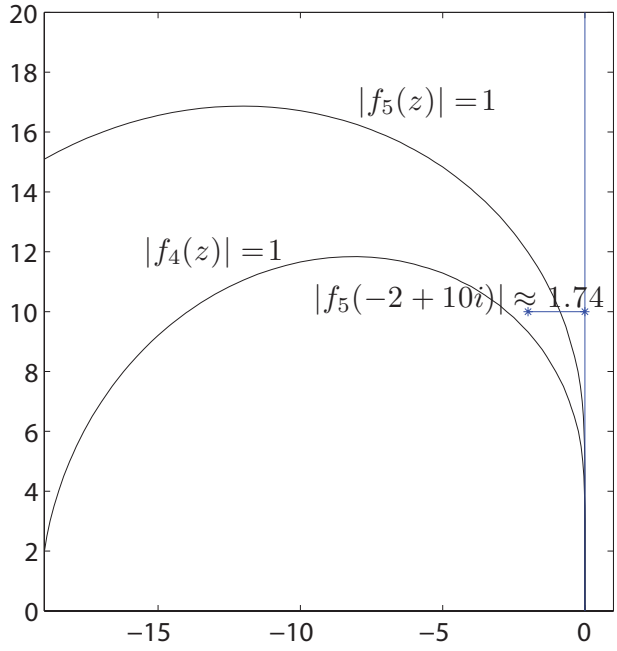

Figure 2.3: Illustration of $|f_n(z)| = 1$ approaching the imaginary axis. For a randomly chosen point $\hat{z} = -2 + 10i$, we can find $|f_5(\hat{z})| > 1$, so $|f_5(z)| = 1$ passes through the right of this point $\hat{z}$.

Padé approximants of the exponential function converge to the exponent at every point in the complex plane [30], p.531-536. If we pick an arbitrary point $z = \alpha + \beta i,\ \alpha < 0$, from the left half plane, we have

$$|\lim_{n\to\infty} f_n(z)| = |e^{-z}| > 1, \tag{2.65}$$

i.e. there exists a sufficiently large N such that $|f_N(z)| > 1$. In the proof of Theorem 2, we showed that $|f_N(\beta i)| < 1$. Assuming that $|f_N(z)|$ is analytic on the line $\mathrm{Im}\, z = \beta$ (since there exists only a finite number of poles of $|f_N(z)|$ this is not a restrictive assumption), there exists a point $z' = \alpha' + \beta i$, where $\alpha < \alpha' < 0$, such that $|f_N(z')| = 1$. This implies that the curve $|f_N(z)| = 1$ goes across the region between $z = \alpha + \beta i$ and βi. Since the point z is randomly chosen from the left half plane, we can conclude that for any point close enough to the imaginary axis, there exists a curve which is even closer to the imaginary axis. Thus, we demonstrated that $|f_n(z)| = 1$ approach the imaginary axis as n grows. The reasoning is illustrated for a particular choice of a point z and $n = 5$ in Figure 2.3.

Next, we analyze the growth of the eigenvalue largest in modulus. We conjecture it to be the real eigenvalue located on the leftmost part of the spectral curves $|f_n(z)| = 1$ in Figure 2.2. All roots of (2.56) are located in the left half of the complex plane. This can be seen from Theorem 4.12 in [28], which states that the curve $|R(z)| = 1$, where $R(z)$ is the $[p/p+1]$ Padé approximant of $exp(z)$ is located in the right half of the complex plane. Since $f_n(z)$ is the same approximant to $exp(-z)$, $|f_n(z)| = 1$ is a mirror image of $|R(z)| = 1$ with respect to the imaginary axis, and the result follows. Below we make a few simple statements about the roots of (2.56).

Proposition 1. *Equation (2.56) always has a zero root.*

Proof. From (2.49) and (2.29), the zero order coefficients in $R(-z)$ and $Q(z)$ are the same and are equal to $\prod_{i=1}^{n} a_i 2^{n-1}$. Using (2.33), $f_n(0) = R(0)/Q(0) = 1$. □

Proposition 2. *The real roots of $f_n(z) = \omega_k$ correspond to $\omega_k = \pm 1$.*

Proof. Consider $f_n(z) = \omega_k$ with a real z. Since f_n is a rational function with real coefficients, the right hand side must be a real number. Hence, $\omega_k = \pm 1$. □

Depending on the number of mesh cells N and the order of approximation p, there might be one real root or a couple of complex conjugate numbers on the left most part of the curve. This is not essential as (2.56) is a discrete version of $|f_n(z)| = 1$. Below we state the conditions for (2.56) to have a real negative root which we will denote by z^* and without loss of generality we will assume that the mesh is such that it exists.

Proposition 3. *For a discretization with an even number of cells N, (2.56) has at least one non-zero real root.*

Proof. If n is an odd number, we rewrite $f_n(z) = -1$ as $R(-z) + Q(z) = 0$. Since the zero order coefficient in $R(-z)$ and $Q(z)$ is the same (Proposition 1), zero is not a root of $R(-z) + Q(z) = 0$. Since n is odd, there exists at least one non-zero real root.

If n is an even number, we rewrite $f_n(z) = 1$ as $R(-z) - Q(z) = 0$. Since the zero order coefficient in $R(-z)$ and $Q(z)$ is the same, $R(-z) - Q(z) = 0$ can be expressed as $zr(z) = 0$, where $r(z)$ is a real polynomial of degree $n-1$. Consequently, it should have one real root which cannot be zero because, as shown in (2.49) and (2.29), the first order coefficients of $R(-z)$ and $Q(z)$ are not the same. □

For an odd number of cells N, an odd degree approximation (even n) results in at least one nonzero real root. With an even degree of approximation, we conjecture that the only real root is zero. Since the eigenvalues always locate on $|f_n(z)| = 1$, which does not depend on N, we can assume N is even for analyzing the size of the spectrum.

If the negative real root z^* exists, it is conjectured to have the largest modulus, and this largest modulus also performs as a bound of all roots when z^* does not exist. Next, we will derive a bound on z^*.

Using (2.31) to write (2.56) in a polynomial form

$$p(z) = {}_1F_1(-n+1, -2n+1, -z) - e^{\frac{2\pi i}{N}j}\,{}_1F_1(-n, -2n+1, z), \tag{2.66}$$

and collecting the terms of the same order, we obtain

$$p(z) = c_n z^n + c_{n-1} z^{n-1} + \cdots + c_1 z + c_0, \tag{2.67}$$

where

$$\begin{aligned} c_n &= -e^{\frac{2\pi i}{N}j} \frac{(-n)_n}{(-2n+1)_n} \frac{1}{n!} \\ c_{n-1} &= -e^{\frac{2\pi i}{N}j} \frac{(-n)_{n-1}}{(-2n+1)_{n-1}} \frac{1}{(n-1)!} + \frac{(-n+1)_{n-1}}{(-2n+1)_{n-1}} \frac{1}{(n-1)!} \\ &\vdots \\ c_0 &= -e^{\frac{2\pi i}{N}j} + 1. \end{aligned} \tag{2.68}$$

Since the sum of all the roots of $p(z) = 0$ satisfies

$$\sum_{i=1}^{n} z_i = -\frac{c_{n-1}}{c_n} = -n^2 + n e^{-\frac{2\pi i}{N}j}, \tag{2.69}$$

Table 2.3: Real eigenvalues of L on a two cell grid.

p	1	2	3	4	5	6
$-z^*$	6	11.8424	19.1569	27.8419	37.8247	49.0518
$(p+1)(p+2)$	6	12	20	30	42	56
p	7	8	9	10	11	12
$-z^*$	61.4815	75.0797	89.8181	105.6720	122.6204	140.6442
$(p+1)(p+2)$	72	90	110	132	156	182
p	13	14	15	16	17	18
$-z^*$	159.7268	179.8529	201.0087	223.1817	246.3603	270.5337
$(p+1)(p+2)$	210	240	272	306	342	380
p	19	20	21	22	23	24
$-z^*$	295.6920	321.8258	348.9264	376.9857	405.9960	435.9500
$(p+1)(p+2)$	420	462	506	552	600	650

we obtain that $\mathrm{Re}(-\frac{c_{n-1}}{c_n}) \geqslant -n(n+1)$. Noticing that all the roots have non-positive real parts, $-n(n+1)$ is a lower bound of the real part of all roots including z^* for all n. We are interested whether this bound is tight and reasonably well represents the growth speed of the largest root. Table 2.3 lists the roots of the largest modulus up to order 24, and Figure 2.4 shows the absolute value of these roots up to order 100. We see that the bound overestimates the roots especially for large n. Next, we show that the asymptotic growth rate is not quadratic. In particular, the following theorem proves that $-cn^2$ is not a root of (2.56) for all $c > 0$.

Theorem 5. *For all $c > 0$, $|f_n(-cn^2)| \propto \frac{1}{n}$ as $n \to \infty$.*

Proof. Consider the hypergeometric function ${}_1F_1(a, b, z)$ defined in (2.29) and (2.30). When a and b are negative integers, the function ${}_1F_1(a, b, z)$ is a polynomial of degree $|a|$. We factor out the term of the highest degree of z and define the function

$$G(a, b, z) = \frac{(b)_{|a|}}{(a)_{|a|}} \frac{|a|!}{z^{|a|}} {}_1F_1(a, b, z), \tag{2.70}$$

or, in an explicit form,

$$G(a, b, z) = \sum_{k=0}^{|a|} \frac{c_k}{z^k}, \quad c_k = C^k_{|a|}(a - b + 1)_k, \tag{2.71}$$

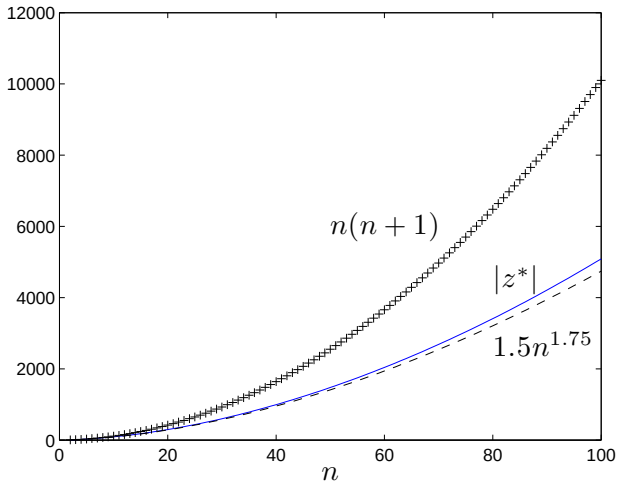

Figure 2.4: Absolute value of the negative real roots on a two cell grid, the upper bound $n(n+1)$ and the lower bound $1.5n^{1.75}$ as a function of $n = p + 1$.

where $C^k_{|a|}$ are binomial coefficients. Substituting (2.70) into (2.31) yields

$$f_n(z) = \frac{{}_1F_1(-n+1,-2n+1,-z)}{{}_1F_1(-n,-2n+1,z)} = \frac{(-1)^{n-1}n}{z}\frac{G(-n+1,-2n+1,-z)}{G(-n,-2n+1,z)}. \tag{2.72}$$

Next, we will prove that $\lim_{n\to\infty} G(-n,-2n+1,-cn^2) = e^{-\frac{1}{c}}$. Substituting $a = -n, b = -2n+1, z = -cn^2$ into (2.71) gives

$$\begin{aligned} G(-n,-2n+1,-cn^2) &= \sum_{k=0}^{n} C^k_n \frac{(n)_k}{(-cn^2)^k} \\ &= \sum_{k=0}^{n} \frac{n!}{k!(n-k)!}\frac{(n)_k}{(-cn^2)^k} \\ &= \sum_{k=0}^{n} \frac{(-1)^k}{c^k k!}[\frac{n!}{(n-k)!}\frac{(n)_k}{(n^2)^k}] \\ &= 1 + \sum_{k=1}^{n} \frac{(-1)^k}{c^k k!}(1-\frac{1}{n^2})(1-\frac{2^2}{n^2})\cdots(1-\frac{(k-1)^2}{n^2}). \end{aligned}$$

For simplicity, we call

$$\tilde{d}_k = \frac{(-1)^k}{c^k k!}, \quad e_0^n = 1, \quad e_k^n = (1 - \frac{0}{n^2})(1 - \frac{1}{n^2})(1 - \frac{2^2}{n^2}) \cdots (1 - \frac{(k-1)^2}{n^2}), \quad k = 1, 2, ..., \tag{2.73}$$

and define d_k^n as

$$d_k^n = \begin{cases} 1, & k = 0, \\ \dfrac{(-1)^k}{c^k k!}(1 - \dfrac{0}{n^2})(1 - \dfrac{1}{n^2})(1 - \dfrac{2^2}{n^2}) \cdots (1 - \dfrac{(k-1)^2}{n^2}) = \tilde{d}_k e_k^n, & k = 1, ..., n, \\ 0, & k > n. \end{cases} \tag{2.74}$$

We also define two partial sums

$$D_l^n = \sum_{k=0}^{l} d_k^n, \quad \tilde{D}_l = \sum_{k=0}^{l} \tilde{d}_k, \quad l = 0, 1, ... \tag{2.75}$$

Then, $G(-n, -2n + 1, -cn^2) = D_n^n$. Since for a fixed k $\lim\limits_{n\to\infty} e_k^n = 1$, we conclude that

$$\lim_{n\to\infty} d_k^n = \tilde{d}_k, \quad \forall k \geqslant 0. \tag{2.76}$$

Noticing that $\lim\limits_{l\to\infty} \tilde{D}_l = e^{-\frac{1}{c}}$, we have

$$\forall\ \epsilon > 0, \exists\ K_1 > 0, s.t. \forall\ k > K_1,\ \left|\tilde{D}_{K_1} - e^{-\frac{1}{c}}\right| < \epsilon. \tag{2.77}$$

And from the definition of $\tilde{d}_k$,

$$\exists\ K_2 > 0, s.t. \forall\ k > K_2, |\tilde{d}_{K_2+1}| < \epsilon. \tag{2.78}$$

Let $K = \max(K_1, K_2)$, then

$$\lim_{n\to\infty} D_K^n = \sum_{k=0}^{K} \lim_{n\to\infty} d_k^n = \sum_{k=0}^{K} \tilde{d}_k = \tilde{D}_K. \tag{2.79}$$

The expression above implies that

$$\exists N > K > 0,\ s.t. \forall\ n > N,\ |D_K^n - \tilde{D}_K| < \epsilon. \tag{2.80}$$

On the other hand, since $\{d_k^n\}$ have alternating signs while $|d_k^n|$ decrease as $k \to \infty$,

$$|D_K^n - D_n^n| < |d_{K+1}^n| < |\tilde{d}_{K+1}| < \epsilon. \; \forall \; n > K \tag{2.81}$$

Combining (2.77), (2.80) and (2.81), we obtain

$$|D_n^n - e^{-\frac{1}{c}}| \leqslant |D_n^n - D_K^n| + |D_K^n - \tilde{D}_K| + |\tilde{D}_K - e^{-\frac{1}{c}}| < 3\epsilon, \quad n > N,$$

i.e.

$$\lim_{n\to\infty} G(-n, -2n+1, -cn^2) = \lim_{n\to\infty} D_n^n = e^{-\frac{1}{c}}. \tag{2.82}$$

Using the same reasoning, we can prove that

$$\lim_{n\to\infty} G(-n+1, -2n+1, cn^2) = e^{\frac{1}{c}}. \tag{2.83}$$

Combining (2.82), (2.83) and (2.72) yields

$$\lim_{n\to\infty} |f_n(-cn^2)| = \lim_{n\to\infty} \left| \frac{G(-n+1, -2n+1, cn^2)}{G(-n, -2n+1, -cn^2)} \right| \frac{n}{cn^2} = \lim_{n\to\infty} e^{\frac{2}{c}} \frac{1}{cn} = 0. \tag{2.84}$$

□

We have proved that regardless of the constant c, $|f_n(cn^2)|$ is small for large enough n and consequently cannot be a root of (2.56). In other words, any quadratic function will overcome the curve $|z^*(n)|$ (Fig. 2.4). If we assume that the real root z^* grows as a power function $-cn^\alpha$, then for $\alpha > 2$, by following the steps in the proof of Theorem 2 we can show that $\lim_{n\to\infty} G(-n, -2n+1, -cn^\alpha) = 1$ and $\lim_{n\to\infty} G(-n+1, -2n+1, cn^\alpha) = 1$. So, in this case, $\lim_{n\to\infty} f_n(-cn^\alpha) = 0$ also implies $z^* = -cn^\alpha$ is not the proper estimate for the root. We conclude that the spectrum of L should grow slower than $-cn^\alpha$. Numerical experiments reveal that $-1.5n^{1.75}$ is an upper bound on z^* for all n (Fig. 2.4). Least square fitting $-cn^\alpha$ for the first one hundred roots gives $-1.4n^{1.78}$. Bounds on the eigenvalues for very large n are reported in Figure 2.5. The computations were performed for a two cell grid using MATLAB. They are believed to be accurate in the sense of small error in $f_n(z^*) - 1$.

Remark 1. We should mention that the Padé approximants $f_n(z)$ are only a good approximation to e^{-z} in regions close to the origin of the complex plane. In Figure 2.6 we plot $|f_{p+1}(z)|$ for real negative z with $p = 1, 2, 3$. The spike in the $p = 2$ plot is related to the nearby pole of $f_3(z)$. We note that this behavior, i.e. that the exponential function grows in magnitude while the Padé approximants decay to zero as $|z|$ increases, is similar for complex z but is more difficult to

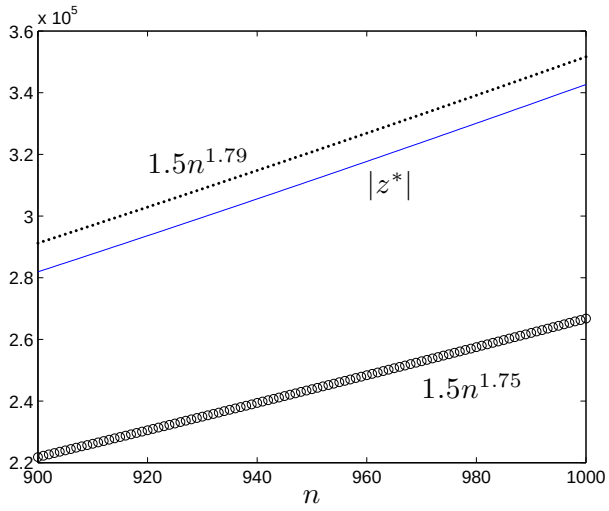

Figure 2.5: Absolute value of the negative real roots on a two cell grid and two bounds for large values of n.

illustrate. Comparing Figures 2.2 and 2.6 reveals that for many eigenvalues λ, $|f_n(\lambda)|$ is far from $e^{-\lambda}$. Although the region where $f_n(z) \approx e^{-z}$ grows with n, it grows slower than the eigenvalues. Consequently, the approximant f_n should not be viewed as an approximation of $exp(-z)$ as far as eigenvalues of L are concerned.

Remark 2. When nonperiodic boundary conditions are used, it is sufficient to analyze the zero inflow boundary conditions. Then, in the first row of (2.14) D_n is absent and we have

$$det(A_n - \lambda I) = 0. \tag{2.85}$$

From (2.85) and the third line of the proof of Theorem 1, λ must be the poles of f_n. It is easy to see that these λ are the only eigenvalues of L and that they have multiplicity N. Furthermore, matrix L has only $p + 1$ linearly independent eigenvectors. As the result, the eigenvalues are badly conditioned and cannot be used for stability analysis. For such matrices, the pseudospectrum is a more suitable analytical tool. Numerical experiments reveal that the pseudospectrum for discretizations with nonperiodic boundary conditions converges to the spectrum with periodic boundary conditions. Therefore, the presented analysis is applicable to a more general case of nonperiodic boundary conditions. A more detailed discussion of the spectrum and pseudospec-

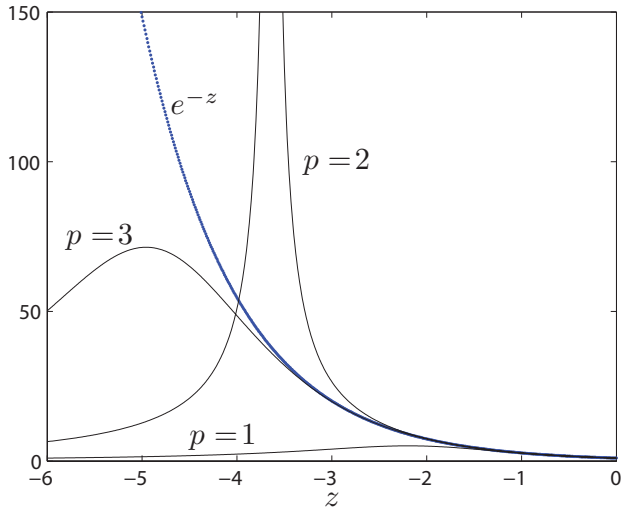

Figure 2.6: Comparison of e^{-z} with $|f_{p+1}(z)|$ for $p = 1, 2, 3$.

trum of the DG discretization can be found in [38].

2.6 Padé approximants and the dispersion relation

The one dimensional linear advection equation (2.2) admits non-trivial solutions of the form

$$u(x,t) = ce^{i(kx-\omega t)}, \tag{2.86}$$

where wave number k and frequency ω satisfy the dispersion relation

$$\omega = ak. \tag{2.87}$$

The numerical scheme usually admits non-trivial solutions similar to (2.86) with k approximated by a numerical wave number $\tilde{k}(\omega)$. This numerical wave number gives information on the ability of the scheme to capture wave propagation. The real part of $\tilde{k}$ is responsible for a phase shift, i.e. the dispersion error, in the numerical solution. The imaginary part of $\tilde{k}$ affects the dissipative or diffusive property of the scheme.

Let us assume that the numerical solution has the form

$$\mathbf{c}_j = e^{-i\omega t} e^{i\tilde{k} j \Delta x} \tilde{\mathbf{c}}, \tag{2.88}$$

where $\mathbf{c}_j$ is the vector of the coefficients. Substituting (2.88) into (2.1) yields

$$\frac{\mathrm{d}}{\mathrm{d}t} e^{-i\omega t} e^{i\tilde{k} j \Delta x} \tilde{\mathbf{c}} = \frac{a}{\Delta x} \left(D_n e^{-i\omega t} e^{i\tilde{k}(j-1)\Delta x} \tilde{\mathbf{c}} + A_n e^{-i\omega t} e^{i\tilde{k} j \Delta x} \tilde{\mathbf{c}} \right), \tag{2.89}$$

or

$$\left(-i\omega \frac{\Delta x}{a} e^{i\tilde{k}\Delta x} I - e^{i\tilde{k}\Delta x} A_n - D_n \right) \tilde{\mathbf{c}} = 0. \tag{2.90}$$

Equation (2.90) has a nontrivial solution when the determinant of the matrix in the bracket is zero. Changing the variables $\lambda = -i\omega \frac{\Delta x}{a}$ and $\xi = e^{i\tilde{k}\Delta x}$, and comparing (2.90) to (2.25), we obtain that $f_n(\lambda) = \xi$, i.e., the numerical dispersion relation is

$$f_n(-i\omega \frac{\Delta x}{a}) = e^{i\tilde{k}\Delta x}. \tag{2.91}$$

When $\omega \frac{\Delta x}{a}$ is small, $f_n(-i\omega \frac{\Delta x}{a}) = e^{i\omega \frac{\Delta x}{a}} + O(\Delta x^{2n})$. Thus, we can estimate

$$\tilde{k} = \frac{-i}{\Delta x} \left[-i\omega \frac{\Delta x}{a} + O(\Delta x^{2n}) \right] = k + O(\Delta x^{2n-1}). \tag{2.92}$$

Noticing that $n = p + 1$, we obtain that the dispersion and dissipation errors are of the order $O(\Delta x^{2p+1})$. We conclude that the growth of the spectrum with the order of approximation p and the dispersion and dissipation errors of the DGM are defined by the way the scheme approximates e^z or e^{-z}.

Chapter 3

Spectrum of the DGM on Nonuniform Grids

3.1 Eigenvalues of L on nonuniform meshes

In this chapter we will analyze the eigenvalues of the spatial discretization matrix L when the mesh is not uniform. We are interested in how small cells in the mesh influence the global stability restriction given by the large cells with the aim to find a time step restriction less severe than the one defined by (1.18). We are particularly interested in cases where small cells comprise a small proportion of the mesh. This can be the case for Cartesian methods with embedded geometries. The cut cells near the embedded boundaries are of irregular size but comprise a small proportion of the mesh. Quite often, they are allowed to be a half of a regular cell size reducing the global time step by a half.

Following the derivation in Section 2.2, we obtain a system of ODEs similar to (2.1)

$$\dot{\mathbf{c}} = \frac{a}{\Delta x} L\mathbf{c}, \tag{3.1}$$

where $\Delta x = \max_{j=1}^{N} \Delta x_j$. With the periodic boundary conditions, L is a block matrix of the form

$$L = \begin{bmatrix} \frac{\Delta x}{\Delta x_1} A_n & 0 & 0 & \dots & 0 & 0 & \frac{\Delta x}{\Delta x_1} D_n \\ \frac{\Delta x}{\Delta x_2} D_n & \frac{\Delta x}{\Delta x_2} A_n & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & \frac{\Delta x}{\Delta x_N} D_n & \frac{\Delta x}{\Delta x_N} A_n \end{bmatrix} = \begin{bmatrix} m_1 A_n & 0 & 0 & \dots & 0 & 0 & m_1 D_n \\ m_2 D_n & m_2 A_n & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & m_N D_n & m_N A_n \end{bmatrix}, \tag{3.2}$$

where $m_j = \frac{\Delta x}{\Delta x_j}$, $j = 1, 2, \dots, N$ and D_n, A_n are $n \times n$, $n = p + 1$ matrices described in (2.7) and (2.13). Following the derivation in Section 2.2, we obtain that the characteristic polynomial of the matrix L is

$$\prod_{j=1}^{N} f_{p+1}(\frac{\lambda}{m_j}) = 1. \tag{3.3}$$

Then, the eigenvalues of L are the roots of the equation (3.3). According to Theorem 3, function $f_{p+1}(z)$ is the $[p/p+1]$ Padé approximant of e^{-z}.

The time step restriction in numerical integration of the system of ODEs (2.1) on a uniform grid is defined by the spectrum of L and the mesh size Δx. If the mesh is refined uniformly by, say, a factor of two, the time step will be reduced by the same factor. However, if only one cell is refined, it is reasonable to assume that the spectrum of L will not change much. In this section, we investigate how a presence of few small cells influences the eigenvalues of L.

Equation (3.3) gives a closed form formula for the eigenvalues of L. However, each $f_{p+1}(\frac{\lambda}{m_j})$ is a rational function. Thus, to solve for λ would require finding the roots of a high order polynomial which is a more complicated task than directly finding eigenvalues of L using a linear algebra method. Nevertheless, (3.3) can give us an insight into the behavior of the spectrum and useful estimates on the eigenvalues.

We start with a simplifying assumption on composition of the mesh and derive a number of results. In particular, we assume that the mesh has cells of two sizes only: Δx and $\Delta x/m$, $m > 1$. We refer to the Δx-sized cells as large cells and to the $\Delta x/m$-sized cells as small or refined cells (smaller cells in practice come from adaptive refinement aiming to resolve either a solution features or the computational domain's geometry). In Section 3.1.1 we establish that there are two distinct cases: m greater and m smaller than some critical value. In the following two sections, we obtain accurate estimates for the spectrum for these two cases. Based on these results, we conjecture a bound on the spectrum when the simplifying assumptions are lifted (Section 3.1.4).

Having obtained the estimates of the spectrum based on various bounds on the roots of (3.3), we discuss when these estimates can be useful. The difficulty with the eigenvalues of L on

nonuniform meshes is that they can be badly conditioned. We try to quantify this in Section 3.1.5. Finally, we summarize the results of Sections 3.1.1-3.1.5 in Section 3.2.

3.1.1 Spectral curves

We start with a uniform mesh, i.e. $m_j = 1,\ \forall j$, in (3.2). Then, (3.3) becomes

$$\prod_{j=1}^{N} f_{p+1}(\lambda) = 1 \tag{3.4}$$

or

$$f_{p+1}(\lambda) = e^{\frac{2\pi i j}{N}},\ j = 0, 1, 2, \ldots, N-1. \tag{3.5}$$

Passing N to infinity, we see that equations (3.4) and (3.5) describe points on the closed curve Γ_p

$$\Gamma_p = \left\{ z \ \middle| |f_{p+1}(z)| = 1\ ,\ z \in C \right\}. \tag{3.6}$$

In the discussion that follows, we will not make a verbal distinction between the discrete eigenvalues (3.5) and continuous equation (3.6) and refer to both as a spectral curve. Examples of the spectral curves for $p = 1, 2, 3, 4$ are shown in Figure 3.1. A simple observation here is that the spectrum size increases with p. This growth is responsible for the $1/(2p+1)$ factor in (1.18).

By (2.26), $f_{p+1}(z)$ is a rational function with the degree of the denominator equal to $p+1$. Hence, it has $p+1$ poles in the complex plane. We list the poles for $p = 1, 2, 3, 4$ in Table 3.1. Denoting the poles by r_i^p, $i = 0, 1, \ldots, p$, we can write $f_{p+1}(z)$ as

$$f_{p+1}(z) = \sum_{i=0}^{p} \frac{w_i^p}{z - r_i^p}, \tag{3.7}$$

where w_i^p are some constants. It follows from (3.7) that $|f_{p+1}(z)|$ decreases as $|z|$ increases when z is sufficiently far away from $\{r_i^p\}$. In particular, $|f_{p+1}(z)| \to 0$ as $|z| \to \infty$. Then, Γ_p is a curve enclosing all poles of $|f_{p+1}(z)|$ with $|f_{p+1}(z)| > 1$ inside the curve and $|f_{p+1}(z)| < 1$ outside the curve. As an example, we plot $f_2(z)$ in Figure 3.2.

Next, we consider a nonuniform mesh with $\Delta x_j = \Delta x / m_j$. Δx is taken to be the size of the largest cell in the mesh. Analogous to the uniform case, we consider a curve

$$\Gamma_p^m = \left\{ z \ \middle|\ \prod_{j=1}^{N} \left| f_{p+1}\left(\frac{z}{m_j}\right) \right| = 1,\ z \in C \right\}, \tag{3.8}$$

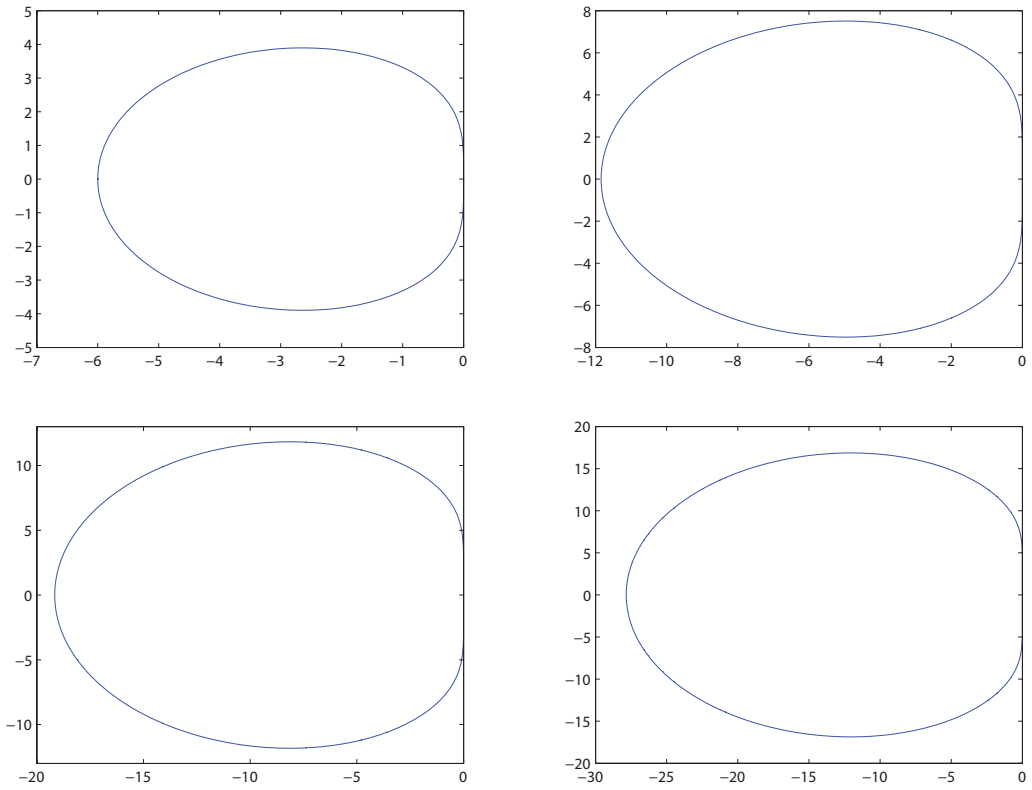

Figure 3.1: Spectrum of the DG discretization plotted as a curve (3.6) for $p = 1, 2, 3, 4$, from left to right, from top to bottom.

and view solutions of (3.3) as points on this curve. Depending on m_j and the location of the poles of $f_{p+1}(z)$, Γ_p^m can be a closed curve or it can consist of several branches.

Let us first consider a simple case where the mesh consists of cells of two sizes only: Δx and $\Delta x/m$. We assume that there are k smaller cells of size $\frac{\Delta x}{m}$ and $N - k$ larger cells of size Δx with a total of N cells. Although the way $\Delta x-$ and $\frac{\Delta x}{m}-$sized cells are distributed in the mesh affects the composition of L (see (3.2)), its eigenvalues are given by (3.3) and, hence, depend only on the relative number of cells of each size. We introduce a parameter $\alpha = k/N$, with $\alpha = 0$ corresponding to a mesh consisting only of cells of size Δx and $\alpha = 1$ corresponding to a mesh consisting only of cells of size $\frac{\Delta x}{m}$. Taking the root N of (3.3), we find that the eigenvalues of L

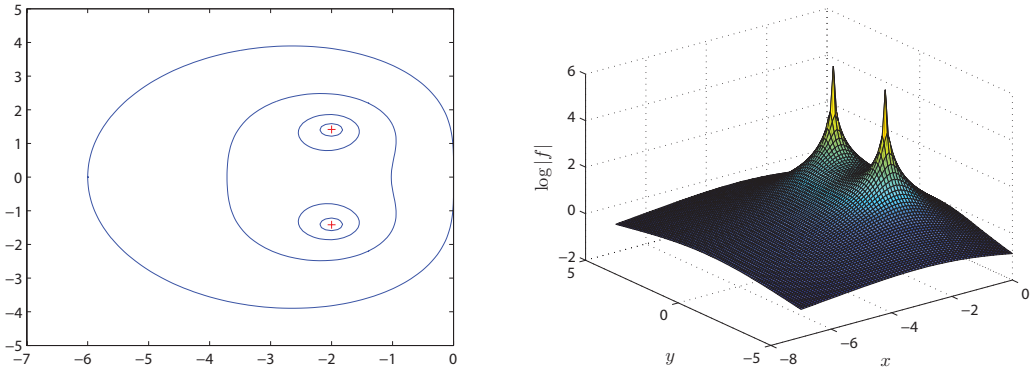

Figure 3.2: Left: isolines of $|f_2(z)| = 1, 10, 10^2, 10^3$ (from the outer to the inner curve); plus signs denote two poles. Right: surface plots of $|f_2(z)|$, $z = x + iy$, log scale.

p	Poles
1	$-2 \pm \sqrt{2}i$
2	-3.637834252744488 $-2.681082873627759 \pm 3.050430199247417i$
3	$-3.212806896871531 \pm 4.773087433276634i$ $-4.787193103128471 \pm 1.567476416895206i$
4	-6.286704751729255 $-3.655694325463563 \pm 6.543736899360069i$ $-5.700953298671815 \pm 3.210265600308537i$

Table 3.1: Poles of $f_{p+1}(z)$ for $p = 1, 2, 3, 4$.

lie on the curve

$$\Gamma_p^{m,\alpha} = \left\{ z \;\middle|\; |f_{p+1}^{1-\alpha}(z) f_{p+1}^{\alpha}(\frac{z}{m})| = 1,\ z \in C \right\},\ 0 < \alpha < 1. \tag{3.9}$$

We investigate how m influences the shape of $\Gamma_p^{m,\alpha}$. We observe that the curve $|f_{p+1}(\frac{z}{m})| = 1$ is a simple scaling (enlargement) of the curve $|f_{p+1}(z)| = 1$ by a factor m and denote it by $m\Gamma_p$. The poles of $f_{p+1}(\frac{z}{m})$ are multiples of the poles of $f_{p+1}(z)$ and are equal to mr_i^p. Inside Γ_p, both $f_{p+1}(z)$ and $f_{p+1}(\frac{z}{m})$ are greater than one in magnitude (Figure 3.3). Similarly, outside $m\Gamma_p$, both functions are less than one in magnitude. Then, the solution of (3.9) should lie in the region between two curves where we have $|f_{p+1}(\frac{z}{m})| > 1$ and $|f_{p+1}(z)| < 1$ (Figure 3.3). If all poles

p	1	2	3	4
M	2.26	2.40	2.48	2.53

Table 3.2: Values of M^{cr} for $p = 1, 2, 3, 4$.

of $m\Gamma_p$, mr_i^p, are located inside Γ_p, then both functions are smooth in the region between two curves. Then, there exists a solution to (3.9) that geometrically forms a closed curve. We cannot make the same statement when $f_{p+1}(\frac{z}{m})$ has poles in this region, i.e. some mr_i^p lie outside Γ_p. In this case, depending on the values of m and α, the curve $\Gamma_p^{m,\alpha}$ might consist of one or several branches. Thus, the refinement factor m determines the type of spectral curves on nonuniform meshes. In Table 3.2, we list the approximate critical values of m, M^{cr}, for $p = 1, 2, 3, 4$. The critical values were obtained numerically by finding m such that mr_i^p lies on Γ_p for one of the poles. When m is smaller than M^{cr}, $\Gamma_p^{m,\alpha}$ is a closed curve; it may have a more complex form otherwise.

3.1.2 Mesh refinement by a factor m less than M^{cr}

We continue with the simple case where the mesh contains $N - k$ cells of size Δx and k cells of size $\frac{\Delta x}{m}$. The refinement factor m is assumed to be greater than one but less than the critical value M^{cr} reported in Table 3.2. We estimate growth of the spectrum as a function of α, i.e. the ratio of small cells to the total number of cells, for a fixed m. In Figure 3.4, left, we plot the spectrum of L on a hundred cell mesh with 0, 20, 40, 60, 80, and 100 small cells and $p = 1$. We also plot the largest eigenvalue by magnitude versus the number of small cells in Figure 3.4, right. We observe that the spectrum enlarges nearly linearly as a function of α.

Noting that -6 is the largest in magnitude root with $\alpha = 0$ and $-6m$ is the largest root with $\alpha = 1$, we substitute

$$z = -6[1 + (m - 1)\alpha] \tag{3.10}$$

into (3.9) to obtain function $g(m, \alpha)$

$$g(m, \alpha) = \left| f_2^{1-\alpha}(-6[1 + (m - 1)\alpha]) f_2^{\alpha}\left(\frac{-6[1 + (m - 1)\alpha]}{m}\right) \right|. \tag{3.11}$$

We plot $g(m, \alpha)$ in Figure 3.5 with $m \in [1, 2.26]$ and observe that $g(m, \alpha) = 1$ when $\alpha = 0, 1$, and $g(m, \alpha)$ is strictly less than one when $\alpha \in (0, 1)$, $m \neq 1$. This means that for z satisfying (3.10), $|f_2^{1-\alpha}(z) f_2^{\alpha}(\frac{z}{m})| < 1$, i.e. these z must be located to the left of the curve $\Gamma_1^{m,\alpha}$. Thus, (3.10) is a bound on the real root which we take to be a proxy for the size of the spectrum. We also note

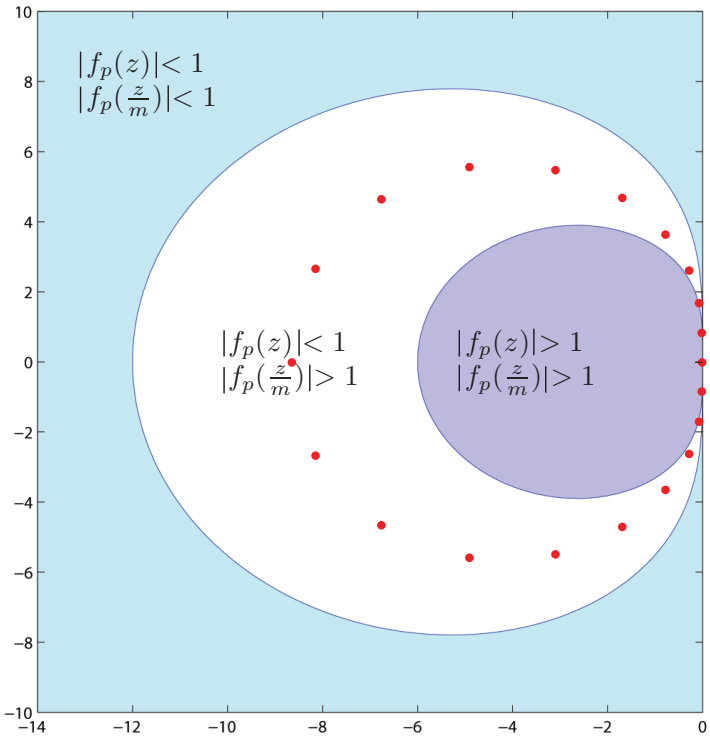

Figure 3.3: Spectral curves Γ_p (inner) and $m\Gamma_p$ (outer). Dots denote solution of (3.9). $p = 1$, $m = 2$.

that the curve plotting the largest eigenvalue and the linear bound are close (Figure 3.4, right), so the bound is tight. Additionally, we plot the largest eigenvalues by magnitude for $p = 2$ and $p = 3$ with $m = 2$ in Figure 3.6 and observe that the growth rate of the spectrum is also slightly lower than linear. The analysis with these orders of approximation is similar and is omitted here.

We conclude that if a mesh consists of only two types of cells and the refinement factor m is less than the critical value given in Table 3.2, increasing the proportion of small cell α leads to enlargement of the spectrum by a factor approximately equal to $1 + (m - 1)\alpha$.

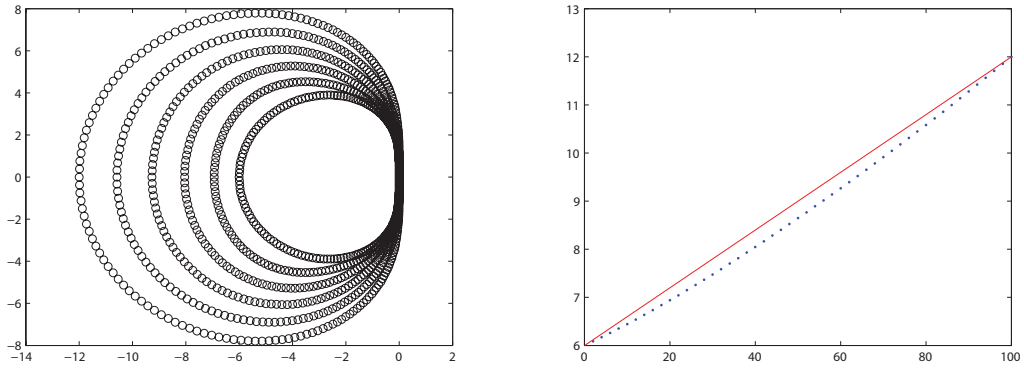

Figure 3.4: Left: Eigenvalues of L for meshes consisting of 100 cells with 0, 20, 40, 60, 80, and 100 half-size cells and $p = 1$. The inner curve corresponds to the mesh with no half-size cells. Right: Largest eigenvalues by magnitude (vertical axis) versus the number of half-size cells, k, in a $N = 100$ cell mesh (horizontal axis); dots denote the eigenvalues and the solid line denotes $z = 6[1 + k/N]$.

3.1.3 Mesh refinement by a factor m greater than M^{cr}

Next, we consider the case when m is greater than the critical factor M^{cr}. By discussion in Section 3.1.1, $\Gamma_p^{m,\alpha}$ might consist of several curves with eigenvalues of L located on those distinct curves. Let us first consider the case when all cells have the same size Δx except for one small cell of size $\frac{\Delta x}{m}$. We will look for an approximation of $(p+1)N$ roots of (3.3). Letting $\tilde{f}_{p+1}(z) = f_{p+1}(\frac{z}{m})$, we can write (3.3) as

$$f_{p+1}(z) = \sqrt[N-1]{\frac{1}{\tilde{f}_{p+1}(z)}}. \tag{3.12}$$

Away from the poles mr_i^p, we have $1 < |\tilde{f}_{p+1}(z)| \leqslant c < \infty$. Then, we obtain

$$\left|f_{p+1}(z)\right| = \left| \sqrt[N-1]{\frac{1}{\tilde{f}_{p+1}(z)}} \right| \to 1, \text{ as } N \to \infty.$$

So,

$$f_{p+1}(z) \approx \omega, \text{ where } \omega = \sqrt[N-1]{1}. \tag{3.13}$$

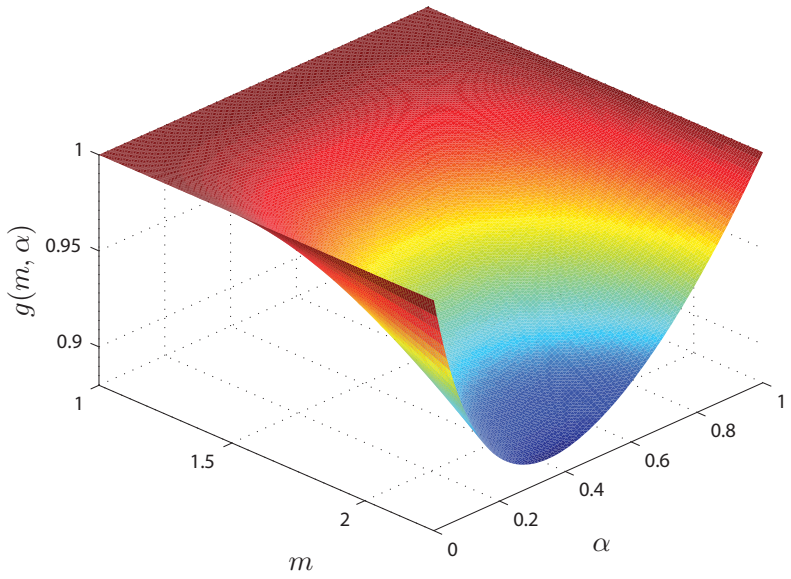

Figure 3.5: Plot of function $g(m, \alpha)$, $p = 1$.

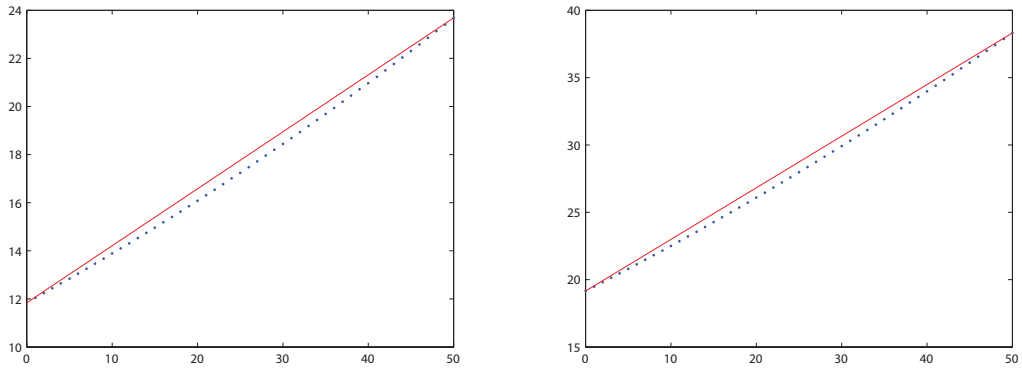

Figure 3.6: Largest eigenvalues by magnitude (vertical axis) versus the number of half-size cells, k, in a 50-cell mesh with $p = 2$ (left) and $p = 3$ (right). Dots denote the eigenvalues and solid line denotes $z = z^\star[1 + k/N]$. $z^\star = 11.84$ with $p = 2$ and $z^\star \approx 19.16$ with $p = 3$ [38].

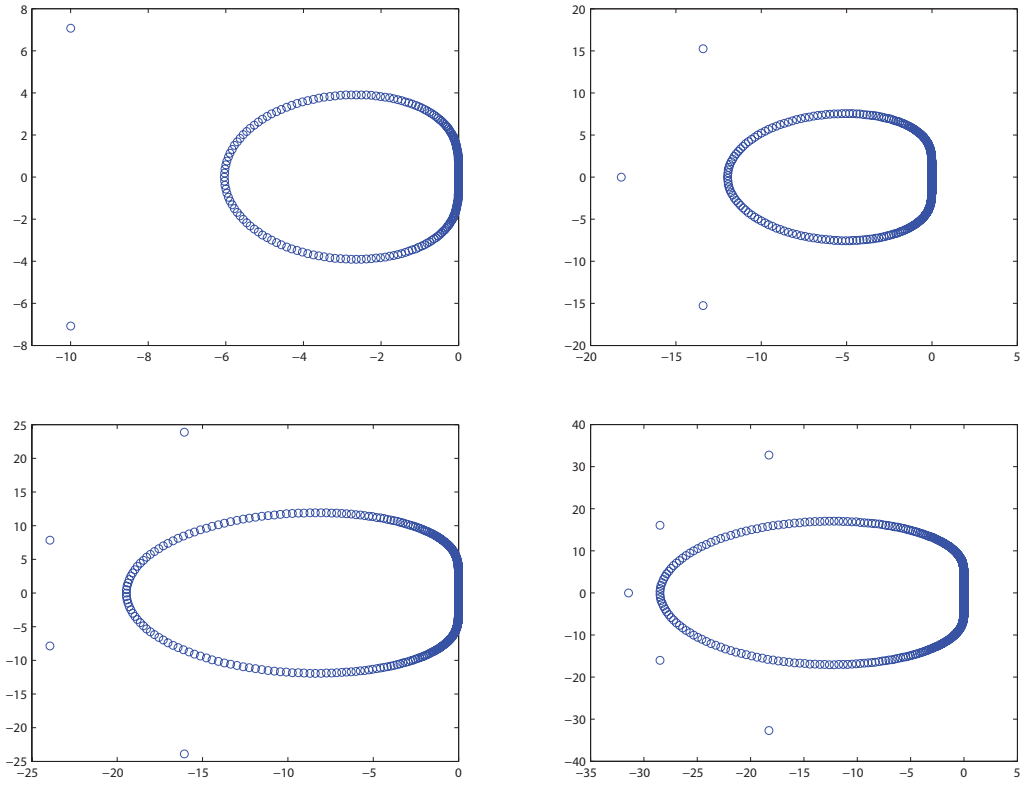

Figure 3.7: Spectrum of L on a 100 cell mesh with 99 cells of size Δx and one cell of size $\frac{\Delta x}{5}$. $p = 1, 2, 3, 4$, left to right, top to bottom.

We can solve (3.13) to obtain $(p+1)(N-1)$ distinct roots. Equation (3.13) is similar to the previously considered case of a uniform mesh (3.4)-(3.5) which leads us to conclude that its roots will be located on a curve similar and close to Γ_p.

Next, we seek to estimate the remaining $(p+1)$ roots of (3.3). We rewrite (3.3) as

$$\tilde{f}_{p+1}(z) = \frac{1}{f_{p+1}^{N-1}(z)}. \tag{3.14}$$

As discussed in Section 3.1.1, the solution of (3.3) is located outside Γ_p (Figure 3.3). Hence, $|f_{p+1}(z)| < 1$ in (3.14). Then, $f_{p+1}^{N-1}(z) \to 0$ as $N \to \infty$, i.e. z must be close to the poles of $\tilde{f}_{p+1}(z)$ in order to satisfy (3.14). Thus, we have an estimate for those $p+1$ roots with relatively large

modulus as

$$z = mr_i^p. \tag{3.15}$$

Similarly, when the mesh has a few small cells of different sizes $\frac{\Delta x}{m_j}$, L has eigenvalues that are close to the corresponding multiples of the poles, i.e. $m_j r_i^p$.

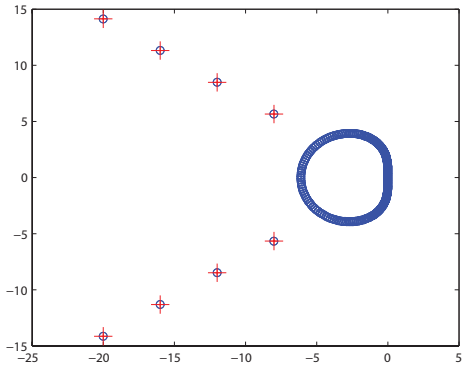

Figure 3.8: Spectrum of L on a 100 cell mesh with 96 cells of size Δx and four cells of sizes $\Delta x/4$, $\Delta x/6$, $\Delta x/8$, and $\Delta x/10$ with $p = 1$. Circles denote exact eigenvalues, plus sizes denote approximations.

We illustrate our findings in Figures 3.7 and 3.8. In the results presented in Figure 3.7, the meshes consist of one hundred cells with 99 cells of size Δx and one cell of size $\Delta x/5$ with $p = 1, 2, 3, 4$. We observe that for all p all eigenvalues lie on a curve similar to one on uniform meshes (Figure 3.1) except for $(p + 1)$ outlying eigenvalues. We compute the eigenvalues using MATLAB and their estimates according to (3.13) and (3.15) . Then, we compare the results to find that they differ by $10e - 14$. This indicates that even though (3.13) and (3.15) are only asymptotically correct when $N \to \infty$, the estimates on a modestly sized one hundred cell mesh are extremely accurate. This is not very surprising given that the power function grows and decays exponentially fast. Figure 3.8 shows eigenvalues on a one hundred cell mesh with first four cells of size $\Delta x/4$, $\Delta x/6$, $\Delta x/8$, and $\Delta x/10$, with $p = 1$. The exact eigenvalues (circles) and the approximation given by (3.15) (plus signs) are again extremely close. The outlying eigenvalues are 4, 6, 8, and 10 multiples of $(-2 \pm \sqrt{2}i)$.

Next, we allow the mesh to have $N - k$ cells of size Δx and k cells of $\frac{\Delta x}{m}$. When k is small, the outlying roots cluster in the neighborhood of points given by (3.15) with the radius of the neighborhood depending on the number of small cells (Figure 3.9). As k increases, the size of

the cluster enlarges until it mergers with the larger curve (Figure 3.10). We do not attempt to find an estimate for these eigenvalues for two reasons. First, we treat it as a part of a more general case considered in Section 3.1.4. Second, pseudospectra of L on such meshes is far from the spectrum making the exact estimates not very useful. The second point is further discussed in Section 3.1.5.

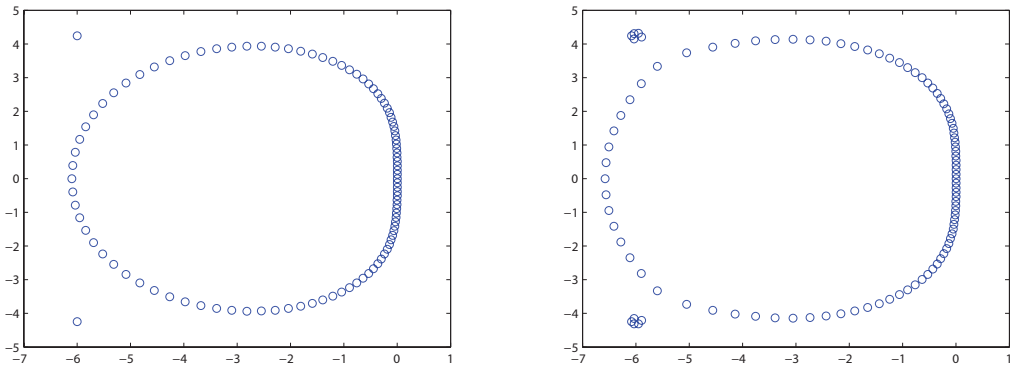

Figure 3.9: Left: The first cell in the $N = 50$ mesh has size $\frac{\Delta x}{3}$ and the rest are of size Δx. Right: The first 5 cells in the $N = 50$ mesh have size $\frac{\Delta x}{3}$ and the rest are of size Δx. $p = 1$.

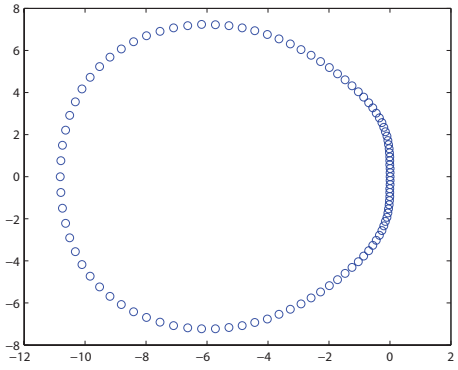

Figure 3.10: A half of the cells in the $N = 50$ mesh have size $\frac{\Delta x}{3}$ and the other are of size Δx.

3.1.4 Mesh with cells of arbitrary sizes

We turn to a general case where we make no assumptions on the composition of a mesh. For a given N-cell mesh, the size of the largest cell is called Δx and $m_j = \Delta x/\Delta x_j$, $j = 1, 2, \ldots, N$. Finding the exact solution of (3.3) for an arbitrary set of $\{m_j\}$ does not seem feasible. Instead, we seek to use the ideas from Sections 3.1.2 and 3.1.3 to conjecture a bound on the spectrum.

Given that the spectrum grows almost linearly as a function of the proportion of small cells for low values of m when only two types of cell sizes are present (Section 3.1.2), we conjecture a linear in m_j bound on the size of the spectrum in the general case as well. That is, we estimate the growth rate by the average of factors m_j

$$\bar{m} = \frac{1}{N}\sum_{j=1}^{N} m_j \tag{3.16}$$

if $m_j < M^{cr}$, $\forall\, j$. Thus, the estimate for the spectrum is $\bar{m}\Gamma_p$. We present the spectrum of L where cell sizes are random values between 1 and M^{cr}, $p = 3$ and $N = 50$ in Figure 3.11. The bound appears to be accurate.

Next, we incorporate into the estimate where $m_j > M^{cr}$ for some j. From Section 3.1.3 and equation (3.15), the largest eigenvalues are proportional to the poles (Figures 3.7 and 3.8). Let $z_{r,i}$, $i = 1, 2, \ldots, p+1$, be a point of Γ_p lying on the line connecting the pole r_i^p and the origin and let $\beta_i = |z_{r,i}|/|r_i^p|$. Then we have the following estimate on the largest eigenvalue on this line

$$\frac{1}{N}\sum_{j=1}^{N} \tilde{m}_j r_i^p, \tag{3.17}$$

where $\tilde{m}_j = \max\left(m_j\beta_i, m\right)$ and $m = \max_j m_j$.

The complete bound on the spectrum of L is taken to be the combination of (3.16) and (3.17). In Figure 3.12 we give a couple of examples. In the first example, we have the first ten cells of a hundred cell mesh being of size $\frac{\Delta x}{3}$ and the rest of size Δx. In the second example, the mesh consists of fifty cells of random sizes. The plus signs denote an estimate on eigenvalues corresponding to the cells close in size to the largest cell. They were calculated using (3.16). The circles denote an estimate on the outlying eigenvalues corresponding to the smaller cells. These were computed using (3.17). Note the alignment of the larger eigenvalues along straight lines which are the lines connecting the origin and r_1^1 and the origin and r_2^1. Both the estimate for the eigenvalues corresponding to large cells and the estimate for the outlying eigenvalues are quite accurate.

The estimates given in this section are less tight than the ones presented in Sections 3.1.2 and 3.1.3. However, they are much closer to the true spectrum than to the spectrum scaled by the size of the smallest cell. We note that for the example in Figure 3.11 the spectrum scaled by the size of the smallest cell $\Delta x/2.48$, would have the largest eigenvalue by magnitude approximately equal to $19.16 * 2.48 = 47.71$, where 19.16 is the largest eigenvalue by magnitude of the $p = 3$ discretization (Figure 3.1). Similarly, for the example in Figure 3.12, left, the value is $6 * 3 = 18$. Thus, even though the estimates in this section are more relaxed, they are still quite accurate. Finally, we note that the conjecture has been extensively tested on multiple cases in which it consistently provides an upper bound on the exact eigenvalues.

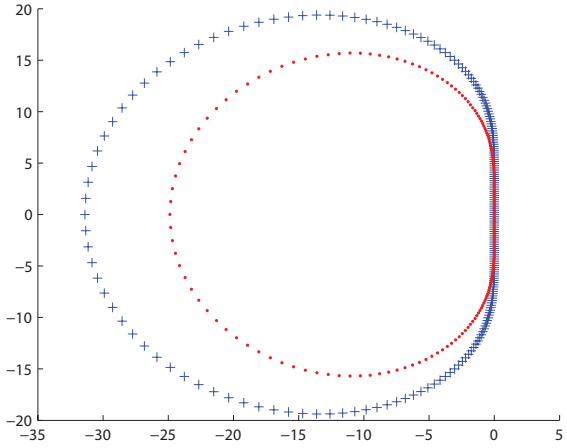

Figure 3.11: Eigenvalues and a bound given by (3.16) with $p = 3$, m_j are random numbers between 1 and 2.48. Dots denote the eigenvalues, plus signs denote the estimate.

3.1.5 Pseudospectra

In numerical simulations, components of L are evaluated at every time integration step. This process is seldom exact, especially for nonlinear problems. In (1.13), we commit errors while evaluating U_j and P_k, using numerical quadratures, etc. Consequently, the entries of L used in computations are perturbed values of the exact L. If L is such that its eigenvalues are very sensitive to perturbations, i.e. small changes to its components result in large changes in eigenvalues, the spectrum of the computed L might be far from the spectrum of the exact L.

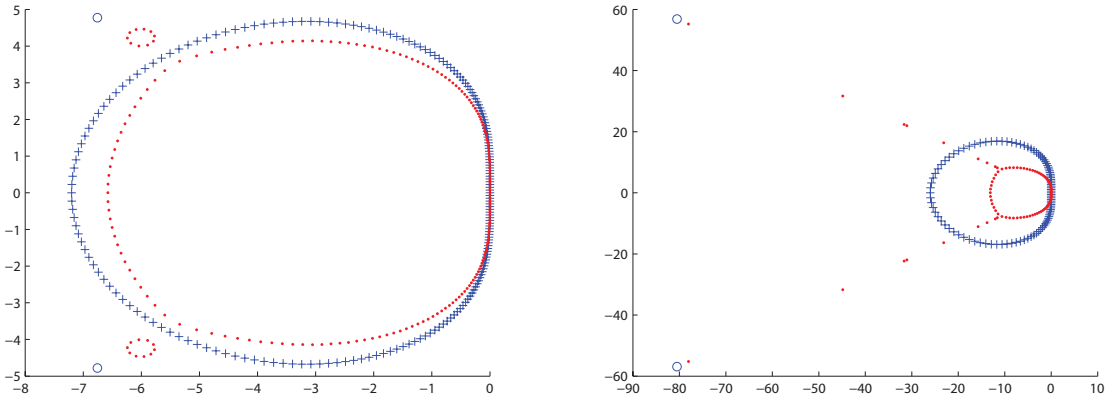

Figure 3.12: Eigenvalues and a computed bound with $p = 1$. Dots denote the eigenvalues, plus signs denote the estimate given by (3.16), and circles denote the estimate in the directions of the poles given by (3.17). Left: The first 10 cells out of 100 cells have size $\frac{\Delta x}{3}$. Right: A mesh consists of 50 cells of random sizes Δx_j.

We define ϵ-pseudospectrum $\sigma_\epsilon(L)$ of matrix L as the set of $z \in \mathbb{C}$ such that

$$z \in \sigma(L + E) \tag{3.18}$$

for some $E \in \mathbb{C}^{N(p+1)\times N(p+1)}$ with $\|E\| < \epsilon$, where $\sigma(L)$ is the spectrum of L. Computation of the pseudospectra in this section follows the algorithm from Section 39 in [52]. We use an equivalent definition of $\sigma_\epsilon(L)$

$$\sigma_\epsilon(L) = \{z \in \mathbb{C} \mid s_{min}(zI - L) < \epsilon\}, \tag{3.19}$$

where $s_{min}(\cdot)$ denotes the minimum singular value. We compute $s_{min}(zI - L)$ on a rectangular grid in the domain of interest in the complex plane using MATLAB built-in functions. A contour plot is used to show the boundaries of pseudospectra.

Computing pseudospectra using (3.19) is time intensive and cannot be done in run time, especially for large-scale problems. Here, we examine spectra and pseudospectra of L on a number of sample meshes with the aim to determine when they are close and when they are far apart. First, we consider fifty-cell meshes where the first cell or the first five cells are of size $\frac{\Delta x}{m}$ and the rest are of size Δx with $p = 1$. We plot spectra and pseudospectra with $m = 2$ (top) and $m = 3$ (bottom) in Figure 3.13. When only one small cell is present in the mesh, the pseudospectra are close to the spectrum (Figure 3.13, left). However, when five small cells

are located next to each other, the pseudospectra grows fast with ϵ (Figure 3.13, right). To test

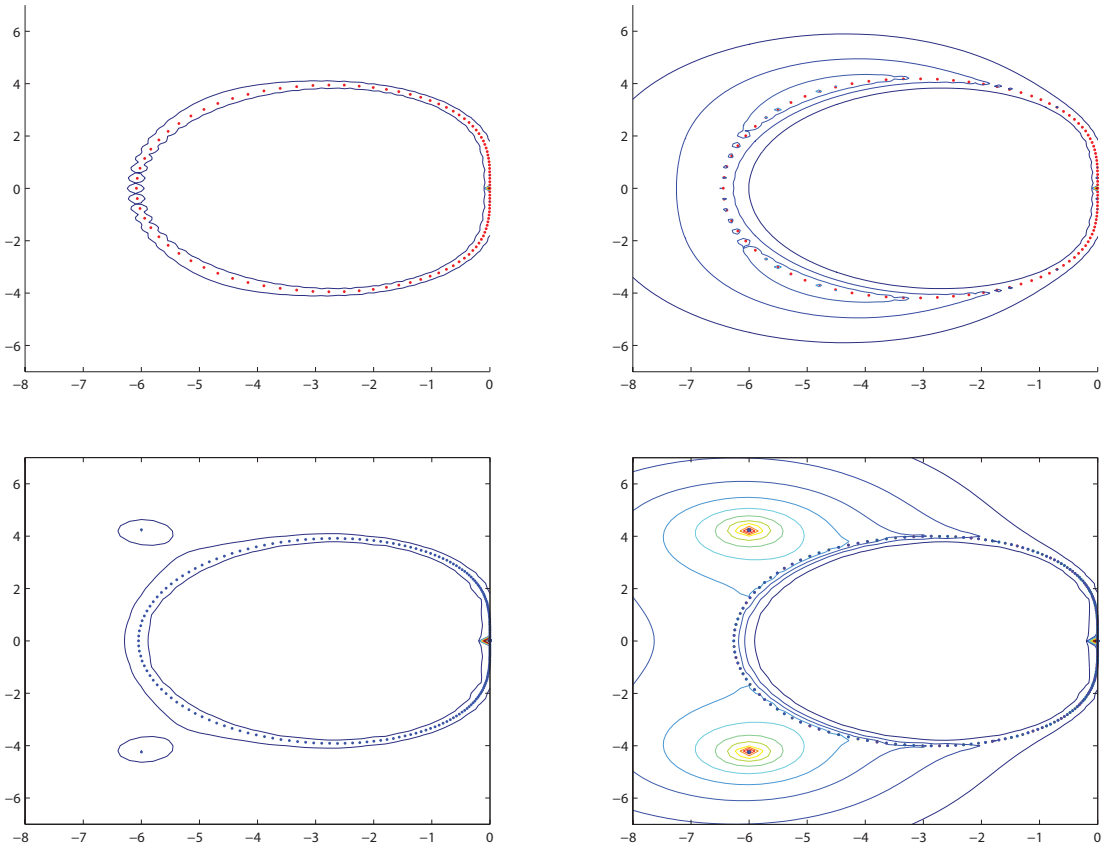

Figure 3.13: Lines denote pseudospectra corresponding to $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-12}$, $p = 1$, $N = 50$. Dots denote the eigenvalues. Meshes consists of cells of sizes Δx and $\Delta x/m$ with one $\Delta x/m$-sized cell (left) and five $\Delta x/m$-sized cells. $m = 2$ (top row) and $m = 3$ (bottom row).

whether the number of small cells or their location influences the size of pseudospectrum, we consider two meshes consisting of twenty cells each with ten cells of size Δx and ten cells of size $\Delta x/m$. However, the ordering of cells in these meshes is different. Mesh one has small and large cells interlaced, i.e. large, small, large, small, etc. Mesh two has its cells arranged into two blocks: ten cells of size Δx followed by ten cells of size $\Delta x/m$. The spectrum and pseudospectra of these meshes are shown in Figure 3.14 with $m = 2$ (top) and $m = 5$ (bottom). The spectrum of both meshes is the same by equation (3.3). However, the pseudospectra are quite different

with the pseudospectrum of the first mesh being close to the spectrum and the pseudospectra of the second growing fast with ϵ. Finally, we test whether only the number of small cells or

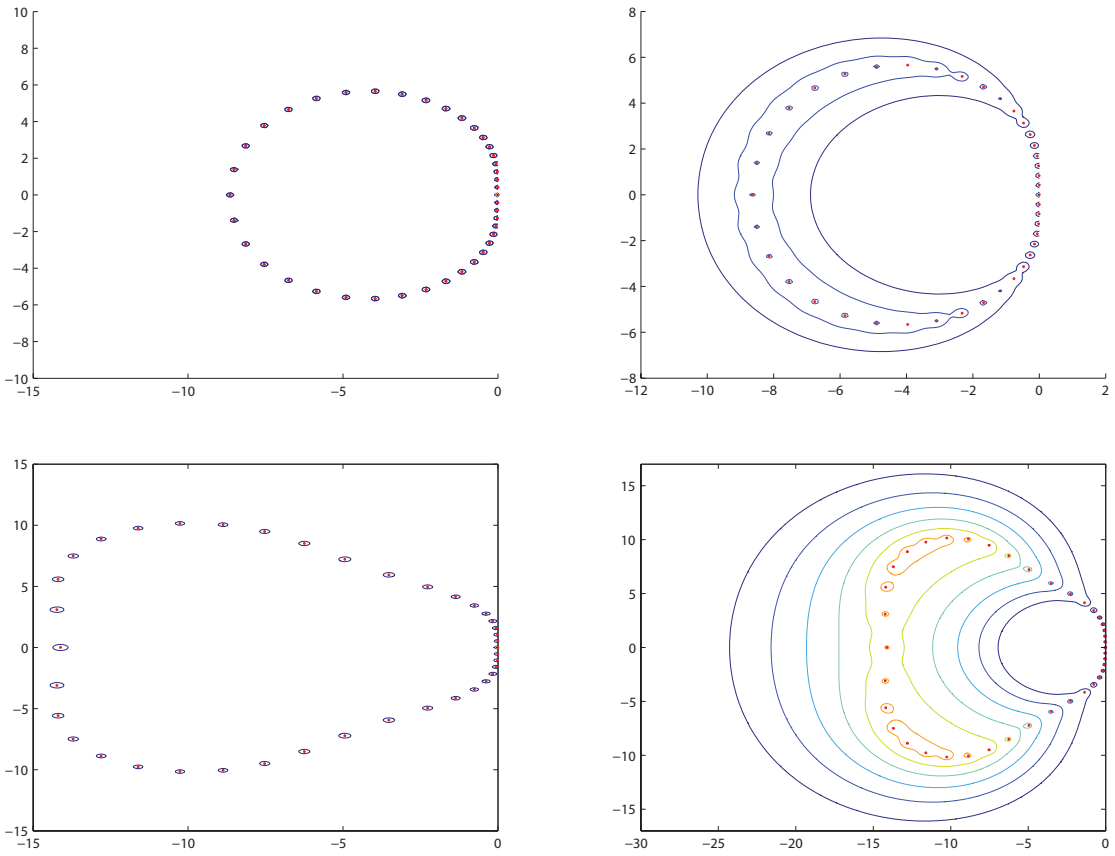

Figure 3.14: Spectrum (dots) and pseudospectra (lines) corresponding to $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-8}$. Meshes consists ten Δx and ten $\Delta x/m$-sized cells. $m = 2$ (top row) and $m = 5$ (bottom row), interlaced cell sizes (left column) and cells arranged in blocks according to size (right column).

the proportion of the number of small cells to the total number of cells influence pseudospectra. Again, we plot the spectra and pseudospectra with $p = 1$ and $m = 2$ and $m = 5$. All meshes have 20 small cells; the results in the left column of Figure 3.15 were computed on sixty cell meshes and the results in the right column on two hundred cell meshes. The pseudospectra seem similar, i.e. not influenced by the total number of cells.

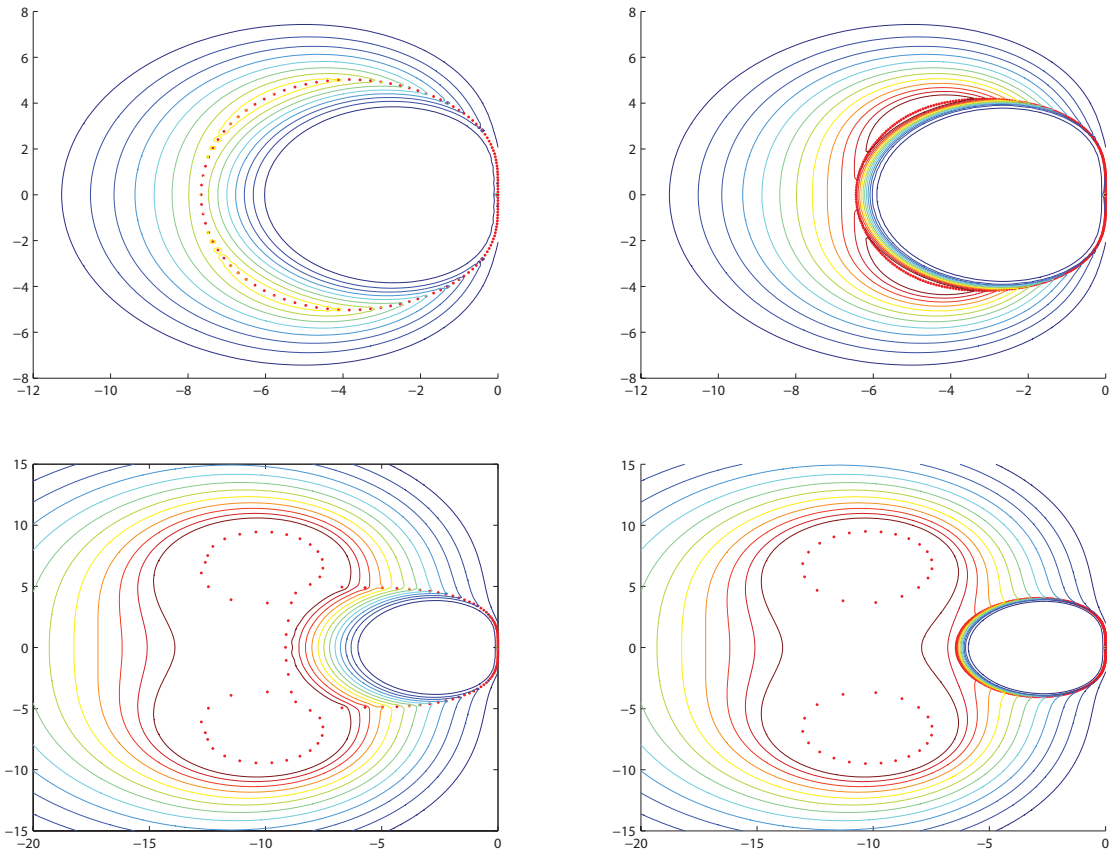

Figure 3.15: Spectrum (dots) and pseudospectra (lines) corresponding to $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-12}$. Meshes consists of cells of sizes Δx and $\Delta x/m$ with twenty $\Delta x/m$ cells. $m = 2$ (top row) and $m = 5$ (bottom row), $N = 60$ (left column) and $N = 200$ (right column).

The conclusion from our experiments is that when small cells are isolated from each other, the pseudospectra are close to the spectrum. However, when small cells are located adjacent to each other in a block of cells the pseudospectra are far from the spectrum. An intuitive explanation for this phenomenon is that the second mesh can be viewed as a composition of two meshes of size Δx and $\Delta x/m$. Then, the pseudospectra on mesh two approaches the spectrum of $\Delta x/m$ sized mesh as ϵ and/or the total number of cells in it increases. A formal argument can be based on Theorem 2.4 for block matrices in [52], p. 20, as L can be split into two blocks corresponding to

Table 3.3: Condition number of the largest eigenvalue of L on $N = 200$ meshes. k is the number of small cells.

	$p = 1$		$p = 2$		$p = 3$		$p = 4$	
k	$m = 2$	$m = 5$	$m = 2$	$m = 5$	$m = 2$	$m = 5$	$m = 2$	$m = 5$
1	2.41	2.65	1.27	3.63	1.33	2.52	1.32	2.01
2	12.10	1.44E8	2.54	16.54	2.98	34.10	1.77	7.77
3	73.84	4.72E10	11.98	4.49E8	3.34	1.17E2	3.03	3.97E2
4	4.69E2	1.31E12	19.45	1.55E9	15.57	1.79E9	4.69	3.36E3
5	2.96E3	6.80E12	74.73	1.27E11	17.95	3.51E9	17.26	9.73E9

cells of each size.

The growth of the size of pseudospectra is the result of the nonnormality of L. L is always diagonalizable with $L = V\Lambda V^{-1}$, but V might be nonunitary. Then, $\|V\|\cdot\|V^{-1}\|$ might be large and the powers of matrix L will depend not only on its eigenvalues but also on the basis of eigenvectors V. The condition number of the matrix V provides an upper bound for the pseudospectrum of L. This is described in the Bauer-Fike theorem ([52], p.20). The sensitivity of each eigenvalue to perturbations is related to its condition number defined by the reciprocals of the cosines of the angles between its left and right eigenvectors. In Table 3.3 we give the condition number of the largest eigenvalues by magnitude for various numbers of small cells in a block in a two hundred cell mesh. We note that when $m > M^{cr}$, the condition number is large when the number of small cells is greater than one ($p = 1$) or two ($p > 1$). When refinement factors are smaller than critical values, more cells can be entered in a row without affecting the condition number much. We also note, that condition numbers grow slower for higher orders of approximation p.

To conclude, when cell sizes are intermingled in a mesh, the eigenvalues are well conditioned and the spectrum can be used to calculate a less restrictive condition on a time step. However, when cells are grouped according to size, the spectrum might not be a reliable instrument. Table 3.3 gives an idea how many cells can be grouped in a block before the pseudospectra starts to grow. We finally note that the pseudospectra converge to the curve $m\Gamma_p$, where m is the largest refinement factor, i.e. to the spectral curve defined by the smallest cells (Figure 3.1) and Figures 3.13 - 3.15).

In the next section, we demonstrate that we still can use the spectrum-defined time step for solution of linear problems but such calculations become unstable for nonlinear equations. A curious case is a mesh consisting of randomly sized elements (Figure 3.12, right), i.e. randomly scaled A_p and D_p in L. Random matrices have well-conditioned eigenvalues [52] and so is L in

this case.

3.2 Numerical tests

In this section we solve a number of problems using the DGM with the time step determined according to the analysis in Sections 3.1.2-3.1.4. We perform long time computations so that if slight instabilities were present, they would become apparent.

Before moving on to numerical experiments, we summarize the findings of previous sections.

Summary of Results

- The spectrum can be used to find a stable time step that is larger than the one prescribed by the classical theory.
- The size ratio of the largest to the smallest cell influences the shape of the spectrum. Two cases need to be considered: the ratio greater and the ratio smaller than the order-dependant critical value reported in Table 3.2.
  - The ratio smaller than the critical one results in a continuous enlargement of the spectrum.
  - The ratio larger than the critical one results in appearance of large eigenvalues located away from the main spectral curve.
- When the number of small cell located next to each other is small (Table 3.3), the spectrum can be reliably used to find a stable time step that is much larger than the one given by the classical CFL restriction.
- When the number of small cells located next to each other is large (Table 3.3), the pseudospectrum converges to the spectrum defined by the smallest cells.

3.2.1 One dimensional linear advection equation

We solve the linear advection equation (2.2) on interval $[-1, 1]$ with periodic boundary conditions using linear approximation in space. The initial condition is $u(x, 0) = \sin(\pi x)$. The mesh has 100 cells of size Δx and one small cell of size $\frac{\Delta x}{5}$, i.e. $m = 5$ and $m > M^{cr}$. The small cell is located near the left endpoint. According to the analysis in Section 3.1.3, matrix L has two large outlying eigenvalues approximately equal to $5(-2 \pm \sqrt{2}i)$. For the fully discrete scheme to be stable Δt

should be small enough so that the eigenvalues of $\Delta t/\Delta x L$ all lie in the absolute stability region of the time integration scheme. We use the classical second order Runge-Kutta (RK2) scheme. Its stability region is given by [27]

$$\left|1 + z + \frac{1}{2}z^2\right| \leqslant 1. \tag{3.20}$$

Substituting $z = \Delta t m(-2 \pm \sqrt{2}i)$, $m > M^{cr}$ into (3.20) and solving for $\max \Delta t$, we find that the time step in this case should be $\Delta t \leqslant \frac{2.685}{m}\frac{\Delta x}{3}$ instead of $\frac{1}{m}\frac{\Delta x}{3}$ given by (1.18), i.e. it can be safely increased by a factor of 2.685. We plot the result of the computation with $\Delta t = \frac{2.685}{5}\frac{\Delta x}{3} = 0.537\frac{\Delta x}{3}$ in Figure 3.16, left. In Figure 3.16, right, we present the computation performed with a slightly larger time step $\Delta t = 0.538\frac{\Delta x}{3}$. This computation was clearly unstable. We conclude that our estimate of the largest eigenvalue is accurate and tight. Similarly, we can compute the time step restriction with $p = 2$ and $p = 3$ paired with RK 3 and RK4. These are given in Table 3.4.

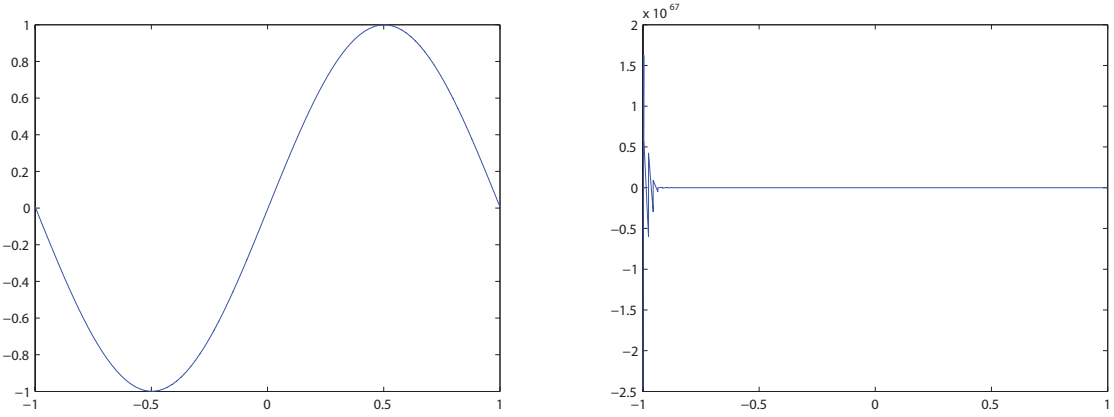

Figure 3.16: Numerical solutions for Example 3.2.1 at time $T = 100$ with $p = 1$, $N = 101$. The left one is using $\Delta t = 0.537\frac{\Delta x}{3}$, the right one is using $\Delta t = 0.538\frac{\Delta x}{3}$.

Next, we solve the problem on a number of meshes with the refinement factor smaller than the critical value and present the results in Table 3.5, where N is the total number of elements and k is the number of small cells. We took $m = 2$. We note that Δx in Table 3.5 is not the same on the three meshes that we used because it depends on the number of small cells and their size. The numbers in the "CFL-exact" column were determined by the exact eigenvalues and were tested in numerical experiments by integrating to the final time $T = 100$. The data in the "CFL-estimate" column refers to the time restriction obtained by using the approximate upper

Table 3.4: CFL number on meshes having a few small cells, $m > M^{cr}$.

p	Time integration scheme	Time step estimation	Classical time step	Improvement
1	RK2	$\frac{0.895\Delta x}{m}$	$\frac{\Delta x}{3m}$	2.685
2	RK3	$\frac{0.594\Delta x}{m}$	$\frac{\Delta x}{5m}$	2.970
3	RK4	$\frac{0.455\Delta x}{m}$	$\frac{\Delta x}{7m}$	3.185

Table 3.5: Time restriction with varying mesh composition.

N	k	Size of small cells	CFL - exact	CFl-estimate
100	1	$\Delta x/2$	$\frac{\Delta x}{3} \times 0.993$	$\frac{\Delta x}{3} \times 0.990$
100	20	$\Delta x/2$	$\frac{\Delta x}{3} \times 0.865$	$\frac{\Delta x}{3} \times 0.833$
100	50	$\Delta x/2$	$\frac{\Delta x}{3} \times 0.694$	$\frac{\Delta x}{3} \times 0.667$

bound discussed in Section 3.1.4. Again the approximated values are slightly below the exact values and provide a safe and tight estimate. We note that for this linear problem on a relatively small mesh, the instabilities due to badly conditioned eigenvalues are not present and we are able to carry out computations with a time step defined by the exact eigenvalues.

Finally, we consider a mesh with several levels of refinement. The mesh is composed as follows: $3 \times \frac{\Delta x}{16}$, $500 \times \frac{\Delta x}{8}$, $500 \times \frac{\Delta x}{4}$, $500 \times \frac{\Delta x}{2}$, $1000 \times \Delta x$, $500 \times \frac{\Delta x}{2}$, $500 \times \frac{\Delta x}{4}$, $500 \times \frac{\Delta x}{8}$, $2 \times \frac{\Delta x}{16}$, where $500 \times \frac{\Delta x}{8}$ denotes five hundred cells of size $\frac{\Delta x}{8}$. Thus, the mesh consists of 4005 elements. This simulates a Cartesian mesh with four levels of refinement and a few cut cells half the size of the smallest regular cell located near the boundaries. The pseudospectrum of L in this case approaches the spectral curve Γ_1 scaled by $m = 8$, i.e. the spectrum defined by the smallest regular cell. We solve the problem with $p = 1$ and $\Delta t = \frac{1}{8}\frac{\Delta x}{3}$ and find the computations stable. We repeat computations with $p = 2$ and $\Delta t = \frac{1}{8}\frac{\Delta x}{5}$ and find them stable as well. We conclude that a few cut cells with the half size of the smallest regular cell do not influence stability restriction on a large mesh. The allowable number of such small cells in a block near the boundary is given in Table 3.3. We also consider a mesh with cut cells that are smaller, i.e. the composition of the mesh is $\frac{\Delta x}{40}$, $500 \times \frac{\Delta x}{8}$, $500 \times \frac{\Delta x}{4}$, $500 \times \frac{\Delta x}{2}$, $1000 \times \Delta x$, $500 \times \frac{\Delta x}{2}$, $500 \times \frac{\Delta x}{4}$, $500 \times \frac{\Delta x}{8}$, $\frac{\Delta x}{40}$. The largest eigenvalues in the spectrum correspond to the smallest cells. First, we notice that $m = 40$ for this mixed-size element mesh and choose the time step using Table 3.4. We solve the problem with $p = 2$ and observe a nearly three-fold improvement over the global time step $\Delta t = \frac{1}{40}\frac{\Delta x}{5}$.

3.2.2 Burgers' equation

We solve the Burgers' equation

$$u_t + uu_x = 0, \qquad x \in [-1, 1], \tag{3.21}$$

with the initial condition $u_0 = 1.5 + 0.5\sin(\pi x)$. The mesh contains 64 elements of which 32 elements are of size Δx and the rest are of size $\frac{\Delta x}{2}$, $\Delta x = 1/24$. The large and small elements are located alternatively. With the linear basis functions, the time step Δt^{improv} can be taken to be equal to $\frac{\Delta x}{3} \times 0.667$ (Table 3.5), instead of $\Delta t^{reg} = \frac{\Delta x}{3} \times 0.5$ corresponding to the size of the smaller cells. Results in Figure 3.17 show that the stability is preserved with the relaxed time restriction. In the top plots we present solutions obtained without use of a limiter as limiters can stabilize a weakly unstable scheme. Note that the amount of work to advance the solution to some time $t = \bar{T}$ with Δt^{improv} is equivalent to a local time stepping, i.e. taking 32 time steps of size Δt^{reg} and 64 steps of size $\Delta t^{reg}/2$ to advance the whole mesh in time by each Δt^{reg}. Admittedly, this example is more a numerical curiosity than a useful numerical technique as meshes of this

type do not appear in practice.

We also solve the problem on a mesh where the cells are arranged in two blocks so that 32 $\frac{\Delta x}{2}$ cells are followed by 32 Δx-sized cells. As expected, the solution with this time step is unstable.

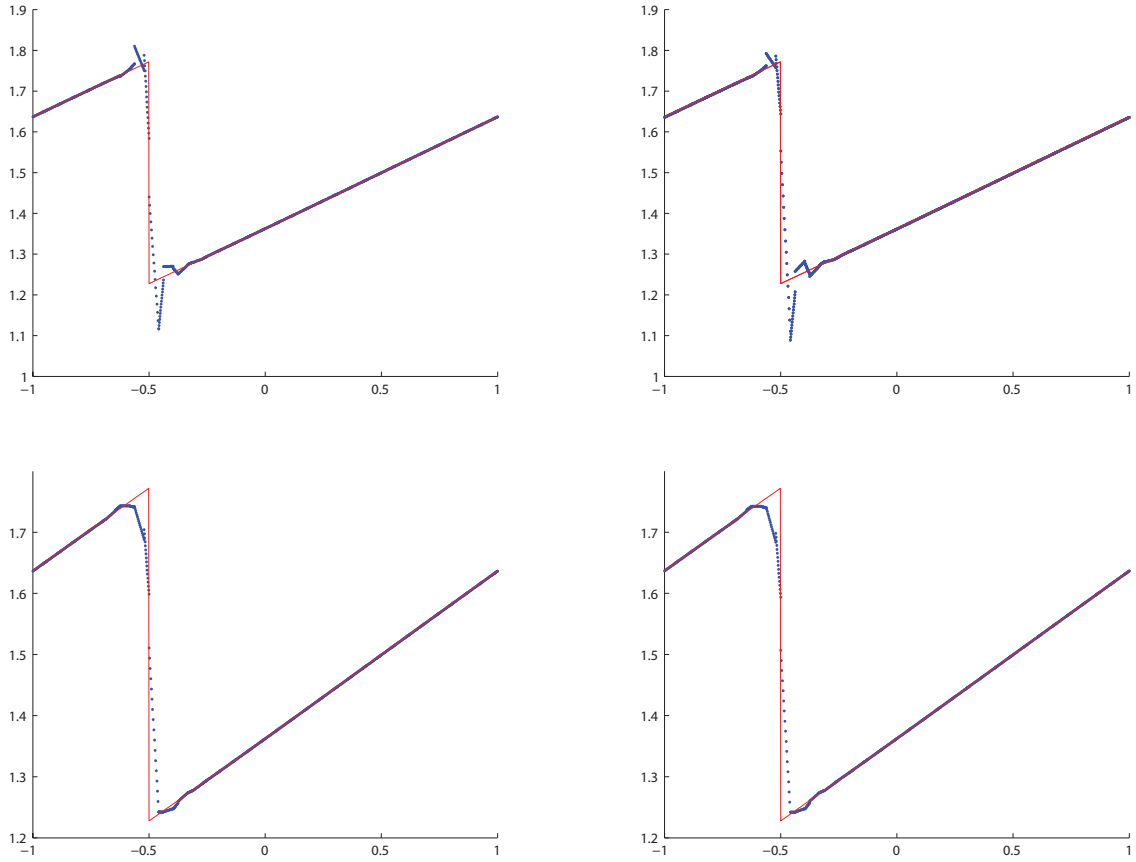

Figure 3.17: Numerical solutions of Burgers' equation with $p = 1$ at time $T = 3$, the left plots are computed with $\Delta t = 0.5\frac{\Delta x}{3}$ and the right ones with $\Delta t = 0.667\frac{\Delta x}{3}$. Top plots: no limiter; bottom plots: with minmod limiter.

3.2.3 Two-dimensional problems

Even though our analysis is not directly applicable to two-dimensional problems, we test if a time step larger than the one prescribed by the standard CFL conditions can be used for these problems as well. We do not aim to predict a stable time step here. Rather, we test what time step can be used before a solution becomes unstable and check if it is roughly in line with one-dimensional findings.

Two-dimensional linear advection

We consider the two-dimensional linear advection equation

$$u_t + au_x + bu_y = 0, \tag{3.22}$$

on the domain $\Omega = [-2,\ 2] \times [-2,\ 2]$ with the initial condition $u(x, y, 0) = \sin \pi(x + y)$ and periodic boundary condition. The velocity vector (a, b) is taken to be $(a, b) = (2, 1)$. The exact solution to this problem is $u(x, y, t) = \sin \pi(x + y - (a + b)t)$. We discretize the domain Ω into a Cartesian grid with 400 elements and then refine four of them by a factor of two (Figure 3.18). The CFL condition on a uniform Cartesian grid is given by [13]

$$\Delta t \leqslant \frac{\Delta x}{(2p + 1)(|a| + |b|)}, \tag{3.23}$$

where Δx is the size of the edges. Due to the presence of 16 small cells, the time step defined by the standard CFL condition for the mesh in Figure 3.18 would be a half of Δt given by (3.23). We ran a series of tests to find experimentally the largest stable time step. It is given approximately by

$$\Delta t \leqslant 0.95 \frac{\Delta x}{(2p + 1)(|a| + |b|)}, \tag{3.24}$$

where Δx is the size of the larger elements. Assuming that the linear growth of the spectrum as a ratio of small cells to the total number of cells is valid for two-dimensional problems, we have two bounds: global and direction-wise. The global bound on the growth of the spectrum is given by a factor $(1 + (m - 1)\alpha) = 1 + 16/(396 + 16) = 1.038$. Consequently the time step should be $1/1.038$ of the uniform case, i.e. $\Delta t \leqslant 0.96 \frac{\Delta x}{3(|a|+|b|)}$, which is slightly above (3.24). We can also look for a bound in the direction of the wave propagation and arrive at the factor of $1 + 4/22 = 1.182$ and $\Delta t \leqslant 0.85 \frac{\Delta x}{3(|a|+|b|)}$.

Euler equations

We consider a flow around a circular cylinder with the free flow Mach number equal to 0.38. The problem is modeled using two-dimensional Euler equations given by (1.2) We solve the

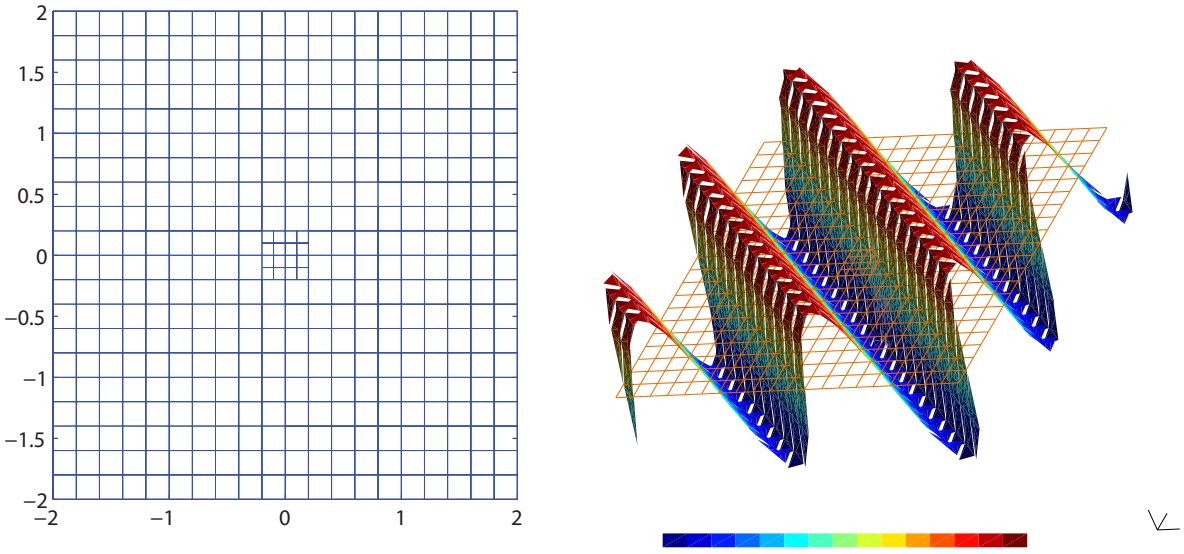

Figure 3.18: Left: A Cartesian grid with refinement. Right: Numerical solution at final time 10 using the time step given by (3.24).

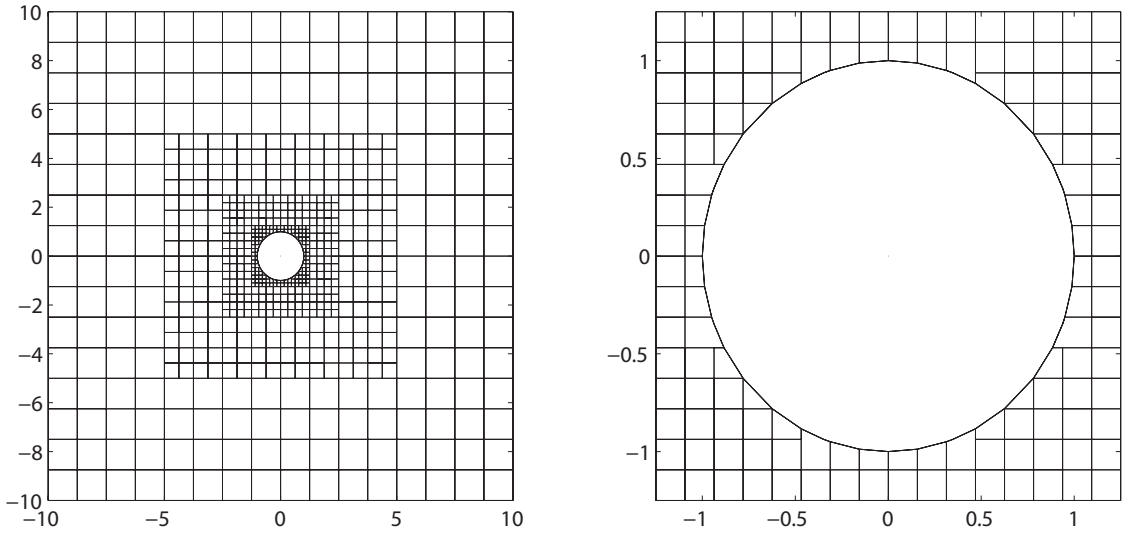

Figure 3.19: Left: A Cartesian grid around a circular cylinder. Right: Zoom near the surface of the cylinder.

problem on a 712 cell Cartesian grid with four levels of refinement shown in Figure 3.19. There are 36 irregular cells located on the surface of the cylinder. The cut cells which were deemed too small, i.e. less than a half of the size of the smallest regular cell, were merged with their neighbors (Figure 3.19, right). More details on the implementation can be found in [45]. We start computations with a uniform free flow and run them until the steady state is reached. Our experiment has revealed that similarly to the one-dimensional case discussed in Example 3.2.1, we can use a time step defined by the size of the smallest regular cell. In particular, the largest stable step can be taken to be 1.15 times the size defined by the smallest regular cell. We note that either a limiter or a smaller time step should be used at the beginning of computations to avoid instabilities due to high gradients in the transient flow.

Chapter 4

DGM on Cartesian Grids with Embedded Geometries

4.1 Introduction

Cartesian grid methods provide an appealing approach for solving problems in complex geometries due to the relative simplicity of mesh generation which is often called the bottleneck of computational fluid dynamics simulations. Creating a high quality body-fitted mesh is a time-consuming process often requiring significant user input. Cartesian mesh generation consists of cutting a geometry from the Cartesian grid which is a more straightforward process and one that can be made more automated [2]. However, it creates the so-called cut cells where the Cartesian grid intersects the boundary of the immersed body. Cut cells may have irregular shapes, which makes integration on them difficult. In addition, the size of such cells can be magnitudes smaller than the size of regular grid cells. Since the global time step is defined by the smallest cell in the mesh (1.18), the CFL condition can be very restrictive when an explicit time integration scheme is used. These two problems are major challenges in developing viable numerical methods for solving convection dominated problems on Cartesian grids.

A number of approaches have been proposed in the literature for dealing with the time step restriction. A particularly simple idea is to merge a small cut cell with its neighbors until a cell of a sufficient size is created [15, 5, 46]. Though conceptually simple, this method is difficult to implement robustly, especially for three-dimensional problems. This prompted development of other approaches such as h-box [29] and flux redistribution [16] methods for finite volume schemes. The h-box method works by reconstruction of the solution on a cut cell using the information contained in a box of the size of a regular cell. Flux redistribution allows only

part of the flux to be used to compute the solution on a small cell. A number of finite difference methods seek to avoid time step restriction and to take into account the curvature of the boundary in treating boundary points, e.g. [19, 50]. Finally, implicit time stepping is used in practical applications and for solving Navier-Stokes equations [44, 21, 42].

Computational savings provided by the structure of Cartesian grids can be especially beneficial for the computationally intensive discontinuous Galerkin methods. Nearly every aspect of evaluation of integrals over a regular cell's boundary and volume can be simplified by exploiting the regularity of the grid. For example, the mapping from physical elements to the canonical element is a simple scaling on Cartesian grids which can simplify and speed-up computing of gradients of the test functions. The issue that makes the Cartesian grid approach for the DGM particularly difficult is evaluation of the integral of the flux and a test function product in (1.21) on cut cells. The integration is usually performed using a numerical quadrature rule [34]. However, classical quadrature rules are known only for standard elements such as triangles, quadrilaterals, etc. While generalized quadrature rules for arbitrary polygons can be computed, e.g. [43], creation of such rules is expensive (a few minutes for each polygon) and very sensitive to round-off errors. This makes them unsuitable for our applications. An interesting approach was proposed in [21]. They generate a quadrature rule for each cut cell by randomly choosing sampling points and computing weights that guarantee that basis functions are integrated exactly. The algorithm requires the number of integration points to exceed the optimal number (upwards of 400 for $p = 5$) and might suffer from a poor choice of points [42] .

We propose to compute the cell volume integration by splitting each irregular cell into triangles and using a standard quadrature rule on each one of them. This has several advantages. Firstly, the quadrature rule is fast to generate and does not suffer from bad conditioning arising either because of the location of the integration points or the cell's shape. Secondly, it is more efficient as it does not involve storage of the values of basis functions at these points or reevaluation of them at each time step. The algorithm for splitting an arbitrary polygon into triangles is presented in Section 4.5. For an arbitrarily shaped cut or merged cell, it might, and often does, produce very thin triangles. This issue cannot be avoided without remeshing, i.e. moving vertices near the boundary, which is difficult to make robust. It is known that numerical approximation on such triangles can be poor [9]. For this reason, creating standard shape cells from cut cells is likely to be detrimental even when implicit time integration is used. However, it is easy to see and we show this in Section 4.5, that numerical quadrature does not suffer from thin element arising in splitting. For the same reason, i.e. to avoid ill-conditioning associated with mapping a badly shaped cell onto a canonical element, we define basis function on a rectangular bounding box containing a cut or merged cell.

We use explicit time integration and cell merging to avoid time step restriction imposed by small cut cells. The reasons for this are as follows. Implicit time integrators are prohibitively

expensive for the DGM where the number of degrees of freedom increases quickly with the order of approximation, especially in three dimensions. Also, one of the main advantages of the DGM, i.e. the locality of approximation, is lost when a global system is constructed in implicit schemes. More involved techniques like the h-box and flux distribution methods are not straightforward to adapt to the discontinuous Galerkin framework. Both methods stabilize the solution on a small cell by modifying the flux on its edges. However, the DGM has a contribution from the integral over cell volume which needs to be stabilized as well.

4.2 Mesh generation and description

Generation of Cartesian grids is a more straightforward process than creation of body fitted meshes. A fast and robust algorithm that we largely follow here is described in, e.g. [2]. Here we briefly describe components of the mesh generator in two dimensions.

Embedded geometry description

The embedded geometry can be described by an analytical expression or given as a ordered set of points. With analytical expressions, we can accurately compute the points where the geometry intersects the grid. A higher order geometrical approximation of the boundary is also easy to obtain. However, for complex geometries, the analytical description might not be available. A more general way to describe a geometry is to provide enough points located on the boundary. Usually, a fine description of the boundary is necessary in order to have enough data points to allow geometry driven mesh refinement. In the discussion below, we will assume that the embedded geometry is given by a set of points and line segments connecting them.

Data structure

Since most cells in a Cartesian grid are squares which are generated through a refinement process, the tree data structure seems to be natural. However, unstructured approach can also handle refinements easily with proper pointers to adjacent elements. Usually, each cell has connections to its vertices and edges, while each edge has pointers to its endpoints and adjacent cells. Connections between edges can be added if a line segment on top of an existing edge is created by refinements.

Painting algorithm

Before dealing with refinement and cut cells, we need to identify whether an element is inside the domain. We solve this problem with the ”ray-casting” approach. For any point not on the boundary, we draw a ray from it and count the number of intersections with the boundary. If the number is even, the starting point is outside the geometry, otherwise it is inside. When all the

vertices of an element are inside, we classify it as an inside element. One issue we need to be careful of is that an arbitrary ray might be tangential to the boundary and give an odd number of intersections. We suggest to cast more than one ray to avoid this issue. Then, the painting algorithm starts from one element where we determine its "color", i.e. if it is inside or outside, and paints its neighboring elements with the same color if they do not intersect the boundary. This algorithm speeds up the process since it greatly reduces the number of elements which need "ray-casting" tests.

Cut cells and refinements

Cut cells appear when the boundary of the embedded geometry cuts through a Cartesian cell. In the algorithm, we need to search for the boundary line segments to find candidates which intersect a Cartesian cell, and then create new vertices and edges for the cut cell. We can launch refinements based on the shape of those cut cells. Several boundary line segments might be included in a single cut cell, so we introduce the total angular variation which is defined as the sum of the difference in direction of two adjacent line segments. This variable measures the change in direction of the boundary edges within a cell. We can assign a threshold for the angular variation. If the variation is greater than the threshold, we decide to refine this cut cell. After looping over all cut cells once, the refinement is propagated several layers into the mesh by refining neighboring cells. Then we check the new cut cells again to see if they need further refinements until the angular variation in each cut cell is less than the threshold or the refinement level reaches a predetermined maximum refinement level.

We assume that the constructed Cartesian grid resolves the geometry to a reasonable degree so that simulations can be carried out. However, we do not restrict types of admissible cut cells. We use straight-sided elements and employ the boundary conditions described in Section 4.4 to compensate for the curvature of the solid wall. Thus, our cut cells are polygons.

When a background frame element intersects the embedded geometry, the simplest case occurs when the element is split into two parts by a straight line: one part inside the computational domain and the other outside. Figure 4.1 shows three possible types of elements in the simplest case. More complicated configurations are illustrated in Figure 4.2. In the region where the embedded body is thin, the background cell might be divided into several parts Figure (4.2, left). When a cell is split into more than two parts, it is natural to treat each part as an independent cell since different parts belong to different flow regions. A part of a cell can be cut out by the geometry (Figure 4.2, right) such as a tail of the airfoil. Assuming linear approximation of the geometry, this type of a cell can be described by a polygon, which is usually concave. Coirier and Powell [15] suggest to refine this frame element until the resulting elements are intersected in the simplest way. Also, it can be argued that an approximation on such a cell might be of poor quality. However, there is no guarantee that we can reach the aimed case in an acceptable

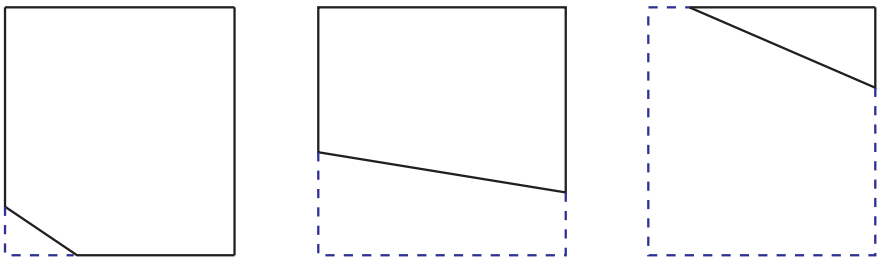

Figure 4.1: Simplest cases of cut cells. The dashed lines indicate the part of the cell outside the computational domain.

number of refinements. For this reason, we choose to accept cells of this type once the specified level of refinement is reached.

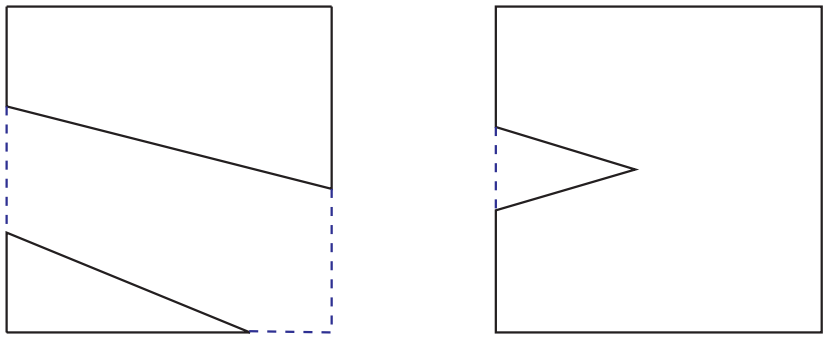

Figure 4.2: More complicated cut cells. The dashed lines indicate the part of the cell lying outside the computational domain.

4.3 Cut cell merging

Using Cartesian grids to discretize the computational domain with an embedded geometry results in fragmented cut cells near the boundaries. Cut cells might have small sizes which would lead to a restrictive CFL number and, consequently, a small time step when an explicit time integrator is used. To avoid this, we will merge a cut cell with a neighboring cell or cells so that the resulting element is comparable in size to the regular cells. If the Cartesian grid has several levels of refinement, e.g. Figure 4.15, we choose the size of the smallest cell as a comparison. We seek to merge a cell if its size is less than a half of a regular cell. We make an assumption here that there is always a neighboring cell or cells such that a merge will result in a reasonably sized cell. It is not difficult to imagine an example where this is not possible. However, in such a case a body fitted grid will also result in a great variation of cell sizes and explicit time integration might not be a suitable approach.

Normally, a cut cell has several neighbors and, consequently, several candidates to be merged with. Obviously, the optimal merging will result in a merged cell of the best possible size and shape. To this effect we want to avoid poorly proportioned elements, e.g., thin triangles (Figure 4.3, top) or thin polygons (Figure 4.3, bottom). To measure irregularity of a polygon shaped cut cell, we enclose it into a bounding box defined as the smallest rectangle with the edges parallel to the main axis containing the cell (Figure 4.3). Denoting the vertices of an element by $\{P_i(x_i, y_i)\}_{i=1}^n$, the bounding box of this element is $B = \{(x, y) | \min_i(x_i) \leqslant x \leqslant \max_i(x_i), \min_i(y_i) \leqslant y \leqslant \max_i(y_i)\}$. Note, that the cells in Figure 4.3 might even be created by discretization of simple objects. The short edge of a bounding box (Figure 4.3, left) can be magnitudes smaller than the long edge. This would result in not only a small CFL condition but a poor approximation as well. The measure of thinness is the aspect ratio of the bounding box, i.e., the ratio of the lengths of the short side to the long side of the box.

We also seek to avoid badly shaped cell such as the polygon ABCGHD in Figure 4.4. The triangular cut cell CGH can be merged with the cell ABCD to the right of it or the cell DHGFE below it. Based on the criterion of choosing the merging direction resulting in the largest bounding box would require merging with the right cell ABCD which would create a badly shaped cell ABCGHD. To avoid this situation, we make an additional requirement that the bounding box of the merged cell does not intersect with the neighboring cells not involved in the current merging process, e.g. cell EDHGF in Figure 4.4. In this example, the better merging direction results in the merged cell DCGFE.

Finally, direction of merging is decided according to the following algorithm. If a small cut cell has only one neighbor we simply merge them together. Otherwise, for each cut cell, we construct bounding boxes for all its neighboring cells, i.e., cells that share an edge with this cut

cell. We eliminate merging directions whose bounding boxes overlap with the other neighboring cells. Among the rest, we choose the merge resulting in a bounding box with the largest aspect ratio. This operation can be performed until the merged cut cells have not reached the desired size. Figure 4.3 illustrates the merging process for the majority of cells. If the cut cell E1 is

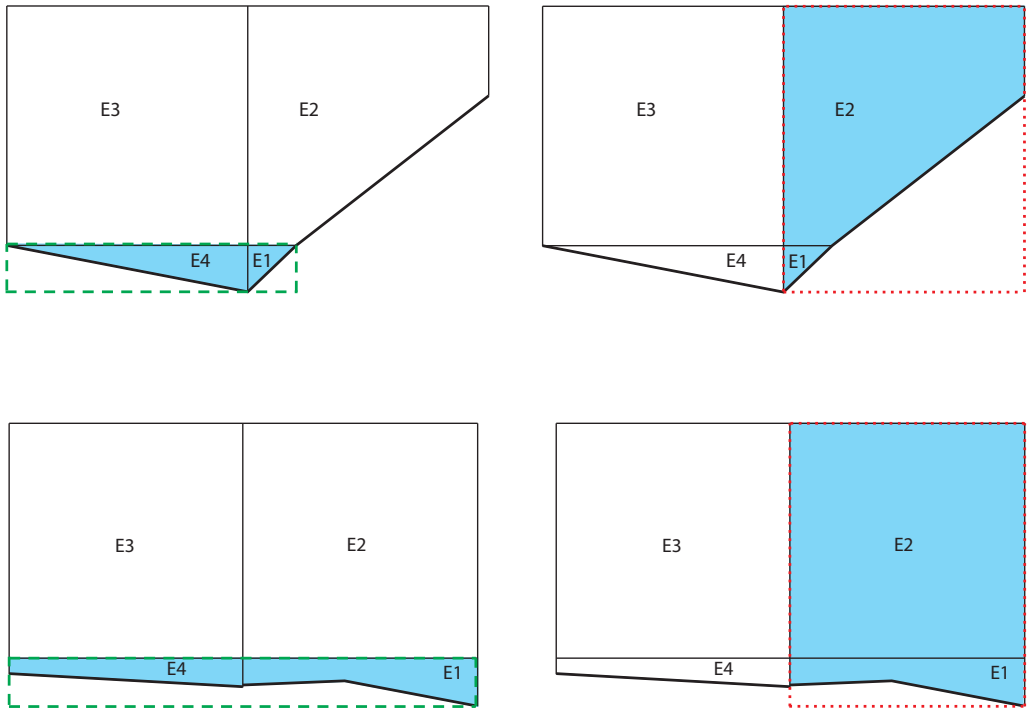

Figure 4.3: Illustration of the direction of merging, plots on the left indicate unacceptable merging direction, and plots on the right show the acceptable merging direction.

merged with its left neighbor E4, the merged cell is still inside a thin bounding box (dashed line). Since merging the cut cell with its upper neighbor E2 will give us a better bounding box (dotted line), we choose to merge it upwards and the merged cell is shown shaded.

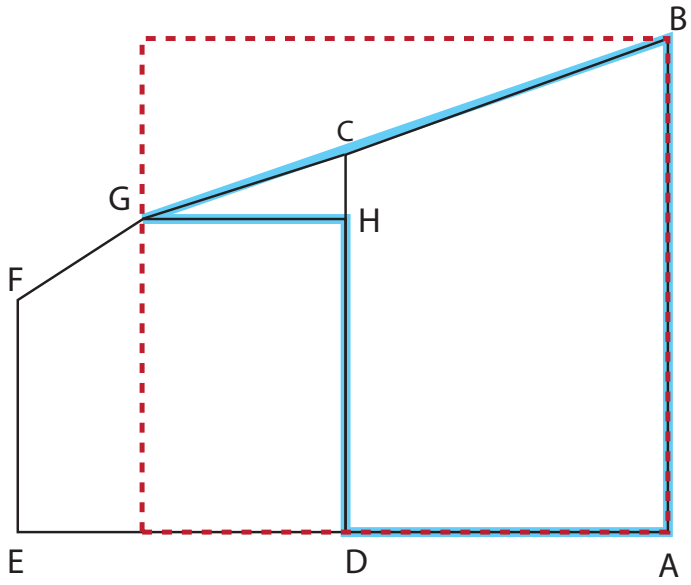

Figure 4.4: Badly shaped merged cell ABCGHD where the bounding box (shown as a dashed line) overlaps the neighboring cell.

4.4 Solid wall boundary conditions

In many applications, embedded geometries are described by curved and complex boundaries. When the surface and the computational domain are discretized with straight-sided edges and elements, such as the ones described in the previous section, a large numerical error is committed near the boundary and a reduced overall convergence rate is observed [9, 37]. The proper way to deal with this issue is to use high-order geometric approximation of the boundary. However, these approximations are not always available, as in our case.

Instead, we follow the approach proposed in [37]. Regular solid wall boundary conditions impose no flow through the wall requirement by creating a ghost state at boundary integration points and setting it to be the mirror reflection of the inner state. The key idea of the curvature

boundary condition is to reflect the flow with respect to the physical (curved) boundary rather than the computational domain. As shown in Figure 4.5, velocity $\boldsymbol{v}$ at an integration point is sought to be orthogonal to the physical normal $\boldsymbol{N}$ instead of the normal to the computational boundary $\boldsymbol{n}$

$$\boldsymbol{v} \cdot \boldsymbol{N} = 0. \tag{4.1}$$

To achieve this, ghost state values at each integration point are computed using the physical normal $\boldsymbol{N}$ and tangential vector $\boldsymbol{T}$

$$\begin{cases} \rho^g &= \rho, \\ \boldsymbol{v}_N^g &= -\boldsymbol{v}_N, \\ \boldsymbol{v}_T^g &= \boldsymbol{v}_T, \\ p^g &= p, \end{cases} \tag{4.2}$$

where the superscript g denotes the ghost state and subscript N, T refers to the projection of the velocity vector onto $\boldsymbol{N}$ and $\boldsymbol{T}$ respectively. These values are passed to a numerical flux evaluator. The resulting Riemann state at the boundary mimics the flow around the physical domain. This has been shown to significantly improve accuracy and convergence rates.

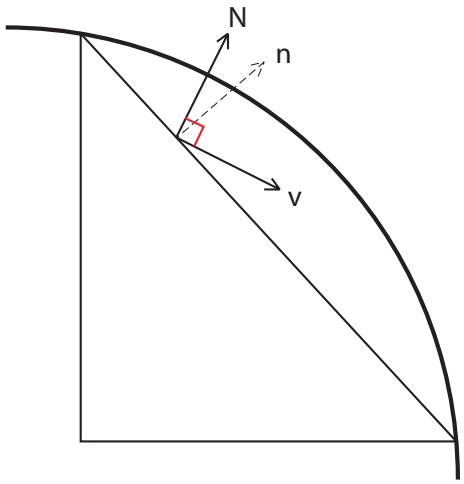

Figure 4.5: Illustration of curvature boundary conditions.

4.5 Volume integration

In order to compute the integral over cell volumes, we need a quadrature rule defined on each computational cell. We create such a rule by splitting a cut cell into triangles. In the case of straight-sided elements, i.e. when the cut cell is a polygon, the splitting algorithm is presented in Section 4.5.1. In the case of curved elements, more care needs to be taken to insure that a triangle being split off does not contain a part outside the domain. We can split polygon Ω_0 into triangles $\{T_i\}_{i=1}^N$ for which we have classic quadrature rules and sum those integrals up to get the integral on Ω_0

$$\int_{\Omega_0} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \sum_{i=1}^{N} \int_{T_i} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{4.3}$$

Each of the integrals $\int_{T_i} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ is calculated by mapping a canonical triangle $\hat{T}$ to the triangle T_i. Let us say the mapping is given by

$$F : \boldsymbol{\xi} \to \boldsymbol{x},\ \boldsymbol{x} = A_i\boldsymbol{\xi} + \boldsymbol{b}_i. \tag{4.4}$$

Since the mapping from T to T_i is linear, A_i is constant matrix and we can write

$$\int_{T_i} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \det(A_i) \int_{\hat{T}} \hat{f}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}. \tag{4.5}$$

A quadrature rule on the canonical element $\hat{T}$ automatically induces a quadrature rule on the element T_i

$$\int_{T_i} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx \det(A_i) \sum_{j=1}^{N_i} \hat{\omega}_j \hat{f}(\boldsymbol{\xi}_j) = \det(A_i) \sum_{j=1}^{N_i} \hat{\omega}_j f(\boldsymbol{x}_j). \tag{4.6}$$

Then, the quadrature rule on the entire element is

$$\int_{\Omega_0} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx \sum_{i=1}^{N} \det(A_i) \sum_{j=1}^{N_i} \hat{\omega}_j f(\boldsymbol{x}_j) = \sum_{l=1}^{L} \omega_l f(\boldsymbol{x}_l). \tag{4.7}$$

We introduce the quadrature error

$$E(f) = \int_{\Omega_0} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \sum_{l=1}^{L} \omega_l f(\boldsymbol{x}_l), \tag{4.8}$$

and assume the quadrature rule we use is accurate for polynomials of degree up to p. Let us say that B_0 is the bounding box of Ω_0, which has size Δx, and all $(p+1)$th order derivatives of $f(x, y)$ are bounded by M_{p+1}. Following analysis in Section 7.2 in [34], the error has an upper bound

$$|E(f)| \leqslant \frac{(2\Delta x)^{p+1}}{(p+1)} 2M_{p+1} \int_{\Omega_0} \mathrm{d}\boldsymbol{x}. \tag{4.9}$$

For quadrature rules with non-negative weights, this upper bound is related to the size of the entire element Ω_0, and the $(p+1)$th derivatives of the integrand. Regardless a chosen subdivision of Ω_0 into triangles for integration, thin or small triangles will not affect the accuracy of numerical integration on the entire element.

4.5.1 Polygon splitting

In this section we describe an algorithm that we use for splitting an arbitrary polygon into triangles. The algorithm recursively splits off a triangle from a given polygon until the remainder is a triangle itself. If the polygon is convex, connecting a vertex to its neighbors gives us a candidate for a triangle to be split. However, since we can not guarantee a cut cell to be a convex polygon, we have to be careful not to create a triangle which does not belong to the domain.

1. Let us assume that a polygon is described by an ordered set of vertices $\{V_j(x_j, y_j)\}_{j=1}^n$. First, we check that vertices are ordered in the counterclockwise direction. We find the left most vertex $V_l(x_l, y_l)$, $x_l = \min_j(x_j)$. Then, the angle $\angle V_{l-1}V_lV_{l+1}$ should be less than π since V_{l-1} and V_{l+1} are to the right of V_l. Calculating the cross product $\overrightarrow{V_{l-1}V_l} \times \overrightarrow{V_lV_{l+1}}$ and checking for the direction of the resulting vector, we determine if the vertices need to be re-ordered. The positive direction corresponds to the counterclockwise ordering.

2. We start from any vertex of the current polygon V_i and check whether the angle $\angle V_{i-1}V_iV_{i+1}$, $i = 1, \ldots, n$ is less than π by calculating the cross product $\overrightarrow{V_{i-1}V_i} \times \overrightarrow{V_iV_{i+1}}$.

$$\text{counterclockwise} \begin{cases} \angle V_{i-1}V_iV_{i+1} < \pi, & \text{if } \overrightarrow{V_{l-1}V_l} \times \overrightarrow{V_lV_{l+1}} \text{ positive direction,} \\ \angle V_{i-1}V_iV_{i+1} > \pi, & \text{if } \overrightarrow{V_{l-1}V_l} \times \overrightarrow{V_lV_{l+1}} \text{ negative direction.} \end{cases} \tag{4.10}$$

3. If $\angle V_{i-1}V_iV_{i+1} > \pi$, as shown in Figure 4.6, left, the triangle $\Delta V_{i-1}V_iV_{i+1}$ will not be inside the cell and, consequently, the computational domain. So, we mark triangle $\Delta V_{i-1}V_iV_{i+1}$ for splitting only if $\angle V_{i-1}V_iV_{i+1} < \pi$. Before proceeding, we check if any of the rest of the vertices is located inside $\Delta V_{i-1}V_iV_{i+1}$ such as, e.g., vertex V_{i+2} in Figure 4.6, right.

This is intended to check if the triangle marked for splitting is only partly contained inside the domain. Obviously, this should be avoided. If $\Delta V_{i-1}V_iV_{i+1}$ cannot be split from the polygon, we move on to check the next vertex. Since the triangular subdivision exists for any polygon, there exists a triangle that can be split in this way.

4. Repeat step 3 to the polygon formed by $\{V_j\}_{j=1}^n \backslash V_i$ until only 3 vertices left. This process splits the polygon into $n-2$ triangles.

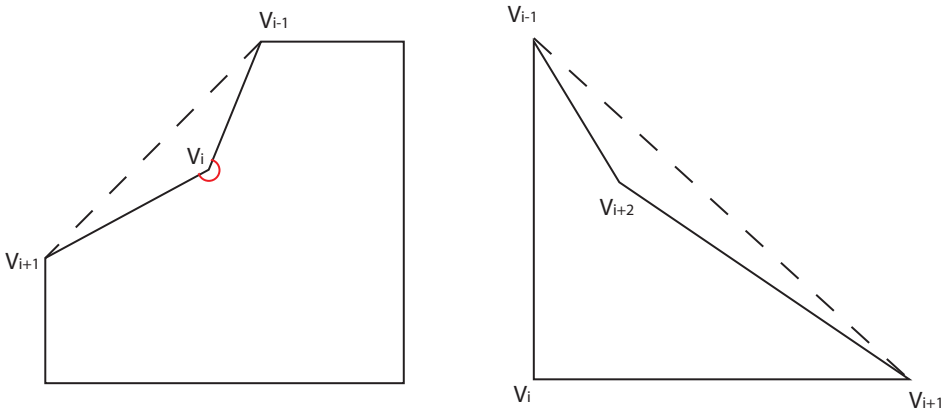

Figure 4.6: Splitting of triangle $\Delta V_{i-1}V_iV_{i+1}$.

Figure 4.7 illustrates the process of splitting a typical cut cell. Since the majority of cut cells have this relatively simple shape, the cost of integrating of them is modest: only two or three times as many points as on a regular triangle. Figure 4.8 illustrates a possible subdivision of a more complex cell. Note that starting the described algorithm with vertex V_1 will result in an inadmissable triangle $\Delta V_7V_2V_3$. This triangle is rejected with the algorithm moving to the next vertex.

4.5.2 Basis functions on cut cells

Although we split a complex cut cell Ω_j into triangles to calculate the volume integrals, the basis functions are defined on the whole element Ω_j. In particular, the basis functions are defined on the rectangular bounding box B_j which is then mapped to the canonical square $[-1,1]\times[-1,1]$.

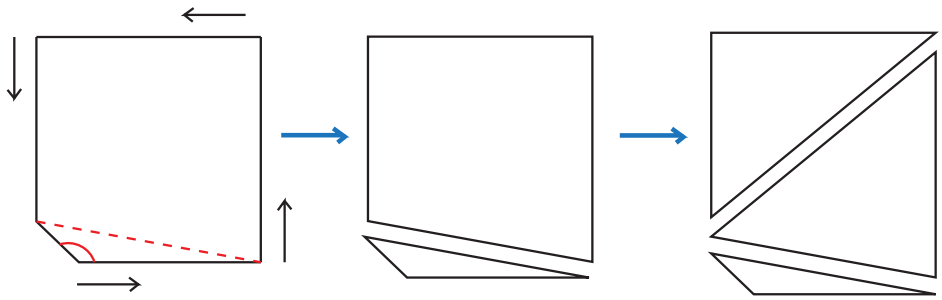

Figure 4.7: Division of a cut cell into triangles.

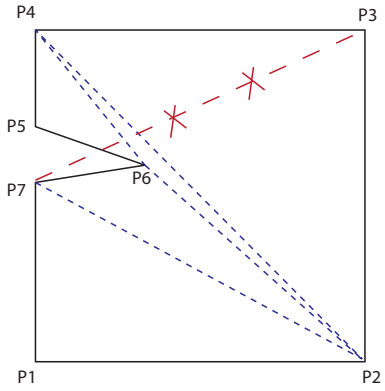

Figure 4.8: Division of a cut cell into triangles.

This avoids the issue of poorly conditioned mappings. We then define the basis as a tensor product of the one-dimensional Legendre polynomials $P_k(\xi)$, $k = 0, 1, \ldots$ [1]. Thus, $\varphi_{ij} = P_i(\xi)P_j(\eta)$, $i = 0, 1, \ldots, p$, $j = 0, 1, \ldots, p$. We can reorder φ_{ij} with a single index and denote them as φ_i, $i = 1, 2, \ldots, (p+1)^2$ in the local representation (1.8). This basis is not orthogonal on cut cells which makes the mass matrix not diagonal. However, the number of cut cell is

small compared to the total number of elements and the extra expense in evaluating, inverting and storing the mass matrix is modest. The basis can be made orthogonal using a Gram-Schmidt orthogonalization. However, this process can be ill-conditioned for badly shaped cut-cells [43].

4.6 Numerical examples

In this section we present a number of computational examples. All problems are based on solutions of the Euler equations given by (1.2). Computations were performed until the solution reached a steady state, defined as the difference between solution coefficients in two successive time steps being on the order of machine precision.

4.6.1 Supersonic vortex

We consider a supersonic isentropic flow between two concentric circular arcs $\Omega = \{(r,\theta)|1 \leqslant r \leqslant 1.384,\ 0 \leqslant \theta \leqslant \frac{\pi}{2}\}$ (Figure 4.9). This problem has a known analytical solution with the exact density, velocity, and pressure given by

$$\rho = \rho_i \left(1 + \frac{\gamma - 1}{2} M_i^2 (1 - (\frac{r_i}{r})^2)\right)^{\frac{1}{\gamma - 1}}, \tag{4.11}$$

and

$$\|\overrightarrow{v}\| = \frac{c_i M_i}{r}, \quad P = \frac{\rho^\gamma}{\gamma}, \tag{4.12}$$

where c_i is the speed of sound on the inner circle and $\gamma = 1.4$ for air. Initially, the Mach number at the inner circle is 2.25 and the density is equal to one. We impose the solid wall boundary conditions on the arcs. The inflow and outflow boundary conditions are imposed on the straight segments of the boundary. The problem is solved on a sequence of uniform Cartesian grids shown in Figure 4.9. Finer grids are obtained from coarser ones by successive division by a factor of two. We compute the exact L^1 errors in density and pressure and report them in Tables 4.1 and 4.2. We tabulate the computations obtained with the curvature boundary conditions (4.2) and the exact boundary conditions, i.e. the boundary conditions given by (4.11) and (4.12). We observe the theoretical $p + 1$ convergence rate for both the exact and curvature boundary conditions with the errors being quite close. The convergence rates were calculated based on the size of regular cells not on the number of elements in the grid. Note that due to the presence of irregular cells, the total number of elements in the grids does not increase precisely by a factor of four under h-refinement.

	Curvature BCs				Exact BCs			
p	p=1		p=2		p=1		p=2	
N	error	rate	error	rate	error	rate	error	rate
156	1.84E-3	-	4.99E-5	-	1.84E-3	-	5.07E-5	-
614	4.26E-4	2.11	4.40E-6	3.50	4.26E-4	2.11	4.44E-6	3.51
2422	1.05E-4	2.02	6.39E-7	2.79	9.83E-5	2.12	4.99E-7	3.15
9661	2.49E-5	2.07	6.26E-8	3.35	2.39E-5	2.04	5.41E-8	3.21

Table 4.1: L^1 errors in density and rates of convergence for the supersonic vortex.

	Curvature BCs				Exact BCs			
p	p=1		p=2		p=1		p=2	
N	error	rate	error	rate	error	rate	error	rate
156	1.58E-3	-	5.88E-5	-	1.58E-3	-	5.98E-5	-
614	3.15E-4	2.33	4.77E-6	3.62	3.15E-4	2.33	4.81E-6	3.64
2422	7.47E-5	2.07	7.17E-7	2.73	7.04E-5	2.16	4.96E-7	3.28
9661	1.69E-5	2.14	6.45E-8	3.47	1.63E-5	2.11	4.92E-8	3.33

Table 4.2: L^1 errors in pressure and rates of convergence for the supersonic vortex.

We also present the results of computations using the regular reflecting boundary conditions (Tables 4.3 and 4.4). The results are poor, as expected, due to the large error near the curved boundaries. Note that the accuracy of the solution decreases when the degree of the approximation is increased from $p = 1$ to $p = 2$. This is due to the ability of the higher order approximation to resolve non-physical rarefaction fans at the sharp corners of the straight-sided boundary. This is discussed in more details in [37].

The meshes used in computations contain cut cells along the curved boundaries. In Figure 4.9, bottom, we show a schematic zoom of two irregular cells. The top cell is a result of merging, the lower cell is a purely cut cell. The subdivisions of the cells into triangles for the purpose of numerical integration are indicated by dashed lines. This splitting created thin triangles with the smallest angles equal to 0.75^o, 3^o, and 4^o (seen in thin triangles from top to bottom of the two zoom inserts). Since both cells have one small edge, it is impossible to split them without creating such badly shaped subelements. Despite this, the solution behaves well and converges with the correct rate.

To better understand the behavior of the error, we compare solutions with $p = 1$ on the finest

p	p=1		p=2	
N	error	rate	error	rate
156	1.01E-2	-	1.60E-2	-
614	2.82E-3	1.84	4.75E-3	1.75
2422	1.09E-3	1.37	2.05E-3	1.21

Table 4.3: L^1 errors in density and rates of convergence for the supersonic vortex using the reflecting boundary conditions.

p	p=1		p=2	
N	error	rate	error	rate
156	1.39E-2	-	2.15E-2	-
614	3.80E-3	1.87	6.30E-3	1.77
2422	1.46E-3	1.38	2.73E-3	1.21

Table 4.4: L^1 errors in pressure and rates of convergence for the supersonic vortex using reflecting boundary conditions.

Cartesian grid in Figure 4.9 and a body-fitted triangular grid shown in Figure 4.10. The triangular mesh has nearly uniform elements with edges of the cells lying on the straight sides being close to the size of the Cartesian grid cells. Thus, the characteristic mesh size is used as a basis for comparison rather than consideration of efficiency or accuracy. The regions where the error is greater than 80% of the maximum L^∞ error are shown in dark. We observe that the error seems to be larger on the inner circle than elsewhere in the domain. However, there is no clear pollution error originating from the boundaries.

We further investigate the error on cut cells. To do so, we compute the average error in density only on cut cells in the L^1 norm

$$e_{cut} = \frac{\sum_{cutcells} \int_{\Omega_c} |\rho - \rho_{exact}| dx}{\sum_{cutcells} \int_{\Omega_c} dx}. \tag{4.13}$$

The results are presented in Table 4.5 for the curvature boundary conditions and the exact boundary conditions. We observe that the errors with the two types of boundary conditions are close. The quadratic approximation has the desirable $p + 1$ convergence rate. However, the linear approximation has a reduced 1.5 convergence rate. A possible explanation is that the linear approximation cannot capture the complex behavior of the flow near the boundary on cut cells. To

test this, we solve a linear problem

$$u_t + 2u_x + u_y = 0, \tag{4.14}$$

on the same sequence of meshes with the exact solution $u(x, y) = \sin \pi(x - 2y)$. We report the result in Table 4.6 where the theoretical rates of convergence can be seen.

	Curvature BCs				Exact BCs			
p	p=1		p=2		p=1		p=2	
N	error	rate	error	rate	error	rate	error	rate
156	3.61E-3	-	7.58E-5	-	3.37E-3	-	6.98E-5	-
614	1.22E-3	1.57	7.45E-6	3.35	1.08E-3	1.64	6.72E-6	3.38
2422	4.47E-4	1.44	1.26E-6	2.57	3.31E-4	1.71	8.75E-7	2.94
9661	1.67E-4	1.43	1.49E-7	3.07	1.21E-4	1.45	1.09E-7	3.01

Table 4.5: Average L^1 errors in density on cut cells only and rates of convergence for the supersonic vortex example.

p	p=1		p=2	
N	error	rate	error	rate
156	6.40E-3	-	2.21E-4	-
614	1.52E-3	2.08	2.99E-5	2.89
2422	4.07E-4	1.90	3.85E-6	2.96
9661	9.97E-5	2.03	4.66E-7	3.05

Table 4.6: Average L^1 errors on cut cells and rates of convergence for the linear problem.

4.6.2 Flow around a cylinder

Next, we consider a subsonic flow around a circular cylinder. The cylinder of radius $r = 1$ is located in the center of the domain $[-10, 10] \times [-10, 10]$ as shown in Figure 4.12. We solve the problem on a sequence of meshes. Each mesh has two levels of refinement by a factor of two. The overall structure of the coarsest mesh is shown in Figure 4.12, left; a zoom of the area near the surface and merged cut cells is shown on the right plot. Finer meshes are obtained by refining each cell in a coarser mesh by a factor of two with the cells located on the boundary

p	p=1		p=2	
N	error	rate	error	rate
176	1.38E-1	-	6.20E-3	-
712	1.85E-2	2.90	6.95E-4	3.16
2828	2.92E-3	2.66	9.81E-5	2.83

Table 4.7: L^1 errors of entropy and rates of convergence for flow around cylinder.

updated with a more accurate geometric description. Computations are initialized with free flow values as follows: the Mach number is $M_\infty = 0.38$, the density and pressure are $\rho_\infty = 1.4$, $P_\infty = 1.0$. Figure 4.13 shows Mach number isolines with $p = 1$ and $p = 2$ on the sequence of meshes. We observe that the quality of the solution improves under both h- and p-refinement. In particular, the solutions with $p = 2$ do not have a kink due to spurious entropy production on the surface. Next, we investigate convergence under h-refinement. We measure errors in entropy defined as

$$\epsilon_{ent} = \frac{P}{P_\infty}/(\frac{\rho}{\rho_\infty})^\gamma - 1. \tag{4.15}$$

The results in the L^1-norm are reported in Table 4.7. The $p + 1$ rate of convergence is observed.

To apprise solution accuracy on the surface we calculate the total pressure loss coefficient

$$P\left(1 + \frac{\gamma - 1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}} / \left[P_\infty\left(1 + \frac{\gamma - 1}{2}M_\infty^2\right)^{\frac{\gamma}{\gamma-1}}\right]. \tag{4.16}$$

Figure 4.14 shows the total pressure loss coefficient on the surface for calculations performed with $p = 1$ and h-refinement. We notice that this quantity converges to unity with decreasing Δx, as expected.

4.6.3 Flow around NACA0012 airfoil

In this example, we test the method in the presence of more complicated cut cells. The geometry of the NACA0012 airfoil geometry is given by

$$y = \pm 0.6(0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4).$$

The airfoil is enclosed in the box $[-10, 10] \times [-10, 10]$ and shifted by a small amount in the vertical direction so that the resulting mesh is not symmetric. The mesh used in computations

is shown in Figure 4.15. The bottom plot shows a badly shaped element at the tail end of the airfoil. There are one hundred cells lying on the surface which makes the mesh relatively coarse. We observe that the merging algorithm described in Section 4.3 merges cut cells in the vertical direction. It can be argued that it is a poor choice for this example as near the surface the flow changes slower in the horizontal direction than in the vertical. However, the issue of anisotropic mesh refinement is not being considered here. It is assumed that anisotropy in the mesh would be created by either a mesh generator or a run time mesh refinement.

We perform computations with $M_\infty = 0.63$, $\rho_\infty = 1.$, $P_\infty = 1.4$ and the angle of attack $\alpha = 2^o$. The isolines of the pressure and Mach number near the surface with $p = 1$ are presented in Figure 4.16. The region near the tail is well resolved without a visible wake in the pressure. We also plot the isolines of the error in entropy in Figure 4.16. There is a noticeable entropy production at the tip of the airfoil which propagates along the surface (Figure 4.17 and Figure 4.16, right). This can be attributed to the coarse grid at the tip of the airfoil (Figure 4.15). The lift and drag coefficients are 0.313 and $5.27e-03$. The results agree with body fitted computations [37] though the drag is larger.

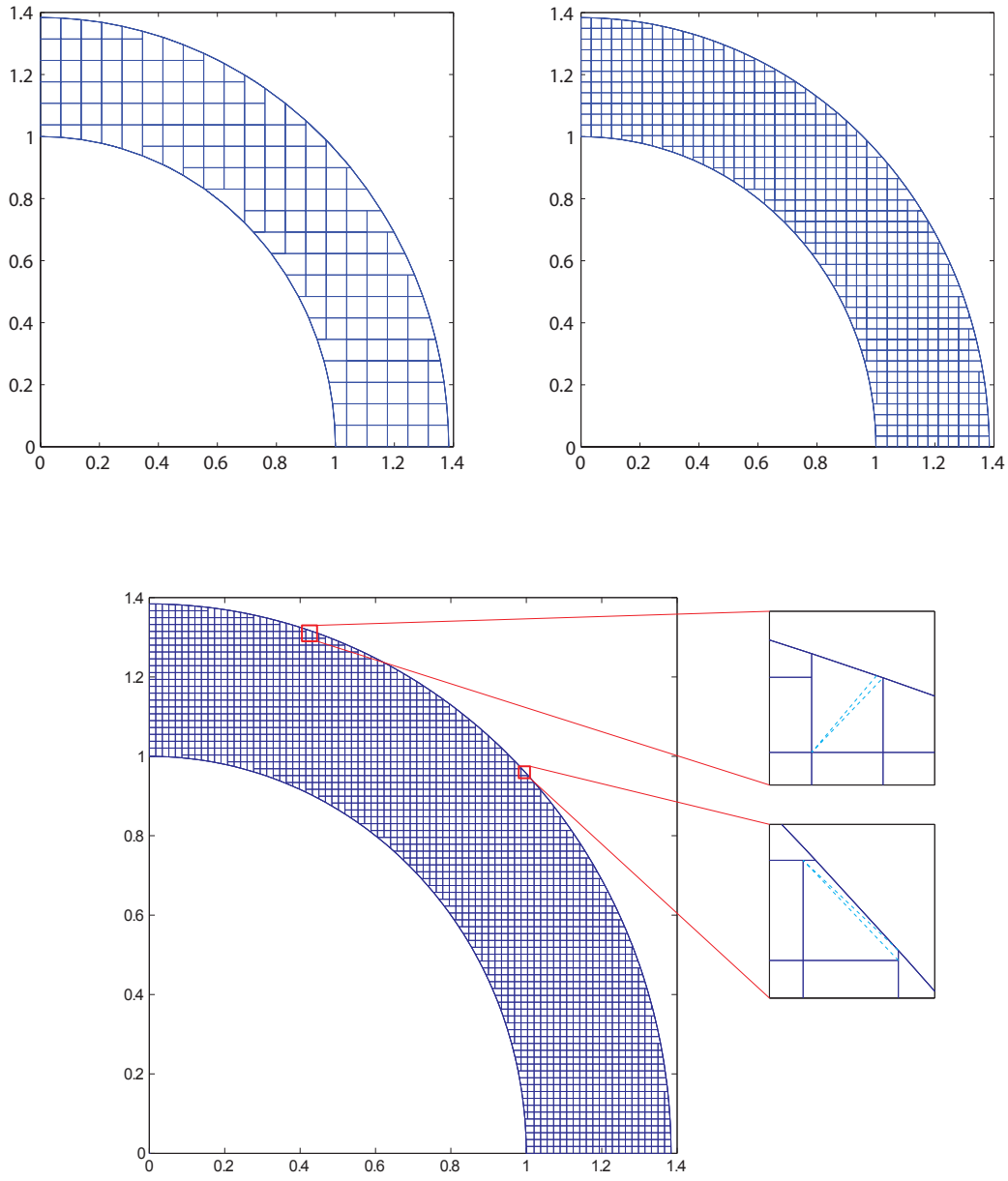

Figure 4.9: Cartesian grids used for the supersonic vortex example (after small cut cell merging.

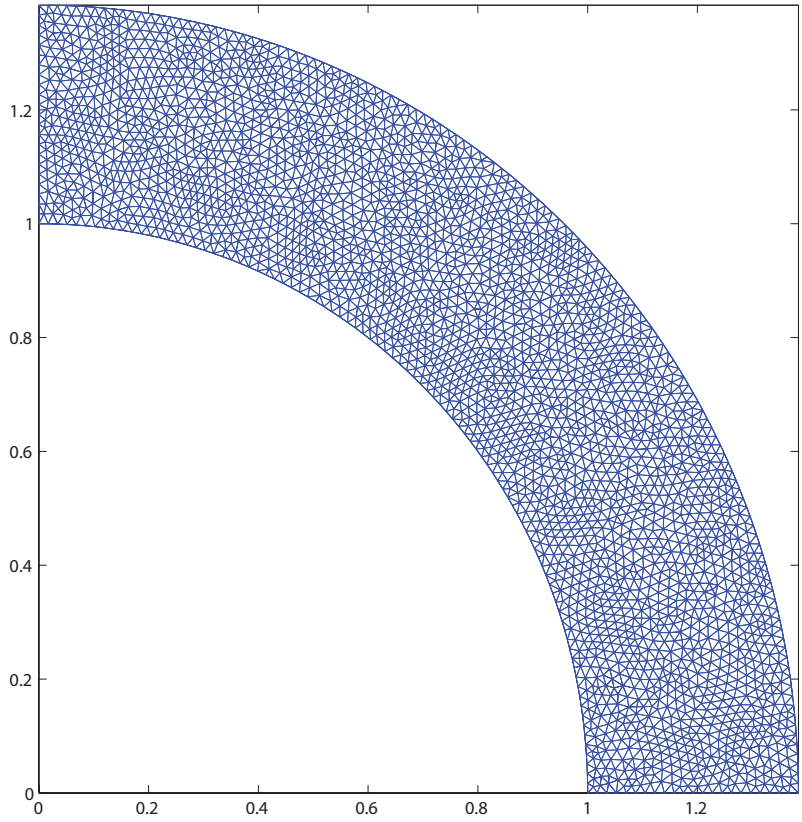

Figure 4.10: Body-fitted grid for the supersonic vortex example.

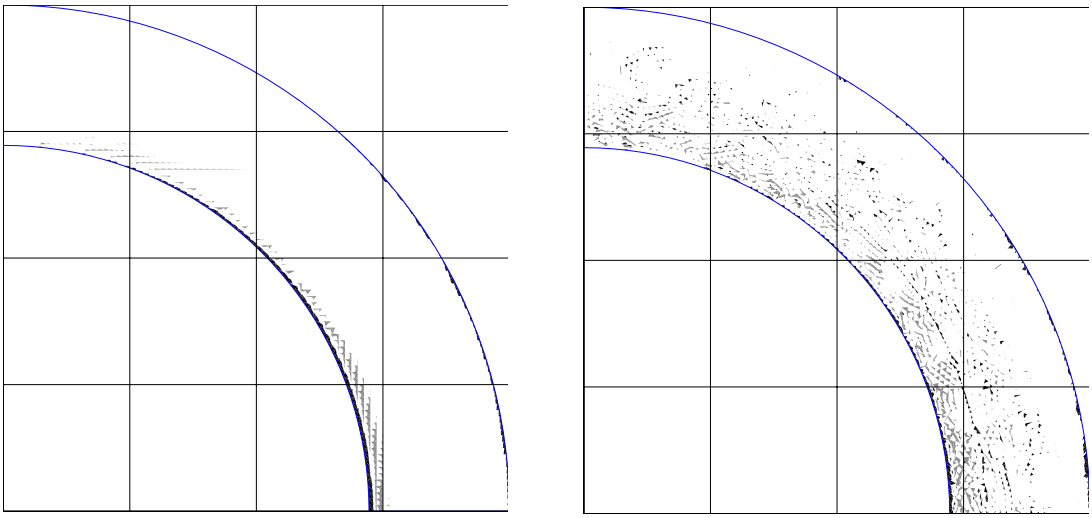

Figure 4.11: Error in density with $p = 1$ for the supersonic vortex problem on a Cartesian grid (left) and on a triangular mesh (right). Darker regions indicates where the pointwise error belongs to 80-100% interval of the maximum absolute error.

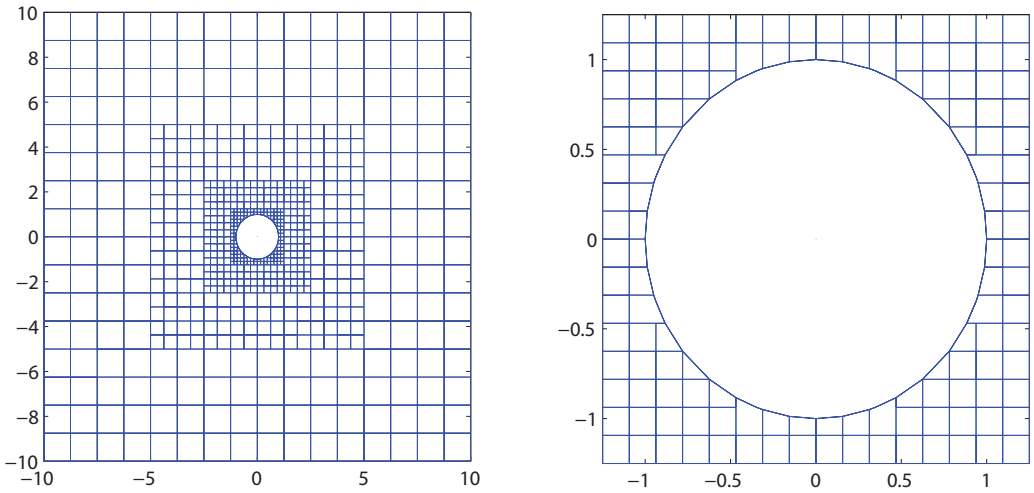

Figure 4.12: Left: Mesh around a cylinder containing 712 elements. Right: zoom of the same mesh near the surface.

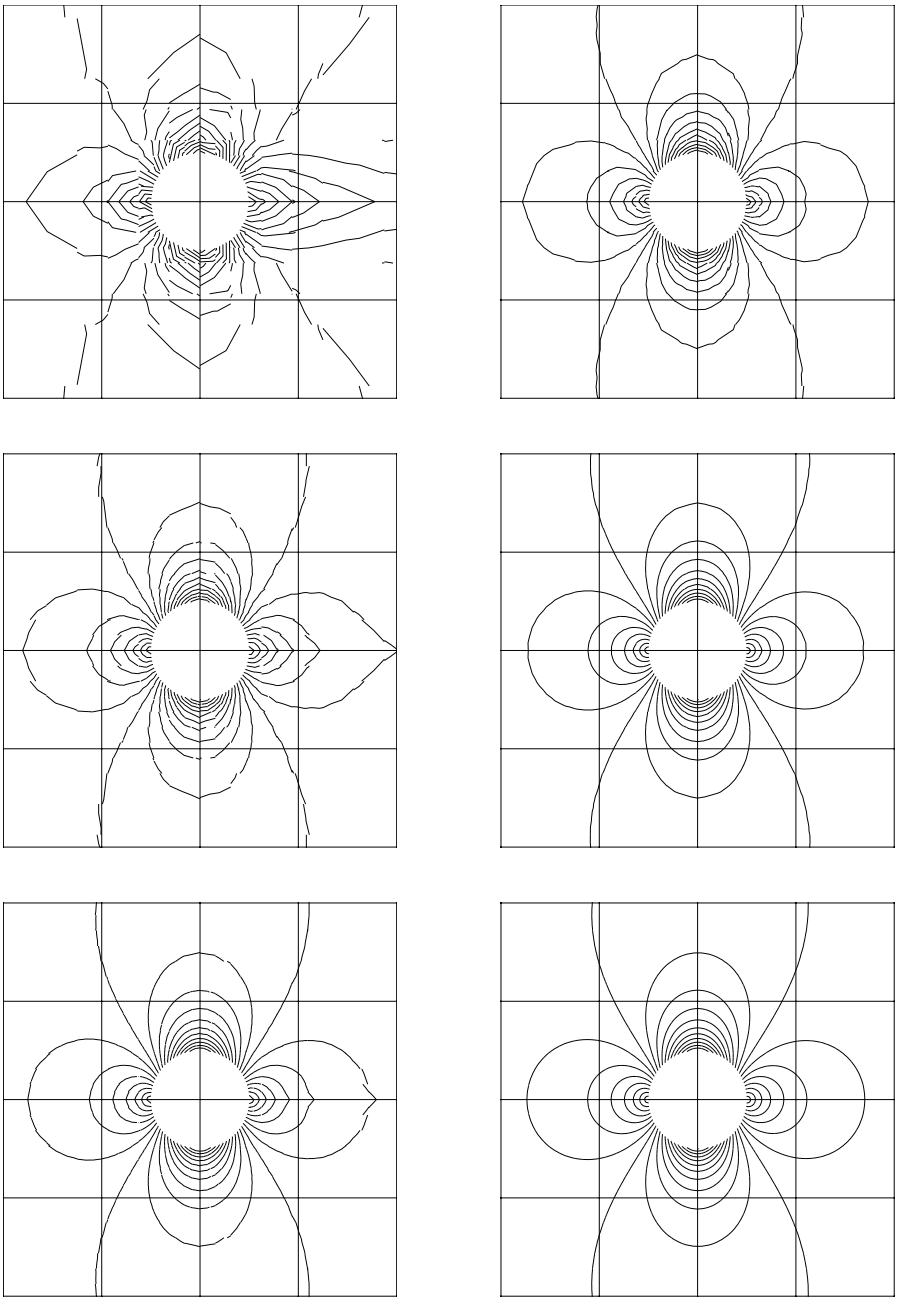

Figure 4.13: Isolines of the Mach number near the surface on the meshes with 176, 712, 2828 elements from top to bottom, and with $p = 1$ (left) and $p = 2$ (right).

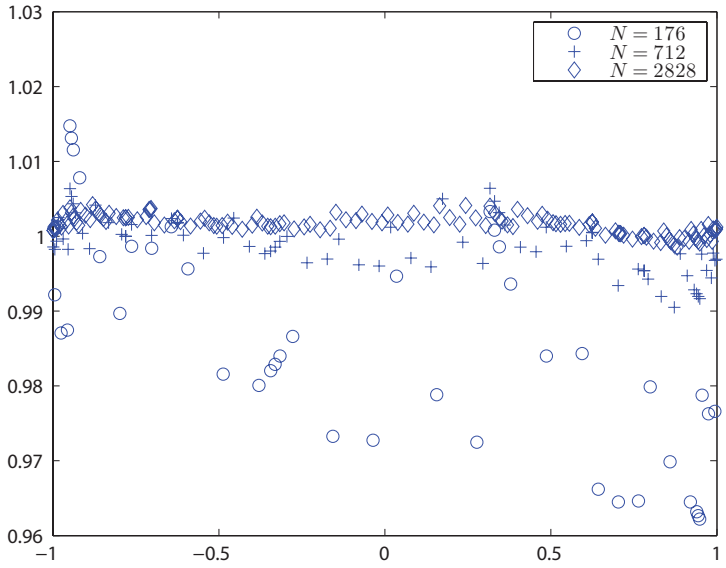

Figure 4.14: Total pressure loss coefficient on the surface of the cylinder, N is the number of elements and $p = 1$.

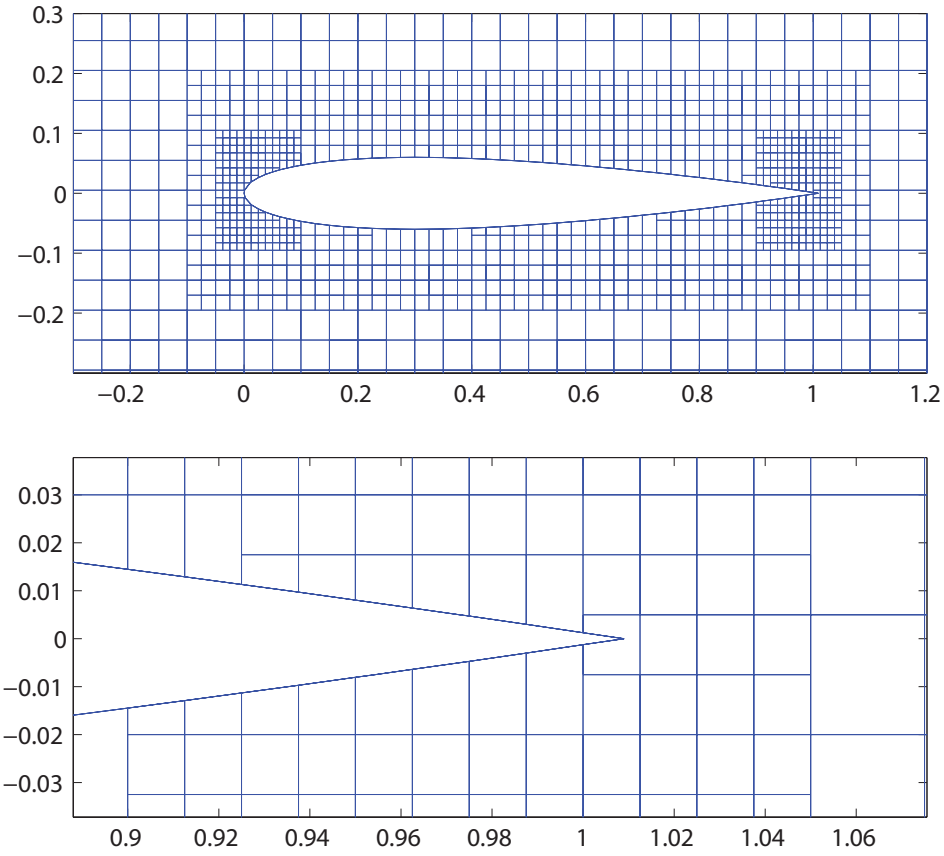

Figure 4.15: Top: part of the mesh around NACA0012 airfoil. Bottom: mesh near the tail.

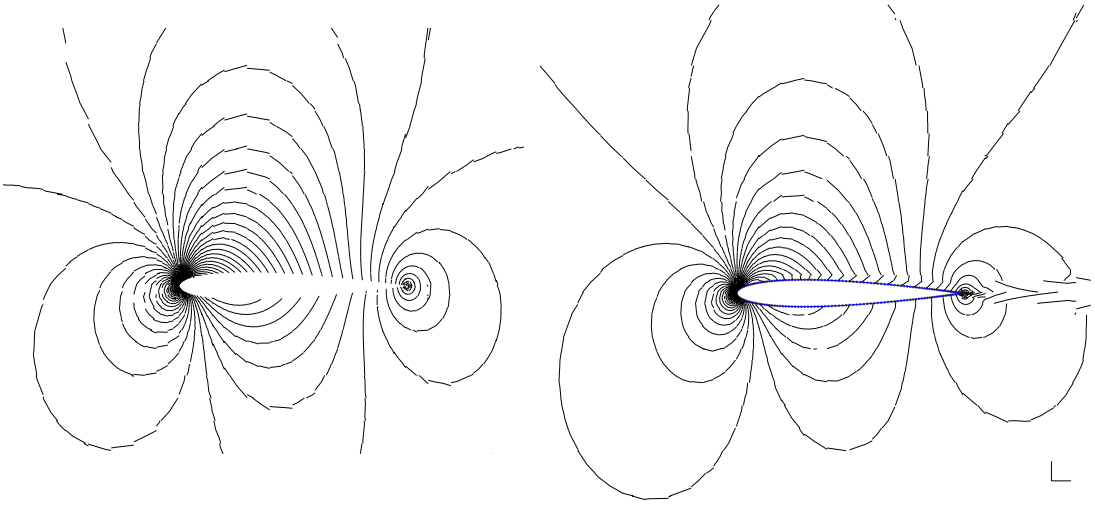

Figure 4.16: Isolines of the pressure (left) and Mach number (right) near the surface of the airfoil with $p = 1$.

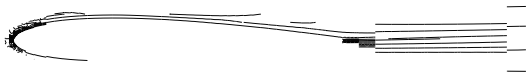

Figure 4.17: Isolines of the entropy near the surface of the airfoil.

Chapter 5

Conclusion and future work

In this thesis we have analyzed the spectrum of the DG spatial discretization for one-dimensional model problems on uniform and nonuniform meshes with the aim to understand the restriction that needs to be imposed on a time step to obtain a stable fully discretized scheme. We have also proposed a practical technique for implementation of the DGM on Cartesian grids with embedded geometries. Major contributions of this thesis are summarized below.

- We have derived a closed form expressions for the eigenvalues of the DG spatial discretization applied to the one-dimensional linear advection equation with periodic boundary conditions and the upwind flux. We have proven that the characteristic polynomial of the spatial discretization matrix L is related to the subdiagonal $[p/p+1]$ Padé approximant of e^{-z}.

- Based on the analytical equation for the eigenvalues, we have shown that $(p+1)(p+2)$ is a guaranteed bound on the size of the eigenvalues which can be used to compute the CFL condition for large p. However, we have also proven that the growth rate of the largest eigenvalue is less than $(p+1)^2$. We conjecture that a more accurate rate is proportional to $(p+1)^{1.75}$. This is in contrast with the currently assumed quadratic rate for the DGM [32] and various spectral methods [31].

- We have analyzed the spectrum of the DG spatial approximation on nonuniform grids for one-dimensional problems. We have shown that the stability restriction on nonuniform grids is global and depends on the composition of the mesh. We showed that when the size ratio of the largest to the smallest cell is less than the critical value, the spectrum of the discretization matrix enlarges roughly by a factor of the linear combination of cell sizes. When the refinement ratio is larger than the critical value and a very few small cells are

present, the spectrum might have eigenvalues that are located far from the main spectral curve but close to the multiples of singular points of the corresponding Padé approximants.

- We provide an estimate, which is easy to compute, on the upper bound of the spectrum on nonuniform grids. We also show that in certain cases the spatial discretization matrix can be highly nonnormal. In this case the pseudospectrum should be involved in determining the CFL number. This estimate can be used to compute the size of a stable time step that is less restrictive than the one defined by the size of the smallest cell. Numerical tests are presented to validate our estimate on relaxing the CFL number.

- We have applied a discontinuous Galerkin method to solutions of the Euler equations on Cartesian grids with embedded geometries. We have combined a number of techniques including integration on cut cells, cell merging and treatment of curved boundaries and demonstrated that the resulting method is accurate and quite viable. We obtain theoretical convergence rates and errors similar to those reported in the literature for a number of benchmark problems on unstructured two-dimensional meshes.

Many issues still remain to be addressed or investigated more deeply.

- A more accurate analytical estimate on the growth rate of the spectrum would be of interest. In particular, more accurate estimate on the distribution of eigenvalues for the DGM on general nonuniform meshes will be useful in determining a reliable and tight CFL restriction.

- An interesting direction of future work is to extend the analysis of the spectrum of the DGM to two-dimensional structured and unstructured meshes.

- Another application of the spectrum analysis is to modify the original DG method to obtain a method with a smaller spectrum. In [7], we show that the coefficients of the DG scheme can be manipulated to decrease the radius of the spectrum, i.e. to increase the CFL number, while preserving the convergence rate in the $\mathcal{L}^2$ norm. The improvement depends on the order of approximation. For example, we can have an improvement up to a factor of three for $p = 1$ and up to a factor of 5.5 for $p = 3$.

- For the DGM on Cartesian grids, the condition number of the mass matrix for different types of irregular cells and the choice of the basis functions needs to be looked at. Construction of higher order boundary elements needs to be incorporated, though this is mostly a question of availability of high-order meshes. Finally, while limiting for high-order methods is difficult, limiting on cut cells is a particularly challenging problem.

References

[1] M. Abramowitz and I.A. Stegun, editors. *Handbook of Mathematical Functions*. Dover, New York, 1965.

[2] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Adaptive Cartesian mesh generation. 1999.

[3] M. Ainsworth. Dispersive and dissipative behaviour of high order discontinuous Galerkin finite element methods. *Journal of Computational Physics*, 198:106–130, 2004.

[4] G. A. Baker and P. R. Graves-Morris. *Padé Approximants*. Addison-Wesley, Reading, Mass.; Don Mills, Ont., 1981.

[5] S. A. Bayyuk, K. G. Powell, and B. van Leer. A simulation technique for 2-D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry. *AIAA-93-3391-CP*, 1995.

[6] M. Berger, M. Aftosmis, and S. Murman. Analysis of slope limiters on irregular grids. Technical Report NASA Report NASA-05-007, 2005.

[7] N. Chalmers, L. Krivodonova, and R. Qin. Relaxing the CFL number of the discontinuous Galerkin method. Submitted to Journal of Computational Physics.

[8] G. Chavent and B. Cockburn. The local projection P^0P^1 discontinuous Galerkin method for scalar conservation laws. *RAIRO, Model. Math. Anal. Numer.*, 23:565, 1989.

[9] P. Ciarlet. *The Finite Element Method for elliptic problems*. SIAM, Philadelphia, 2002.

[10] B. Cockburn, S. Hou, and C.-W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for the conservation laws IV: The multidimensional case. *Mathematics of Computation*, 54:545–581, 1990.

[11] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin methods for scalar conservation laws II: General framework. *Mathematics of Computation*, 52:411–435, 1989.

[12] B. Cockburn and C.-W. Shu. The Runge-Kutte local projection P^1 discontinuous Galerkin method for scalar conservation laws. *RAIRO Model. Math. Anal. Numer.*, 25:337–361, 1991.

[13] B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems. *Journal of Computational Physics*, 141:199–224, 1998.

[14] B. Cockburn and C.-W. Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing*, 16:173–261, 2001.

[15] W. J. Coirier and K. G. Powell. An accuracy assesment of Cartesian mesh approaches for the Euler equations. *Journal of Computational Physics*, 117:121–131, 1995.

[16] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics*, 211:347–366, 2006.

[17] E. Constantinescu and A. Sandu. Multirate timestepping methods for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 33:239–278, 2007.

[18] A.J. Crossley and N.G. Wright. Time accurate local timestepping for the unsteady shallow water equations. *International Journal for Numerical Methods in Fluids*, 48:775–779, 2005.

[19] A. Dadone and B. Grossman. Ghost-cell method for analysis of inviscid three-dimensional flows on Cartesian grids. *Computer and Fluids*, 36:1513–1528, 2007.

[20] D. A. Dunavant. High degree efficient symmetrical gaussian quadrature rules for the triangle. *International journal for numerical methods in engineering*, 21:1129–1148, 1985.

[21] K. J. Fidkowski and D. L. Darmofal. A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 225:1653–1672, 2007.

[22] J. E. Flaherty. *Course Notes on Finite Element Analysis*.

[23] N. Gödel, S. Schomann, T. Warburton, and M. Clemens. GPU accelerated Adams-Bashforth multirate discontinuous Galerkin simulation of high frequency electromagnetic fields. *IEEE Transactions on magnetics*, 48(8):2735–2738, 2010.

[24] S. K. Godunov. A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations. *Math. Sbornik*, 47:271–306, 1969.

[25] S. Gottlieb, D. Ketcheson, and C.-W. Shu. High-order stability preserving time discretizations. *Journal of Scientific Computing*, 38(3):251, 2009.

[26] M. J. Grote, A. Schneebeli, and D. Schötzau. Discontinuous galerkin finite element method for the wave equation. *SIAM Journal on Numerical Analysis*, 44:2408–2431, 2006.

[27] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff problems*. Springer, Berlin, second edition, 2000.

[28] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and differential-algebraic problems*. Springer, Berlin, second edition, 2002.

[29] C. Halzel, M. J. Berger, and R. J. LeVeque. A high-resolution rotated grid method for conservation laws with embedded geometries. *SIAM Journal on Scientifical Computation*, 26-3:785–809, 2005.

[30] P. Henrici. *Applied and Computational Complex Analysis. Special functions-integral transforms-asymptotics-continued fractions*, volume 2. John Wiley & Sons, 1977.

[31] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for time dependent problems*. Cambridge University Press, Cambridge, 2007.

[32] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods. Algorithms, Analysis, and Applications*. Springer, 2007.

[33] F. Q. Hu and H. L. Atkins. Eigensolution analysis of the discontinuous Galerkin method with nonuniform grids. *Journal of Computational Physics*, 182:516–545, 2002.

[34] E. Isaacson and H. B. Keller. *Analysis of Numerical methods*. Dover, 1994.

[35] L. Krivodonova. Limiters for high-order discontinuous Galerkin methods. *Journal of Computational Physics*, 226:879–896, 2007.

[36] L. Krivodonova. An efficient local time-stepping scheme for solution of nonlinear conservation laws. *Journal of Computational Physics*, 229:8537–8551, 2010.

[37] L. Krivodonova and M. Berger. High-order accurate implementation of solid wall boundary conditions in curved geometries. *Journal of Computational Physics*, 221:492–512, 2006.

[38] L. Krivodonova and R. Qin. An analysis of the spectrum of the discontinuous Galerkin method. To appear in Applied Numerical Mathematics.

[39] L. Krivodonova and R. Qin. Linear stability analysis of the discontinuous Galerkin method on nonuniform grids. In preparation.

[40] E. J. Kubatko, C. Dawson, and J. J. Westerink. Time step restrictions for Runge-Kutta discontinuous Galerkin methods on triangular grids. *Journal of Computational Physics*, 227:9697–9710, 2008.

[41] R. J. LeVeque, editor. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, Cambridge, 2002.

[42] J. Modisette and D. Darmofal. Toward a robust, higher-order cut-cell method for viscous flows. *AIAA paper*, 2010–721, 2010.

[43] S.E. Mousavi, H. Xiao, and N. Sukumar. Generalized gaussian quadrature rules on arbitrary polygons. *International Journal of Numerical Methods in Engineering*, 82:1:99–113, 2010.

[44] S. M. Murman, M. J. Aftosmis, and M. J. Berger. Implicit approaches for moving boundaries in a 3-D Cartesian method. *AIAA papers 2003-1119*, 2003.

[45] R. Qin and L. Krivodonova. A discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries. *Journal of Computational Science*. Online available.

[46] J. J. Quirk. An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies. *Comput. Fluids*, 23:125–142, 1994.

[47] J. Remacle, J. E. Flaherty, and M. S. Shephard. An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems. *SIAM Review*, 45:53–72, 2003.

[48] P. Le Saint and P. Raviart. On a finite element method for solving the neutron transport equation. In C. de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 89–145, New York, 1974. Academic Press.

[49] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[50] S. Tan and C.-W. Shu. Inverse Lax-Wendroff procedure for numerical boundary conditions of conservation laws. *Journal of Computational Physics*, 229(21):8144–8266, 2010.

[51] E. Toro. *Riemann solvers and numerical methods for fluid dynamics*. Springer, 1999.

[52] L. N. Trefethen and M. Embree. *Spectra and pseudospectra - The behavior of nonnormal matrices and operators*. Princeton University Press, Princeton and Oxford, 2005.

[53] T.C. Warburton and G.E. Karniadakis. A discontinuous Galerkin method for the viscous MHD equations. *Journal of Computational Physics*, 152:608–641, 1999.