SPECTRAL ELEMENT DISCRETIZATION OF THE UNSTEADY NAVIER-STOKES EQUATIONS AND ITS ITERATIVE SOLUTION ON PARALLEL COMPUTERS

THÈSE N° 1380 (1995)

PRÉSENTÉE AU DÉPARTEMENT DE GÉNIE MÉCANIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Wouter COUZY

Licencié en mathématiques de l'Université d'Amsterdam, Pays-Bas
de nationalité hollandaise

acceptée sur proposition du jury:

Prof. M. Deville, rapporteur
Prof. J. Descloux, corapporteur
Prof. P. Fischer, corapporteur
Prof. R. Keunings, corapporteur
Prof. E. Mund, corapporteur
Prof. I. Ryhming, corapporteur

Lausanne, EPFL
1995

Abstract

In this thesis we will show that unsteady, incompressible fluid-flow problems, as described by the three-dimensional Navier-Stokes equations, can efficiently be simulated by a parallel computer code based on a spectral element discretization. This efficiency relies a great deal on the numerous improvements to spectral methods on the theoretical, algorithmic and computational levels which have been proposed over the last decade. Consequently, the application of spectral methods is no longer limited to regular problems in simple geometries, but is extended to complex situations. The spectral element method in particular has received much attention, because it combines the practical advantages of the well-established finite element method with the "spectral" ability to reduce the number of degrees of freedom to obtain a prescribed level of accuracy. Furthermore, it embodies the concept of domain decomposition which is closely related to parallelism. The latter aspect is important for an effective implementation on the leading ("affordable") supercomputers of the nineties, which are almost exclusively based on parallel processors with distributed memory.

The continuous Navier-Stokes equations describe the velocity and pressure of a fluid in a domain and the spectral element method reduces them to a set of discrete equations. This discretization technique has become rather mature and furnishes the framework for this thesis. The main challenge, however, lies in the development of fast algorithms for the solution of the discrete systems. To this end, we will analyze existing and new methods of decoupling the velocity components from the pressure. Iterative methods have shown their merits for the solution of the resulting, decoupled set of equations, provided that they are properly preconditioned. This condition has led to the introduction of a large number of preconditioning methods for velocity and pressure operators. The common factor of these techniques is that they are based on fast, local solves, combined with a strategy to deal with the element interfaces.

Another important issue is the time discretization of the unsteady equations. It is common practice to choose an implicit time-integration scheme for the linear terms, and an explicit one for the nonlinear terms. The way in which these two schemes are combined is not trivial, especially when two important conditions, high-order accuracy in time and stability, have to be satisfied simultaneously. In fact, the concepts of precision and stability are related: the more stable a time scheme, the larger the maximum time step that can be used. From a computational viewpoint, it is desirable to advance in time with the maximum allowed time step. An acceptable level of accuracy in time is then only guaranteed by high-order time schemes.

The construction of a high-performance code requires not only readily parallelizable numerical solution methods, but also fast algorithms to manage the (little) communication involved. The use of iterative methods for the solution of the discrete systems enhances the parallel implementation, especially when the preconditioners are block diagonal (communication free). The communication algorithms are analyzed and optimized, and the use of very fast, architecture-specific routines has to be balanced against the portability of the code. All the algorithmic, numerical, computational, and parallel improvements are first tested individually on small problems. The parallel spectral element code is based on the outcome of these tests and is used for the simulation of the three-dimensional flow over a backward-facing step.

Résumé

Cette thèse se donne comme ambition de montrer que la simulation numérique des écoulements de fluides instationnaires et incompressibles, décrits par les équations tridimensionnelles de Navier-Stokes, peut être traitée de manière efficace par un programme de calcul basé sur une discrétisation spatiale de type spectral. Cette efficacité repose largement sur de nombreuses améliorations apportées dans le domaine des méthodes spectrales ces dix dernières années tant au niveau de la théorie que des algorithmes et des coûts de calcul. Par conséquent, le champ d'applications des méthodes spectrales ne se limite plus aux géométries simples, mais s'étend aux situations complexes. En particulier, la méthode des éléments spectraux est devenue très populaire parce qu'elle combine les avantages de la méthode des éléments finis avec la "capacité spectrale" de réduire le nombre de degrés de liberté pour atteindre une précision fixée. En outre, elle incorpore naturellement l'idée de décomposition en sous-domaines, notion intimement liée au parallélisme et donc à la réduction des coûts de calcul. Ceci est la clé pour une implémentation efficace sur les ordinateurs les plus puissants de notre époque constitués presque sans exception de processeurs parallèles à mémoire répartie.

Les équations continues de Navier-Stokes décrivent l'évolution de la vitesse et de la pression d'un écoulement de fluide dans un domaine matériel; la méthode des éléments spectraux les réduit à un ensemble d'équations discrètes. Cette technique de discrétisation est fiable et bien établie, et constitue le point de départ de ce travail. Cependant, le développement d'algorithmes rapides pour la résolution du système discret est un défi majeur. A cette fin, on étudiera le découplage des composantes de vitesse et de pression en comparant des algorithmes existants et originaux. Les méthodes itératives se sont avérées très appropriées afin de mener à bien la résolution des systèmes discrets découplés. Toutefois, des techniques adéquates de préconditionnement ont été developpées afin d'assurer un taux de convergence acceptable des méthodes itératives. Pour ce faire, un nombre de préconditionneurs des opérateurs de vitesse et de pression ont été proposés et validés. Ces techniques ont toutes en commun des solutions rapides et locales à un élément et une stratégie de traitement d'interfaces.

Un autre sujet important abordé dans ce travail est la discrétisation temporelle des équations instationnaires. La manière classique est de choisir un schéma implicite pour les termes linéaires et un schéma explicite pour les termes convectifs. La mise au point de telles méthodes qui doivent être à la fois stables et d'une précision d'ordre élevé, n'est pas évidente. En fait, les concepts de stabilité et de précision sont liés: plus le schéma temporel est stable, plus le pas de temps admissible est grand. Du point de vue de la rapidité de calcul, il est souhaitable que l'évolution dans le temps se fasse avec un pas maximal. Un niveau de précision acceptable est alors obtenu uniquement par une méthode d'ordre élevé.

Le développement d'un programme de calcul de haute performance demande non seulement des méthodes numériques parallélisables, mais aussi des algorithmes rapides qui génèrent une communication entre processeurs la plus réduite possible. L'implémentation parallèle est facilitée par l'utilisation de méthodes itératives, notamment en présence de préconditionneurs diagonaux par bloc. On montrera à travers l'analyse et l'optimisation des algorithmes de communication, que les plus performants sont souvent exclusifs pour certains types d'architecture de machine. Cette spécificité doit être mise en regard avec la perte de portabilité du programme de calcul.

Toutes les améliorations au niveau des algorithmes, de la parallélisation et de l'efficacité de calcul sont appliquées dans un premier temps à des problèmes académiques en vue de tests et d'optimisations. Ensuite, le programme de calcul final a été utilisé avec succès pour la simulation d'un écoulement tridimensionnel sur une marche descendante.

Acknowledgements

Although the name of only one author appears on the cover of this thesis, a number of people deserve to be mentioned for their respective contributions:

First of all I would like to thank my promotor, Professor Deville (EPFL). His continuous interest in my work and his personal, pleasant approach provided me the right working conditions.

I am very grateful to Professor Keunings (UCL) for taking over the administrative burden during the last two years of my contract.

The present work has been made possible thanks to the financial support of the Belgian State, Prime Minister's Office, Science Policy Programming in the framework of the incentive programme "Information Technology - Computer Science of the Future".

Research is teamwork, and therefore it was a real pleasure to join "l'équipe spectrale". The (ex-)members of this group did not only answer patiently my endless questions, but became also very good friends.

During my four years at MEMA (UCL), I have constantly benefited from the scientific know-how and the kind atmosphere: It really was a nice time. I also had the privilege to visit the IMHEF (EPFL) on several occasions. The welcome was always friendly and the cooperation outstanding.

Cheers to all the old pals I could always rely on.

Finally I wish to thank my parents. Their warmth and support were always present and meant a lot to me.

Contents

Chapter 1

Introduction

1.1 Numerical simulation of incompressible fluid flows

The complex problem of the numerical simulation of incompressible fluid flows has been studied by several generations of scientists all over the world. The number of combinations of applications, models, discretization techniques in time and space, solution methods for the algebraic equations, etc. seems to be infinite. Consequently, there is progress, but at small pace.

Over the last decades a new challenge was born by the rapidly increasing performance of computational resources. Nowadays, advance is not only to be made on the aforementioned domains, but the methodology should also be tuned to modern computer architectures. This means that, ideally, superlinear progress can be made with respect to the capacity of computers. In practice, however, it is not even trivial to advance at the *same* pace as the supercomputers, the performance of which is estimated to increase by a factor of ten to fifty each decade. Besides the important improvement of the single-processor hardware, the latest (and probably also the next) step forward has been achieved by connecting more and more independently working computing nodes by fast communication networks. More precisely, over the last ten years a new generation of parallel distributed memory machines has been developed. The performance of these computers has overtaken that of the fastest single-processor machines. In the next years, parallel technology will further improve by putting together more chips and by using faster communication hardware. Furthermore, the individual chips that constitute the parallel computer will be able to perform more operations per time unit. The increasing interest in parallel computing is not only driven by the need for ever faster machines. Economical considerations are also very important, considering that the rate of Mflops (millions of floating-point operations per second) per dollar is much higher for parallel than for single-processor supercomputers.

Not every discretization technique or solver is equally suited for these new parallel architectures and it is the task of the scientist to pick out the best candidates and to modify and optimize them where necessary. A method is optimal in the sense of

parallel computing when it is scalable. This means that the solution of a problem of size S is obtained on P processors at the same speed as a similar problem of size $2S$ on $2P$ processors. Another vital, but often neglected aspect in the "battle of the Flops" is the way the program is implemented on a computer. It happens often that a close inspection of only a small percentage of the total amount of (Fortran) lines leads to a speed-up that computer constructors can only dream of. A good understanding of phenomena like scalar optimization, memory hierarchies and efficient inter-processor communication techniques has become as important as a good theoretical insight.

The goal of this thesis is to describe the development of an efficient solver for the incompressible, unsteady Navier-Stokes equations based on (potentially) accurate discretizations in time and space. Such a solver could be used to simulate complex three-dimensional flow phenomena which find their origin in hydraulics, aerodynamics, haemodynamics, etc. Our choice for the spectral element discretization method (first proposed by Patera [57]) is not only guided by the know-how on spectral methods which is available at our group and by its applicability to the class of problems that we want to study: (Spectral element methods are very well suited for problems in which high regularity is guaranteed or for a class of problems in which high regularity is not the exception, like incompressible fluid mechanics [47].) Our choice is also based on the fact that the spectral element method (SEM) in combination with iterative solvers is readily parallelizable, as was shown in a number of articles by Fischer and coworkers (see e.g. [26] and [28]). As a matter of fact, the SEM has already been described in a large series of papers (see e.g. [57], [65], [47], [46], [26], [28] and [74]) and is nothing more than a tool for the present work. The emphasis of this thesis will be rather on the vast domain in between discretization and actual simulation. This involves the analysis of high-order time schemes with improvement of the stability properties and the construction of efficient iterative solvers for the discrete equations. We constantly have to keep in mind that the developed algorithms should be applicable to parallel, distributed-memory computers.

1.2 Outline

As stated in the previous paragraph, we focus on high-order discretization methods in time and space. This allows either to look for very accurate solutions, or, if precision is less important or unrealistic to obtain, to reduce the number of grid points/time steps in comparison with more classical discretization methods to obtain the same level of accuracy. As far as the spatial discretization is concerned, spectral methods feature exponential convergence with respect to the degree of the polynomial expansions, provided that the solution is smooth enough. This explains that spectral methods are especially suited for problems in which high regularity is common. Another argument to use this type of methods is that numerical dissipation and dispersion errors are almost absent. For this reason, classical discretization techniques, like the finite volume and finite element method, are sometimes combined with a spectral computation of the non-linear term. As an example of such an application, we mention the simulation of turbulence, e.g. by Schumann [70]. The fact that the spectral grid points are clustered at the boundaries can be considered as another advantage when trying to solve flows

that are dominated by boundary-layer dynamics. The SEM is based on the decomposition of the domain in a number, say K, of nonoverlapping subdomains (spectral elements). On each of these elements the solution is expanded in tensor-product based polynomials of high degree, say N, with typically $4 \leq N \leq 15$. This decomposition allows for local refinements at places where the solution is (expected to be) rapidly changing. The variational formulation provides automatically the continuity of the solution at the interfaces between adjacent elements, and deformed geometries can be handled without difficulties. All these factors make the SEM highly flexible as regards geometry, accuracy, and parallelization.

Once the possibility to obtain a high spatial precision has been established, we do not want these accurate results to be contaminated with large time errors. Although most of the simulations in this thesis concern steady flows, it is also important to study transient phenomena that occur at higher Reynolds numbers. As an example, we mention the confined flow around a cylinder. For the direct simulation of turbulent flows, which could be a future goal, high-order (i.e. up to an order of three or four) accuracy in time is important to perform statistics. Here, however, our main concern is to maintain a reasonable precision when proceeding with the largest time step that is allowed without violating certain stability conditions, due to the explicit treatment of the nonlinear term.

Because of considerations based on the condition number of the operators (see Chapter 2) and on the fact that we want to avoid indefinite, nonsymmetric matrices, all terms of the Navier-Stokes equations are discretized in time by an implicit scheme, except for the nonlinear convection term. This implicit/explicit splitting can be performed in several ways and should not degenerate the order of the time scheme. In Chapter 3, two splitting methods are compared with respect to stability and accuracy.

The final set of discrete equations is solved by an iterative method, which is preferred to direct methods for a number of reasons. We should start by remarking that matrix-vector multiplications (we will call also them "evaluations") are performed by tensor-product based multiplications. This reduces the operation count of an evaluation from $O(KN^6)$ to $O(KN^4)$, but has as a side-effect that matrices are never built up. There are several consequences attached to this remark which motivate our choice for an iterative solver: First, matrix evaluations, which are the basic operations of every iterative solver, are performed at a very low cost. Moreover, these operations can be written in BLAS, a basic linear algebra package which is optimized on almost every supercomputer, and requires very little memory, avoiding time-consuming swapping. Second, sparsity patterns are difficult to localize which implies that, in the case of a direct solver, the whole matrix and its inverse would have to be kept in memory. Furthermore, the efficiency of the direct solvers that are used in computational fluid dynamics rely on the existence of such sparsity patterns.

A priori, the discrete set of equations is not suited to be solved by an iterative method. The absence of the pressure in the continuity equation causes a zero block, yielding a very slow convergence, if there is any convergence at all. Therefore, it is a good idea to decouple the velocity field from the pressure, leading to four independent (semi-)positive definite, symmetric systems of equations; one for each of the three velocity components and one for the pressure. This decoupling can either be performed

on the continuous equations (see e.g. Karniadakis et al. [42] and Timmermans [74]) or on the discrete equations (see e.g. Blair Perot [9]). A disadvantage of the former approach is that an additional boundary equation for the pressure is introduced, which is not obvious from the theoretical point of view (see e.g. Orszag et al. [56]). On the other hand, this additional equation is held responsible for the filtering of the spurious pressure modes, allowing for a discretization of the velocities and the pressure on the same (Gauss-Lobatto-Legendre) grid. We choose for a decoupling applied to the discrete equations, according to [9], in which all the boundary conditions are already incorporated. The spurious pressure modes are avoided by using a staggered grid for the pressure. Like for the explicit/implicit splitting, an extra time-error is introduced by the decoupling procedure, which is analyzed in Chapter 5. Following the theory of Blair Perot [9], this is readily done by considering the decoupling method as a generalized block decomposition. A series of already existing methods (like fractional step, Uzawa) is studied and some new, high-order projection schemes are derived. It turns out that it is often advantageous to compute a correction to the pressure. Some recommendations on which method to use in what situation (steady/unsteady problem, large/small time step, etc.) are given.

A fast inversion method for the Helmholtz equations is important, especially when the Uzawa decoupling method is used. In Chapter 4, we will discuss a direct method based on the fast diagonalization method (FDM, see [44]). The FDM evaluates the inverse of a tensorizable, separable operator at the price of two conjugate gradient iterations. Unfortunately, this technique applies only to very simple geometries, i.e. a box, decomposed in nondeformed parallelepipedic spectral elements. Still, it can be useful for the simulation of cavity or square-duct flows. When we allow more complicated (but still nondeformed) geometries, a Schur complement method is used to decouple the interior from the interface variables. The FDM is then applied to the interior nodes and the interface system can be solved either by a direct method or a preconditioned conjugate gradient method (PCGM). Finally, in the case of generally deformed geometries, a preconditioning technique is proposed based on, again, fast diagonalization techniques at the interior nodes and the incomplete, block-diagonal Schur operator for the interface variables.

Although the condition number of the pressure operator depends on a number of factors like the Reynolds number, the time step, and the decoupling method, we can in general say that the pressure problem is difficult to solve. Therefore, preconditioning is essential. A good preconditioner should not only be "spectrally close" to the original operator in order to reduce the number of iterations, but it should also be cheap to invert. In Chapter 6, this subject is studied extensively. Most of the ideas are based on a paper by Rønquist [63], who proposed to compute the pressure in two pressure subspaces, a coarse, global and a fine, elemental one. The pressure levels on the coarse grid (the "skeleton") are computed by a direct method and the resulting pressure on the fine grids is solved by a PCGM. Numerical tests have shown that this method leads to a condition number that is independent of the number of spectral elements. The main difference between the preconditioning methods developed in Chapter 6 and those of the paper of Rønquist [63] is the way in which the elemental preconditioner is evaluated. Moreover, many pressure operators have to be considered corresponding to the different decoupling methods that have been proposed in Chapter 4, e.g.

high-order projection, pressure correction and Uzawa methods. An original two-stage preconditioning technique is developed for the latter operator. Its primary goal is to reduce the number of expensive iterations and, hence, cpu time, rather than to reduce the number of global iterations.

Following the discussion in Section 1.1, the Navier-Stokes solver based on the methodology of the Chapters 2-6 has been implemented on parallel computers. As our target machine we chose the Cray T3D at the EPF in Lausanne, Switzerland, consisting of 256 DEC Alpha chips, interconnected by a fast communication network. The memory is distributed over the processors and communication paradigms (or, in Cray terminology, "programming models") have to be used to transfer information from one processor to another. Among the three available models, we have chosen PVM (Parallel Virtual Machine), since it was available from the beginning. Some rules of the thumb for parallel programming on the T3D are presented and a modified parallel version of the PCGM (see Meurant [53]) is investigated. It turns out that a good parallel efficiency is obtained and that the single-processor performance is also relatively good. This opens the way to large-scale simulations of realistic, three-dimensional flows. We have also studied the efficiency of the code on the Intel Paragon at the ETH in Zürich, Switzerland, using both PVM and NX, the Intel communication library. Finally, a small test using the work-sharing programming model on the T3D is presented.

In Chapter 8, some larger problems are simulated. As a first test case, we investigate the overall order of the time-integration scheme. Then, we proceed with the simulation of a three-dimensional backward-facing-step flow, for which clear experimental data is available (see Armaly et al. [1]) for a wide range of Reynolds numbers. Furthermore, the two-dimensional case has been extensively studied by other authors, providing much material for comparison. This parallel simulation consists of 128 spectral elements with a degree up to 11 in each spatial direction, which can not be done on any serial machine on the market.

Finally, some conclusions are drawn and suggestions for future work are indicated.

Chapter 2

Spectral discretization method

In this chapter, we will first introduce some basic concepts to help us produce mathematical expressions that reflect the quality (to be defined later) of spectral solutions for a model problem. Then, a particular discretization technique, the spectral element method, is discussed in detail. We will comment both on its theoretical and implementational aspects. The extension to the Stokes problem is presented, where special care has to be taken to avoid spurious pressure modes. Finally, the discretization of the nonlinear term is introduced, allowing for full Navier-Stokes computations

For a more detailed discussion on the interest of spectral discretization techniques in the field of computational fluid dynamics, the reader is referred to Chapter 1. For the preparation of this chapter the textbooks and articles of the following authors were very useful: Bernardi and Maday [6], Canuto et al. [14], Maday and Patera [47], Maday et al. [46], [49], Patera [57] and Rønquist [65]. In these papers and books, many of the proofs of the presented theorems and error estimates can be found.

2.1 Basic concepts

This section is devoted to some basic concepts that play an important role in spectral discretizations. A simple problem will be discretized by a Legendre-Galerkin method and error estimates will be presented. For sake of simplicity, we introduce a linear model problem A in an open, one-dimensional domain $(-1,1)$, with solution $u \in X$, and f the data of the problem; X a certain set of admissible functions in one variable:

$$Au(x) = f(x) \qquad x \in (-1,1) \tag{2.1}$$

$$u(x) = 0 \qquad \text{for } x = \pm 1. \tag{2.2}$$

Instead of solving the continuous problem, an approximate solution u_N to u will be sought in a finite-dimensional subspace X_N (N a positive integer) of X. Typically, the subspace X_N is defined by all the polynomials in one variable of degree smaller than or equal to N satisfying the boundary conditions (2.2).

The numerical simulations will supply us with a solution $u_N \in X_N$ that is determined

by its nodal values $\{\hat{u}_i\}_{i=0}^N$ in a finite set of points $\{x_i\}_{i=0}^N$ on the interval $[-1,1]$

$$u_N(x) = \sum_{i=0}^{N} \hat{u}_i h_i(x), \tag{2.3}$$

with h_i the Lagrangian interpolation polynomial of degree N which takes the unit value at x_i and is zero in x_j $(j \neq i)$. Of course, it is interesting to study the difference between the solution u and its approximation u_N. Roughly speaking, this error consists of three components: First, there is an "approximation" error due to neglecting the terms $\hat{u}_i h_i$ for $i > N$ in (2.3). This approximation error is also called "truncation" or "projection" error. There is also an "aliasing" error due to interpolation and an integration error, causing that $\hat{u}_i \approx u(x_i)$ instead of $\hat{u}_i = u(x_i)$. As we will see in this chapter, the candidate u_N supplied by the spectral method is optimal. This can be shown by using concepts as "best approximations" and "accurate quadratures". A priori, terms like "optimal", "best" and "accurate" are subjective, so we require measures to quantify the quality of the discrete solution. To this end, we introduce two Hilbert spaces, $\mathcal{L}_w^2$ and $\mathcal{H}_w^m$, on the interval $(-1,1)$. First, $\mathcal{L}_w^2(-1,1)$ is defined as the space of measurable functions on $(-1,1)$ that are square integrable on the same interval with respect to a strictly positive, integrable weight function w:

$$\mathcal{L}_w^2(-1,1) = \{v \text{ measurable}; \int_{(-1,1)} v^2(x) w(x) dx < \infty\}, \tag{2.4}$$

with inner product

$$(v_1, v_2)_{\mathcal{L}_w^2(-1,1)} = \int_{(-1,1)} v_1(x) v_2(x) w(x) dx \tag{2.5}$$

and norm

$$\| v \|_{\mathcal{L}_w^2(-1,1)} = \sqrt{(v,v)_{\mathcal{L}_w^2(-1,1)}}. \tag{2.6}$$

For any positive integer m, we define the Sobolev space $\mathcal{H}_w^m(-1,1)$ as

$$\mathcal{H}_w^m(-1,1) = \{v \in \mathcal{L}_w^2(-1,1); \forall \alpha \text{ integer}, 1 \leq \alpha \leq m, \frac{\partial^\alpha v}{\partial x^\alpha} \in \mathcal{L}_w^2(-1,1)\}. \tag{2.7}$$

The Sobolev space is equipped with inner product and norm

$$(v_1, v_2)_{\mathcal{H}_w^m(-1,1)} = \int_{(-1,1)} \sum_{|\alpha| \leq m} \left\{ \frac{\partial^\alpha v_1}{\partial x^\alpha}(x) \frac{\partial^\alpha v_2}{\partial x^\alpha}(x) \right\} w(x) dx \tag{2.8}$$

$$\| v \|_{\mathcal{H}^m_w(-1,1)} = \sqrt{(v,v)_{\mathcal{H}^m_w(-1,1)}}. \tag{2.9}$$

For future use, we also need the space $\mathcal{P}_N(-1,1)$, given by

$$\mathcal{P}_N(-1,1) = \{v(x), x \in (-1,1); v \text{ a polynomial of degree } \leq N\}. \tag{2.10}$$

Let us denote by $\{\phi_i\}_{i=0}^{\infty}$ the set of orthogonal basis functions for X (from now on, $X = \mathcal{L}^2_w(-1,1)$) such that the subset $\{\phi_i\}_{i=0}^{N}$ spans the subspace $\mathcal{P}_N(-1,1)$. Every function $v \in \mathcal{L}^2_w(-1,1)$ can be written as

$$v = \sum_{i=0}^{\infty} \tilde{v}_i \phi_i, \text{ with } \quad \tilde{v}_i = \frac{(v,\phi_i)_{\mathcal{L}^2_w(-1,1)}}{(\phi_i,\phi_i)_{\mathcal{L}^2_w(-1,1)}}. \tag{2.11}$$

The truncated series $P_N v \in \mathcal{P}_N(-1,1)$ is given by

$$P_N v = \sum_{i=0}^{N} \tilde{v}_i \phi_i. \tag{2.12}$$

This truncation can be interpreted as an orthogonal projection operator $P_N : \mathcal{L}^2_w(-1,1) \rightarrow \mathcal{P}_N(-1,1)$ that satisfies for every function $v \in \mathcal{L}^2_w(-1,1)$

$$\forall \psi_N \in \mathcal{P}_N(-1,1); \quad (v - P_N v, \psi_N)_{\mathcal{L}^2_w(-1,1)} = 0. \tag{2.13}$$

Note that (2.13) is not necessarily true with respect to the $\mathcal{H}^m_w$ inner product. The set $\{\tilde{v}_i\}_{i=0}^{\infty}$ is often referred to as the *spectrum* of the function v.

Spectral convergence is obtained for the truncation error with respect to the $\mathcal{L}^2_w(-1,1)$ norm: For every $v \in \mathcal{H}^m_w(-1,1)$

$$\| v - P_N v \|_{\mathcal{L}^2_w(-1,1)} \leq C N^{-m} \| v \|_{\mathcal{H}^m_w(-1,1)}. \tag{2.14}$$

In other words, the truncated series is the *best approximation* of v in $\mathcal{L}^2_w(-1,1)$ norm, that is, the power of $1/N$ is equal to the difference of the order of the Sobolev spaces between the left- and right-hand sides of the equation.

The following estimate expresses the truncation error of the derivative. For every function $v \in \mathcal{H}^m_w(-1,1)$

$$\| v - P_N v \|_{\mathcal{H}^n_w(-1,1)} \leq C N^{-\frac{1}{2}} N^{2n-m} \| v \|_{\mathcal{H}^m_w(-1,1)}, \tag{2.15}$$

with $1 \leq n \leq m$. By taking $n = m = 1$ in Equation (2.15) it can be seen that this approximation is no longer optimal, which is explained by the fact that truncation is not the same as orthogonal projection in $\mathcal{H}^m_w(-1,1)$ space.

Exponential decay of the spectrum is guaranteed if the orthogonal basis functions $\{\phi_i\}_{i=0}^{\infty}$ are the eigenfunctions of the singular Sturm-Liouville problem (see e.g. [14]). Two families of polynomials are of particular interest: Legendre and Chebyshev polynomials. In this text, we will restrict ourselves to the former family, which is orthogonal with respect to the weight function $w(x) = 1$. According to this particular choice, we will omit the index w henceforth from definitions (2.4) and (2.7). The $ith-$order Legendre polynomial L_i is given by the following recursion formula:

$$L_{i+1}(x) = \frac{2i+1}{i+1} x L_i(x) - \frac{i}{i+1} L_{i-1}(x), \tag{2.16}$$

with $L_0(x) = 1$ and $L_1(x) = x$.

Sometimes it is also helpful to analyze the spectrum of an approximation u_N. This can be done by discrete transforms. Some spectral methods use these transforms, which should be very fast, to compute derivatives in the spectral space, rather than in the physical space. For methods based on Legendre polynomials such a fast discrete transform does not exist, but we will show that this is not a drawback of the Legendre spectral element method, since all computations can be done efficiently in the physical space.

Let us go back to the model problem (2.1), (2.2). The first step towards a numerical solution consists in mapping $A_N u_N - f$ by an orthogonal projection on a proper subspace $Y_N \subset X$ (A_N denotes the discrete operator and $u_N \in X_N$, again, the discrete solution), yielding the variational formulation: Find $u_N \in X_N$ such that

$$(A_N u_N - f, v_N)_N = 0 \qquad \forall v_N \in Y_N, \tag{2.17}$$

with $(,)_N$ a bilinear form which is an inner product on Y_N. The choices for $(,)_N$ and Y_N determine the method, i.e. Tau, Galerkin or collocation. (Note that for collocation approximations the projection is defined from $Z \rightarrow Y_N$, with $Z \subset X$, see [14].) Formulation (2.17) expresses that every spectral scheme is actually a method of weighted residuals. Here, we take $(,)_N = (,)_{\mathcal{L}^2(-1,1)}$ and $Y_N = X_N$, yielding the Galerkin formulation: Find $u_N \in X_N$ such that

$$(A_N u_N, v_N)_{\mathcal{L}^2(-1,1)} = (f, v_N)_{\mathcal{L}^2(-1,1)} \qquad \forall v_N \in X_N. \tag{2.18}$$

This formulation, however, has only a theoretical interest, since, the exact evaluation of the integrals in (2.18) is very expensive (if not impossible). Therefore, we resort to numerical integration rules to compute these integrals. This procedure will introduce an additional error, and we require that the order of magnitude of this "integration" error is the same as the truncation error (also called projection or approximation error). This suggests Gauss-Lobatto quadrature rules associated with Legendre polynomials. To this end, we define the set of Gauss-Lobatto-Legendre points $\{\xi_i\}_{i=0}^{N}$ with associated weights $\{\rho_i\}_{i=0}^{N}$. We have

$$-1 = \xi_0 < \xi_1 < ... < \xi_N = 1, \qquad \frac{dL_N}{dx}(\xi_i) = 0, \quad i = 1, .., N-1, \tag{2.19}$$

and

$$\rho_i = \frac{2}{N(N+1)} \frac{1}{L_N^2(\xi_i)}, \quad i = 0, .., N. \tag{2.20}$$

The GLL intergration rule is exact for polynomials of degree smaller than or equal to $2N-1$:

$$\forall v \in \mathcal{P}_{2N-1}(-1,1); \int_{(-1,1)} v(x)dx = \sum_{i=0}^{N} v(\xi_i)\rho_i. \tag{2.21}$$

We introduce the discrete inner product $(,)_{GL}$ corresponding to GLL numerical integration

$$(v_1, v_2)_{GL} = \sum_{i=0}^{N} v_1(\xi_i)v_2(\xi_i)\rho_i \quad \forall v_1, v_2 \in \mathcal{L}^2(-1,1). \tag{2.22}$$

In order to get an estimate for the error committed by replacing the continuous inner product (2.5) by the discrete one (2.22), it is of help to introduce the interpolation operator $I_N : \mathcal{L}^2(-1,1) \rightarrow \mathcal{P}_N(-1,1)$ based on the Gauss-Lobatto-Legendre points. For every function $v \in \mathcal{L}^2(-1,1)$, the interpolation operator can be defined with respect to Legendre or Lagrangian polynomials:

$$I_N v(x) = \sum_{i=0}^{N} \frac{(v, L_i)_{GL}}{(L_i, L_i)_{GL}} L_i(x) = \sum_{i=0}^{N} v(\xi_i)h_i(x), \tag{2.23}$$

yielding $I_N v(\xi_i) = v(\xi_i)$, $i = 0, .., N$. The interpolant $I_N v$ is the projection of v on $\mathcal{P}_N$ with respect to the discrete inner product (2.22). From

$$(v_1 - I_N v_1, v_2)_{GL} = 0 \quad \forall v_1, v_2 \in \mathcal{L}^2(-1,1), \tag{2.24}$$

and

$$\begin{aligned} |(v_1, v_2) - (v_1, v_2)_{GL}| \;\leq\; & C \Big(\| v_1 - P_{N-1} v_1 \|_{\mathcal{L}^2(-1,1)} \\ & + \| v_1 - I_N v_1 \|_{\mathcal{L}^2(-1,1)} \Big) \| v_2 \|_{\mathcal{L}^2(-1,1)}, \end{aligned} \tag{2.25}$$

for $\forall v_1 \in \mathcal{L}^2(-1,1)$, $\forall v_2 \in \mathcal{P}_N(-1,1)$, the importance of the interpolation operator for the determination of an error estimate becomes clear The interpolation error is

bounded by (see e.g. [6])

$$\| v - I_N v \|_{\mathcal{L}^2(-1,1)} \leq CN^{-m} \| v \|_{\mathcal{H}^m(-1,1)} \quad (2.26)$$

$$\| v - I_N v \|_{\mathcal{H}^1(-1,1)} \leq CN^{1-m} \| v \|_{\mathcal{H}^m(-1,1)} \quad (2.27)$$

for $v \in \mathcal{H}^m(-1,1)$. The truncation error expressed in (2.15) plays an important role in the determination of Equations (2.26) and (2.27), as well as the properties of the Gauss-Lobatto-Legendre integration rules.

It is not possible to give estimates for the error $\| u - u_N \|_{\mathcal{H}^1(-1,1)}$ without being more specific about the operator A in (2.1). For example, if A is the operator $-d^2/dx^2$ (yielding $A_N = A$) the discretized problem reads as: Find $u_N \in X_N$ such that

$$(A_N u_N, v_N)_{GL} = (f, v_N)_{GL} \quad \forall v_N \in X_N, \quad (2.28)$$

or, equivalently

$$\sum_{i=0}^{N} \frac{du_N}{dx}(\xi_i)\frac{dv_N}{dx}(\xi_i)\rho_i = \sum_{i=0}^{N} f(\xi_i)v_N(\xi_i)\rho_i \quad \forall v_N \in X_N. \quad (2.29)$$

In [6], [14], and [47] it is shown that the discrete solution u_N of (2.29) is optimal, i.e. for m and n integers, $m \geq 1$, $n \geq 2$ and $u \in \mathcal{H}^m(-1,1)$, $f \in \mathcal{H}^n(-1,1)$

$$\| u - u_N \|_{\mathcal{H}^1(-1,1)} \leq C\{N^{1-m} \| u \|_{\mathcal{H}^m(-1,1)} + N^{-n} \| f \|_{\mathcal{H}^n(-1,1)}\}. \quad (2.30)$$

The proof of this error estimate can be found in the aforementioned literature and is based on the specific operator A, Equation (2.25) and, indirectly, on Equations (2.26), (2.27), and (2.14).

2.2 Spectral element method

The results of the previous paragraph were obtained for a Legendre-Galerkin spectral method. However, for these monodomain discretizations, it is not important to take the Galerkin form (2.18) as a starting point. In fact, there is an equivalence between Galerkin methods and collocation methods, which rely on a "strong" formulation, i.e. the residual $A_N u_N - f$ cancels on a set of "collocation" points. This is explained in e.g. [14] and [6].

When the domain is decomposed into a number (say K) of nonoverlapping subdomains on each of which the solution is approximated by Nth-order polynomials, the weak approach, however, leads to a totally different treatment of the interelemental interfaces. More precisely, the variational statement imposes "automatically" weak C^1 conditions, which means that u_N is continuous but with continuous derivatives only for $N \to \infty$. Collocation methods require that the continuity of the solution and its

normal derivatives is imposed explicitly (but not necessarily in a strong way), giving rise to intra-element iteration schemes. Recently, Gervasio et al. [32] developed such an iteration scheme for elliptic problems. The convergence rate of this method is independent of the polynomial degree N.

In the following, we will focus on the spectral element discretization technique (see Patera [57]) based on Legendre polynomials. This method is well documented in Maday and Patera [47] and in Rønquist [65]. By introducing a simple model problem, we will show that the spectral accuracy is maintained when the computational domain is decomposed in a number of spectral elements. The advantages of a decomposition of the domain are manifold and have been discussed in Chapter 1. We will immediately tackle the three-dimensional problem

$$-\Delta u = f \qquad \text{on } \Omega \tag{2.31}$$

$$u = 0 \qquad \text{on } \partial\Omega, \tag{2.32}$$

where $\partial\Omega$ denotes the boundary of the open domain Ω. The Equations (2.31), (2.32) have an equivalent variational form: Find $u \in \mathcal{H}_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega f v dx \quad \forall v \in \mathcal{H}_0^1(\Omega), \tag{2.33}$$

with

$$\mathcal{H}_0^1(\Omega) = \{v \in \mathcal{H}^1(\Omega); v(x_1, x_2, x_3) = 0 \text{ if } (x_1, x_2, x_3) \in \partial\Omega\}. \tag{2.34}$$

We then break up the domain Ω in K parallelepipedic nonoverlapping subdomains Ω_k

$$\Omega_k = (a_k, a_k') \times (b_k, b_k') \times (c_k, c_k') \qquad k = 1, .., K \tag{2.35}$$

$$\overline{\Omega} = \cup_{k=1}^K \overline{\Omega}_k, \qquad \Omega_i \cap \Omega_j = \emptyset \quad i, j = 1, .., K, \quad i \neq j. \tag{2.36}$$

We denote by l_x^k, l_y^k, and l_z^k the dimensions of the subdomain Ω_k

$$l_x^k = a_k' - a_k, \quad l_y^k = b_k' - b_k, \quad l_z^k = c_k' - c_k \quad k = 1, .., K. \tag{2.37}$$

The next step is to define the discrete space

$$X_N = \mathcal{H}_0^1(\Omega) \cap \mathcal{P}_{N,K}^3(\Omega), \tag{2.38}$$

with

$$\mathcal{P}_{N,K}^3(\Omega) = \{v \in \mathcal{L}^2; \quad v \mid_{\Omega_k} \in \mathcal{P}_N^3(\Omega_k)\} \tag{2.39}$$

and $\mathcal{P}_N^3(\Omega_k)$ the space of all polynomials on Ω_k of degree smaller than or equal to N in three variables. This leads to the following Galerkin formulation: Find $u_N \in X_N$ such that

$$\int_\Omega \nabla u_N \cdot \nabla v_N dx = \int_\Omega f v_N dx \qquad \forall v_N \in X_N. \tag{2.40}$$

For the same reasons as explained in Section 2.1, numerical quadrature based on the mapped Gauss-Lobatto-Legendre points $\{\xi^1_{h,k}\}^{k=1,..,K}_{h=0,..,N}$, $\{\xi^2_{i,k}\}^{k=1,..,K}_{i=0,..,N}$, and $\{\xi^3_{j,k}\}^{k=1,..,K}_{j=0,..,N}$,

$$\xi^1_{h,k} = a_k + \frac{1}{2}(a'_k - a_k)(\xi_h + 1) \qquad h = 0,1,..,N;\ k = 1,..,K \tag{2.41}$$

$$\xi^2_{i,k} = b_k + \frac{1}{2}(b'_k - b_k)(\xi_i + 1) \qquad i = 0,1,..,N;\ k = 1,..,K \tag{2.42}$$

$$\xi^3_{j,k} = c_k + \frac{1}{2}(c'_k - c_k)(\xi_j + 1) \qquad j = 0,1,..,N;\ k = 1,..,K, \tag{2.43}$$

is applied to compute the integrals in Equation (2.40) to obtain: Find $u_N \in X_N$ such that

$$\sum_{k=1}^{K}\sum_{h,i,j=0}^{N}\left\{\frac{l^k_y l^k_z}{2l^k_x}\frac{\partial}{\partial x_1}u_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k})\frac{\partial}{\partial x_1}v_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k}) \;+\right.$$

$$\frac{l^k_x l^k_z}{2l^k_y}\frac{\partial}{\partial x_2}u_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k})\frac{\partial}{\partial x_2}v_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k}) \;+$$

$$\left.\frac{l^k_x l^k_y}{2l^k_z}\frac{\partial}{\partial x_3}u_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k})\frac{\partial}{\partial x_3}v_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k})\right\}\rho_h\rho_i\rho_j \;=$$

$$\sum_{k=1}^{K}\sum_{h,i,j=0}^{N}\frac{l^k_x l^k_y l^k_z}{8}f(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k})v_N(\xi^1_{h,k},\xi^2_{i,k},\xi^3_{j,k})\rho_h\rho_i\rho_j \qquad \forall v_N \in X_N. \tag{2.44}$$

Equation (2.44) is the full 3D spectral element discretization of problem (2.31), (2.32) and is nothing else than the three-dimensional, multidomain analogue of Equation (2.29). The interfaces have been naturally taken care of by the variational form in combination with a proper choice for X_N. An error estimate for the solution u_N of (2.44) is given by

$$\| u - u_N \|_{\mathcal{H}^1(\Omega)} \leq C\{N^{1-m} \| u \|_{\mathcal{H}^m(\Omega)} + N^{1/2-n} \| f \|_{\mathcal{H}^n(\Omega)}\} \tag{2.45}$$

for $u \in \mathcal{H}^m_0$ and $f \in \mathcal{H}^n$, with $n, m \geq 1$ [47].

2.3 Practical considerations

In the previous Section 2.2, we introduced the spectral element method for a typical 3D problem. We will now explain in what (efficient) way a solution u_N can be obtained from (2.44). First, we will have to choose carefully the test functions v_N. This choice will not influence the error estimate (2.45), but has a great impact on the conditioning of the matrices and on the implementation on computers (e.g. the reduction of the operation count due to tensor-product factorizations and sparsity patterns). Unlike the *h-p* finite element methods (see for example [72] for an overview), where Legendre polynomials are taken to span the space X_N, the spectral element method relies on a tensor-product (nonorthogonal) basis of Lagrangian interpolants $\{h_i\}_{i=0}^N \subset \mathcal{P}_N$, such that

$$h_i(\xi_j) = \delta_{ij} \qquad i,j = 0,..N, \tag{2.46}$$

with δ_{ij} the Kronecker delta. This implies that a function $w_N \in X_N$ is uniquely represented by (see Equation (2.3))

$$w_N(x_1,x_2,x_3)_{|\Omega_k} = \sum_{h,i,j=0}^{N} (\hat{w}_N)_{hij}^k h_h(x_1)h_i(x_2)h_j(x_3), \qquad k = 1,..,K, \tag{2.47}$$

with

$$(\hat{w}_N)_{hij}^k = 0 \qquad \text{if } (\xi_{h,k}^1, \xi_{i,k}^2, \xi_{j,k}^3) \in \partial\Omega \tag{2.48}$$

$$(\hat{w}_N)_{hij}^k = (\hat{w}_N)_{pqr}^l \qquad \text{if } (\xi_{h,k}^1, \xi_{i,k}^2, \xi_{j,k}^3) = (\xi_{p,l}^1, \xi_{q,l}^2, \xi_{r,l}^3). \tag{2.49}$$

Equation (2.48) accounts for the homogeneous Dirichlet boundary conditions and Equation (2.49) for the continuity along the interfaces. We then choose v_N nonzero (unit value) in only one Gauss-Lobatto-Legendre (GLL) point or, in the case of an interface, in two or more coinciding GLL points (see Equation (2.49)), and zero elsewhere. We arrive at the discrete equations:

$$\sum_{k=1}^{K}{}' \sum_{l,m,n=0}^{N} A_{\alpha\beta\gamma lmn}^k u_{lmn}^k = \sum_{k=1}^{K}{}' B_{\alpha\beta\gamma}^k f_{\alpha\beta\gamma}^k \qquad \alpha,\beta,\gamma = 0,..,N. \tag{2.50}$$

Here, $\sum'$ denotes direct stiffness summation, corresponding to summing the contributions at the interfaces (see Equation (2.49)) and taking the boundary conditions into account (see Equation (2.48)). Note that we omitted the subscript N for the discrete solution (u_{lmn}^k instead of $(u_N)_{lmn}^k$). The matrices A^k and B^k are defined on an arbitrary element Ω_k as

$$\sum_{l,m,n=0}^{N} A_{\alpha\beta\gamma lmn}^k u_{lmn}^k = \frac{l_y^k l_z^k}{2l_x^k} \sum_{q=0}^{N} D_{q\alpha} \sum_{l,m,n=0}^{N} D_{ql}\delta_{\beta m}\delta_{\gamma n}\rho_q\rho_m\rho_n u_{lmn}^k$$

$$+ \frac{l_x^k l_z^k}{2l_y^k} \sum_{q=0}^{N} D_{q\beta} \sum_{l,m,n=0}^{N} D_{qm} \delta_{\alpha l} \delta_{\gamma n} \rho_l \rho_q \rho_n u_{lmn}^k$$

$$+ \frac{l_x^k l_y^k}{2l_z^k} \sum_{q=0}^{N} D_{q\gamma} \sum_{l,m,n=0}^{N} D_{qn} \delta_{\alpha l} \delta_{\beta m} \rho_l \rho_m \rho_q u_{lmn}^k$$

$$\alpha\beta, \gamma = 0, .., N \tag{2.51}$$

$$B_{\alpha\beta\gamma}^k f_{\alpha\beta\gamma}^k = \frac{l_x^k l_y^k l_z^k}{8} \rho_\alpha \rho_\beta \rho_\gamma f_{\alpha\beta\gamma}^k \qquad \alpha, \beta, \gamma = 0, .., N, \tag{2.52}$$

with $D_{pq} = \frac{dh_q}{dx}(\xi_p)$. Equation (2.50) is often written in its short form

$$Au_N = Bf. \tag{2.53}$$

Note that when Au_N is evaluated according to (2.51), $6KN^6 + O(KN^5)$ multiplications have to be performed. As was recognized by Orszag [55], this number can be reduced to $6KN^4 + O(KN^3)$ by rewriting (2.51) as

$$\sum_{l,m,n=0}^{N} A_{\alpha\beta\gamma lmn}^k u_{lmn}^k = \frac{l_y^k l_z^k}{2l_x^k} \sum_{q=0}^{N} D_{q\alpha} \rho_q \rho_\beta \rho_\gamma \sum_{l=0}^{N} D_{ql} u_{l\beta\gamma}^k$$

$$+ \frac{l_x^k l_z^k}{2l_y^k} \sum_{q=0}^{N} D_{q\beta} \rho_\alpha \rho_q \rho_\gamma \sum_{m=0}^{N} D_{qm} u_{\alpha m\gamma}^k$$

$$+ \frac{l_x^k l_y^k}{2l_z^k} \sum_{q=0}^{N} D_{q\gamma} \rho_\alpha \rho_\beta \rho_q \sum_{n=0}^{N} D_{qn} u_{\alpha\beta n}^k \quad \alpha, \beta, \gamma = 0, .., N. \tag{2.54}$$

Without this tensor-product factorization, which is due to the choice of the Lagrangian basis (and not to the absence of deformation), spectral element methods would be inefficient. Note that in the case of parallelepipedic spectral elements, the two one-dimensional matrices D can be multiplied in a preprocessing stage, reducing the number of multiplications to $3KN^4 + O(KN^3)$, and that A and B are symmetric matrices. Moreover, B is diagonal. Matrix A is never constructed explicitly and only three one-dimensional matrices have to be stored in order to compute an evaluation of Au.

The above discussed features (tensor-product factorizations, no explicit construction of the operator and low storage for evaluation) make iterative methods attractive to solve Equation (2.53). Since A is (semi-) positive definite and symmetric, the best

candidate is the conjugate gradient method (CGM). This method can be found in any textbook on numerical solution methods (see for example Golub and Van Loan [33]) and we refer to Chapter 7, where a parallel variant is discussed, for more details. We recall here that the CGM computes (assuming infinite arithmetic) an approximation u_{cg} to u_N in a finite number of iterations (smaller than or equal to KN^3, the number of unknowns), such that $\| u_{cg} - u_N \|$ is smaller than a certain, user-defined tolerance. In each iteration a set of direction vectors is computed at the cost of one matrix-vector multiplication. The number of iterations is proportional to $\sqrt{\kappa_A}$, with κ_A the condition number of the matrix; $\kappa_A = \lambda_{\max}/\lambda_{\min}$. For the operator A based on spectral element discretizations, we find (see e.g. [47])

$$\kappa_A \approx O(K_1^2 N^3), \tag{2.55}$$

where K_1 denotes the number of spectral elements in a typical space direction. This condition number is in general larger than for the operators based on classical finite difference, finite volume, or finite element methods on regular grids. This is explained by the fact that the GLL grid points are not equidistant, but concentrated near the boundaries. In fact, the minimum distance between two GLL points is of order $K_1^{-1}N^{-2}$. If the finite element operator is constructed on the GLL grid, the same condition number is obtained as for the spectral element operator.

Preconditioning techniques (see Chapter 1 and Chapter 6 for a general discussion; see [47] for more relevant details) can be used to reduce the number of iterations. Basically, a preconditioner (say P) should meet two requirements: First, it should be "spectrally close" to the original operator, meaning that the condition number of the product of P^{-1} and the operator should be close to the unit value. Second, the preconditioner should be easy to invert, since at each iteration a direction vector is to be premultiplied by P^{-1}. For this reason, we will always define a preconditioner by its inverse. In the typical case of the preconditioned conjugate gradient method (PCGM) for problem (2.53), we will use a diagonal preconditioner $P^{-1} = diag(A)^{-1}$.

In most practical situations, the domain can not be subdivided in a set of parallelepipeds. If that is the case, generally deformed spectral elements have to be used, leading to more complicated formula. The tensor-product structure is, however, maintained. The only negative effect is that the constant of the leading term in N expressing the operation count is larger. For example, an evaluation of the weak "second-order" derivative costs $6KN^4 + O(KN^3)$ in the case of deformed geometries, whereas $3KN^4 + O(KN^3)$ operations are needed in the nondeformed case. In the next section, we will further elaborate on this matter.

2.4 Deformed geometries

Let us consider Ω, a three-dimensional, open domain, decomposed into K spectral elements $\{\Omega_k\}_{k=1}^K$. In most practical situations, the geometry (or decomposition of the geometry) does not allow that Equation (2.35) is satisfied, that is, not every spectral element is a parallelepiped. In that case, a mapping F^k is introduced that links each spectral element Ω_k to the reference domain $\hat{\Omega} = (0,1)^3$. More precisely, F^k is such that

$F^k(\overline{\hat{\Omega}}) = \overline{\Omega}_k$, and $F^k(r_1, r_2, r_3) = (x_1^k, x_2^k, x_3^k)$, with $r \in \overline{\hat{\Omega}}$ and $x^k \in \overline{\Omega}_k$. Furthermore, we assume that F^k is in $\mathcal{H}^\mu$ for some $\mu > 1$ and that the Jacobian J_F is larger than some constant $p_1 > 0$. Then, each $w_N \in X_N$, with X_N as in (2.38) can be expanded on the subdomain Ω_k as

$$w_N^k(x_1, x_2, x_3) = w_{N|\Omega_k} \circ F^k(r_1, r_2, r_3) = \sum_{h,i,j=0}^{N} (\hat{w}_N)_{hij}^k h_h(r_1) h_i(r_2) h_j(r_3). \qquad (2.56)$$

Problem (2.40) becomes: Find $u_N \in X_N$ such that $\forall v_N \in X_N$

$$\int_{\hat{\Omega}} \nabla u_N \circ F \cdot \nabla v_N \circ F |J_F| dr = \int_{\hat{\Omega}} f_N \circ F \; v_N \circ F |J_F| dr, \qquad (2.57)$$

with F taking into account all the local mappings $\{F^k\}_{k=1}^K$. In order to avoid complex notations and long formulas, we will not give the complete, detailed expressions corresponding to Equation (2.57). The final formulation of the elliptic problem discretized on deformed spectral element can be found in, for example, Rønquist [65] and Timmermans [74] (in two dimensions). In Maday and Rønquist [50], it is suggested that the numerical integration of Equation (2.57) can be performed with a degree superior to N. However, the conclusion of the same paper is that the gain in precision of this so-called overintegration hardly compensates the additional cost. When "normal" Nth-order Gauss-Lobatto-Legendre numerical integration is applied to Equation (2.57), it is shown [50] that for $u \in \mathcal{H}_0^m$ and $f \in \mathcal{H}^n$, the error is bounded by the following expression:

$$\begin{aligned} \| u - u_N \|_{\mathcal{H}^1(\hat{\Omega})} \leq \; C \; \Big\{ & N^{-\min\{\mu-1, m-1\}} \| F \|_{\mathcal{H}^\mu(\hat{\Omega})} \| u \|_{\mathcal{H}^{\min\{\mu,m\}}(\hat{\Omega})} \\ & + \; (N-1)^{-\min\{\mu, m-1\}} \| F \|_{\mathcal{H}^\mu(\hat{\Omega})} \| u \|_{\mathcal{H}^{\min\{\mu,m-1\}}(\hat{\Omega})} \\ & + \; N^{-\min\{\mu,n\}} \| F \|_{\mathcal{H}^\mu(\hat{\Omega})} \| f \|_{\mathcal{H}^{\min\{\mu,n\}}(\hat{\Omega})} \Big\}. \end{aligned} \qquad (2.58)$$

In most practical situations, the continuous mapping F is unknown, and has to be approximated. One way of doing this is by transfinite interpolation, as proposed by Gordon and Hall [34]. This method is very well explained in the framework of spectral methods by Schneidesch [69, Chapter 4]. The term "transfinite" is used because this interpolation is exact for a non-denumerable number of points, i.e. the points at the element boundaries. In short, when a parametric description of the element boundaries $\{\partial\Omega_k\}_{k=1}^K$ is available, an approximation F_N to F is constructed such that $F_N^k(\partial\hat{\Omega}) = \partial\Omega_k$. To the author's knowledge, error estimates similar to (2.58), where F

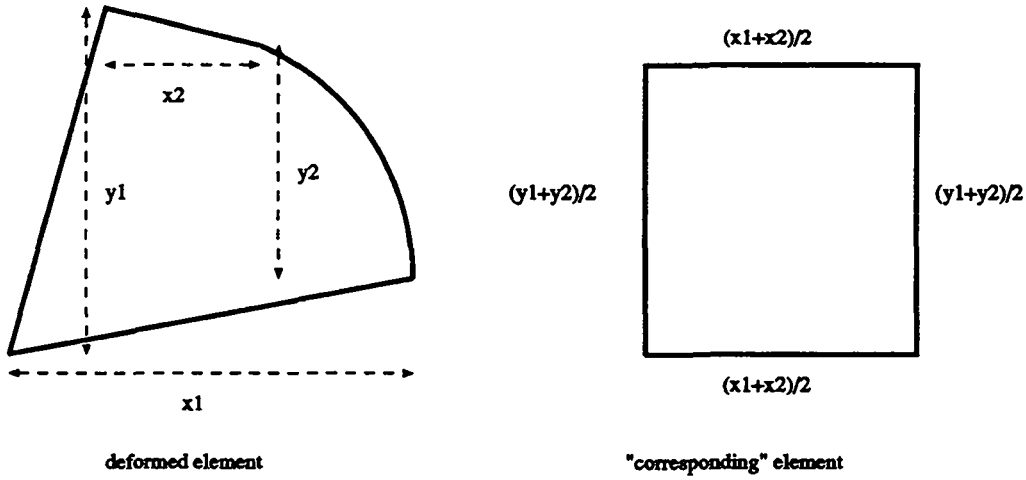

Figure 2.1: **Construction of a corresponding, nondeformed element (2D case). The differences of the x-coordinates of two opposite corners are defined as $x1$ and $x2$. The length of the nondeformed element in x-direction is then defined as the average of $x1$ and $x2$. Idem for y.**

is replaced by F_N, are only known if the interpolation is exact for all the grid points (see e.g. Métivet [52] or Schneidesch [69]). However, the transfinite interpolation error of boundary data is quadratic for the interior nodes, and the exponential decay of the error can only be verified numerically. Schneidesch [69] has found for a number of test cases that, indeed, spectral accuracy is preserved in the case of deformed geometries.

Throughout this thesis we will use isoparametric mappings, that is, the geometry is represented in X_N. For the (Navier-)Stokes equations, this might imply that the hydrostatic pressure mode is not preserved, but we have never encountered any problems in practice. Some preliminary tests, where we represented the geometry on the Gauss-Legendre grid did, however, show some difficulties, especially for small values of N.

Another issue is the influence of deformation on the condition number of the operator. This is important since many of the developed preconditioners (see Chapters 4 and 6) are constructed on nondeformed grids in order to allow fast elliptic solves. For example, let us consider the case of a geometry consisting of one two-dimensional, deformed spectral element Ω. We are interested in the condition number of the spectral element Laplace operator A as a function of the amount of deformation, and in the condition number of $A_{nd}^{-1}A$, with A_{nd} the same operator as A, but constructed on a "corresponding", nondeformed spectral element. The "corresponding" element (a concept which will return in other chapters) is defined by taking the average dimensions of the deformed element, as illustrated in Figure 2.1.

In Table I the condition numbers of A and $A_{nd}^{-1}A$ are given, and in Table II the number of PCGM iterations to solve $Au = B(1,..,1)^T$ with or without preconditioner A_{nd}. The two-dimensional, deformed spectral element Ω is given by its four sides

$\{\Gamma_i\}_{i=1}^4$:

$$\Gamma_1 = \{-1\} \times [-1,1], \qquad \Gamma_2 = \left\{\left(1+\alpha(y^2-1), y\right) \mid -1 \le y \le 1\right\},$$

$$\Gamma_3 = [-1,1] \times \{-1\}, \qquad \Gamma_4 = \{(x, 1+\alpha\sin(\pi(x+1)/2)) \mid -1 \le x \le 1\}. \quad (2.59)$$

In other words, Ω is the unit square with two deformed sides and the amount of

	A			$A_{nd}^{-1}A$		
N	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 1.0$	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 1.0$
5	10.1	10.2	18.8	1.00	1.29	6.21
7	21.6	21.8	52.8	1.00	1.42	11.2
9	37.5	38.2	122.3	1.00	1.52	15.6
11	58.6	60.4	214.8	1.00	1.59	17.9
13	85.9	90.7	334.5	1.00	1.66	20.8

Table I: **Condition number as a function of N and the amount of deformation α.**

	A not preconditioned			A preconditioned by A_{nd}		
N	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 1.0$	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 1.0$
5	10	16	18	1	12	17
7	21	33	43	1	13	33
9	35	49	69	1	14	42
11	48	64	97	2	15	51
13	60	80	127	2	15	57

Table II: **Number of PCGM iterations as a function of N and the amount of deformation α.**

deformation depends on α. From the Tables I and II, it is clear that the operators A and A_{nd} are the same in the case of $\alpha = 0$, corresponding to no deformation. When the rate of deformation is about 10% of the size of the element ($\alpha = 0.2$), the condition number of the preconditioned operator is only slightly larger than one, and the number of iterations is almost independent of N. For very large deformations, the condition number is still considerably smaller for the preconditioned operator, but the reduction of the number of iterations is no longer spectacular. This is because the number of iterations is related to the square root of the condition number. Finally, we remark that further numerical studies have shown that deformation affects the eigenvalues in the high and in the low range.

2.5 The steady Stokes problem

An efficient solution algorithm for the steady or unsteady Stokes equations is the basis of almost every Navier-Stokes solver. Especially in our case, where the nonlinear term will be treated explicitly (see Section 2.7), it is very important to analyze the theory on the spectral element discretization of the Stokes problem (see e.g. [47], [6], [46], [65]) and to develop efficient iterative and/or semi-direct solution methods. A central point in this discussion is the choice of the pressure space, which has to be done with great care in order to satisfy the so called inf-sup condition, according to Brezzi [11] and Babuška [5]. This condition guarantees existence and unicity of a solution to the Stokes problem, and, hence, the absence of spurious pressure modes and nonphysical wiggles. As we will see in this section, the inf-sup condition is satisfied if we take the pressure in a polynomial space that is of an order less by two than the velocity space.

Let us start by presenting the steady Stokes equations in an open domain Ω with boundary $\partial\Omega$:

$$\begin{aligned} -\nu\Delta\underline{u} + \nabla p &= \underline{f} && \text{on } \Omega && (2.60)\\ -\text{div } \underline{u} &= 0 && \text{on } \Omega && (2.61)\\ \underline{u} &= \underline{0} && \text{on } \partial\Omega, && (2.62) \end{aligned}$$

with $\underline{u} = (\underline{u}_1(x_1,x_2,x_3), \underline{u}_2(x_1,x_2,x_3), \underline{u}_3(x_1,x_2,x_3))^T$ the velocity, $p = p(x_1,x_2,x_3)$ the pressure and $\underline{f} = (\underline{f}_1(x_1,x_2,x_3), \underline{f}_2(x_1,x_2,x_3), \underline{f}_3(x_1,x_2,x_3))^T$ a body force for $(x_1,x_2,x_3) \in \Omega$. The kinematic viscosity is denoted by ν. The Stokes problem in its discrete form reads then: Find $(\underline{u}_N, p_N) \in X_N \times M_N$ such that

$$\begin{aligned} &\forall \underline{v}_N \in X_N \qquad \nu(\nabla\underline{u}_N, \nabla\underline{v}_N)_N - \langle p_N, \text{div } \underline{v}_N\rangle_N = (\underline{f}, \underline{v}_N)_N && (2.63)\\ &\forall q_N \in M_N \qquad -\langle \text{div } \underline{u}_N, q_N\rangle_N = 0, && (2.64) \end{aligned}$$

where X_N and M_N are the velocity and pressure spaces respectively and $(,)_N$ and $\langle,\rangle_N$ discrete inner products, to be defined later.

A naive and wrong choice for the spaces X_N and M_N would be $X_N = \mathcal{H}_0^1(\Omega)^3 \cap \mathcal{P}_{N,K}^3(\Omega)^3$, $M_N = \mathcal{L}_0^2(\Omega) \cap \mathcal{P}_{N,K}^3(\Omega)$. The space $\mathcal{L}_0^2(\Omega)$ is defined by

$$\mathcal{L}_0^2(\Omega) = \{v \in \mathcal{L}^2(\Omega) | \int_\Omega v dx = 0\}, \tag{2.65}$$

and fixes the pressure level. The incorrectness can be seen by introducing the concept of a spurious pressure mode: A function $q_N \in M_N$ is called a spurious mode of the Stokes problem (2.63), (2.64) if $\forall \underline{v}_N \in X_N$

$$\langle \text{div } \underline{v}_N, q_N\rangle_N = 0. \tag{2.66}$$

Now let us suppose that there exists a solution $(\underline{u}_N, p_N)$ to (2.63), (2.64) and that q_N satisfies (2.66). Then, $p_N + \alpha q_N$ is also a solution of (2.63), (2.64), for any real α, and

the uniqueness of the solution is no longer insured. As an example of a spurious mode we have $q_N = L_N(x) \in M_N$. Associated to this phenomena is the theory by Brezzi [11] and Babuška [5]. They presented a condition that, when satisfied, guarantees the uniqueness of the solution to the Stokes problem. This condition is known as the inf-sup condition: Problem (2.63), (2.64) is well posed if there is a real $\beta_N > 0$ such that

$$\beta_N = \inf\{q_N \in M_N, q_N \neq 0\} \sup\{\underline{v}_N \in X_N\} \frac{\langle \text{div } \underline{v}_N, q_N \rangle_N}{\| \underline{v}_N \|_{\mathcal{H}^1(\Omega)^3} \| q_N \|_{\mathcal{L}^2(\Omega)}}. \tag{2.67}$$

By definition (2.66), we see immediately that the inf-sup condition (2.67) is not satisfied if there exists a spurious mode in M_N. β_N is called the inf-sup constant and its value influences precision and convergence of the Uzawa pressure operator, as will be seen in the next section.

Taking into account the presence of spurious modes for the naive choice of the spaces X_N and M_N, another suggestion has to be made. It is possible to determine Z_N, the set containing all the spurious modes

$$Z_N = \{q_N \in M_N; \quad \langle \text{div } \underline{v}_N, q_N \rangle_N = 0 \quad \forall \underline{v}_N \in X_N\} \tag{2.68}$$

and to take $M_N = \mathcal{P}^3_{N,K}(\Omega) \cap Z_N^{\perp}$. This is, however, not a practical choice since some low-order polynomials can not be represented. A well-known solution is the so-called $\mathcal{P}_N - \mathcal{P}_{N-2}$ formulation, with

$$X_N = \mathcal{H}^1_0(\Omega)^3 \cap \mathcal{P}^3_{N,K}(\Omega)^3, \qquad M_N = \mathcal{L}^2_0(\Omega) \cap \mathcal{P}^3_{N-2,K}(\Omega). \tag{2.69}$$

This $\mathcal{P}_N - \mathcal{P}_{N-2}$ approach is well documented and can be considered as a classical method. The absence of spurious modes has been proven and has been described in many papers (see e.g. [47], [46] and [65]). Its implementation is also simple by taking $N+1$ point Gauss-Lobatto-Legendre quadrature rules to evaluate $(,)_N$, analogously to (2.22), and $N-1$ point Gauss-Legendre quadrature rules to evaluate $\langle,\rangle_N$. The Gauss-Legendre grid is shifted with respect to the Gauss-Lobatto-Legendre grid, that is, the two grids do not have any point in common.

For the mono-element case it has been shown (see e.g. [6]) that the inf-sup constant scales like $N^{-1/2}$ in two dimensions, and like N^{-1} in three dimensions. This dependence on the mesh size is not present in most finite element discretizations and is undesirable since β_N^{-1} appears in the error estimates for the pressure. Still on one element, it is shown [49] that for the unique discrete solution $(\underline{u}_N, p_N) \in X_N \times M_N$ as defined in (2.69) of the Stokes problem (2.63), (2.64) we have the following error estimates

$$\| \underline{u} - \underline{u}_N \|_{\mathcal{H}^1(\Omega)^3} \leq C \left\{ N^{1-m} \left(\| \underline{u} \|_{\mathcal{H}^m(\Omega)^3} + \| p \|_{\mathcal{H}^{m-1}(\Omega)} \right) + N^{-n} \| \underline{f} \|_{\mathcal{H}^n(\Omega)^3} \right\} \tag{2.70}$$

$$\| p - p_N \|_{\mathcal{L}^2(\Omega)} \leq C \left\{ N^{2-m} \left(\| \underline{u} \|_{\mathcal{H}^m(\Omega)^3} + \| p \|_{\mathcal{H}^{m-1}(\Omega)} \right) + N^{1-n} \| \underline{f} \|_{\mathcal{H}^n(\Omega)^3} \right\},$$

(2.71)

with $(\underline{u}, p)$ the solution of the continuous problem in $\mathcal{H}^m(\Omega)^3 \times \mathcal{H}^{m-1}(\Omega)$ and $\underline{f} \in \mathcal{H}^{n-1}(\Omega)^3$, $N, n \geq 2, m \geq 3$. These error estimates are optimal for the velocity, but not for the pressure. In two dimensions, the error estimate for the pressure is a factor $N^{-1/2}$ better, due to the inf-sup constant which is hidden in Equation (2.71). Fortunately, there is numerical evidence (see [46]) that, at least for values of N that are typically used for spectral element discretizations, the estimate (2.71) is pessimistic and that β_N can be considered as only weakly dependent on N. There is also a direct relation between β_N and the condition number of the Uzawa operator. This will be discussed in the next section.

We conclude this section with some remarks about other possible choices for the pressure space. Some authors (see e.g. Azaiez and Coppoletta [4]) use the internal GLL grid points instead of the GL points. This leads to a marginally better inf-sup constant. It is obvious that a main disadvantage of the $\mathcal{P}_N - \mathcal{P}_{N-2}$ formulation is that functions have to be inter- and extrapolated from one grid to another, giving rise to more expensive operations like derivation. Alternative approaches based on $\mathcal{P}_N - \mathcal{P}_N$ spaces exist. Phillips and Roberts [59], for example, use a singular-value decomposition of the pressure operator to filter the spurious modes. Recent work (Canuto [13], Canuto and Van Kemenade [15]) points out that stabilization by finite element bubble functions suppresses pressure oscillations. Other authors (see e.g. Demaret and Deville [23] and Pinelli and Vacca [60]) claim that some finite element resp. finite difference preconditioners are responsible for filtering the pressure wiggles. Another, more intuitive way to handle this problem is by considering a decoupling method applied to the *continuous* equations (see discussion in Chapter 5). This requires an additional boundary equation for the pressure (see Orszag et al. [56], Karniadakis et al. [42] and Timmermans [74]), for example $\frac{\partial p}{\partial n} = 0$, which presumably eliminates spurious pressure modes. Although nonphysical oscillations have never been reported, proof is still lacking.

2.6 The Uzawa method

The absence of the pressure operator in the continuity equation makes the coupled system (pressure/velocity) difficult to solve by an iterative method. Although this problem can be dealt with by strategies based on fill-in (see e.g. Dahl and Wille [22] for a finite element implementation), it is common practice to choose for a decoupling method, which is discussed in Chapter 5. In this section we will introduce the Uzawa method [2] and illustrate the relation between β_N and the condition number of the pressure matrix. The book of Bernardi and Maday [6] and the paper of Maday et al. [46] are suggested to interested readers.

First, we present the discrete Stokes equations in matrix form, with spaces as defined in (2.69) and the inner products evaluated by Gauss-Lobatto-Legendre and Gauss-

Legendre quadrature rules as described in the previous sections. We write

$$\nu A\underline{u}_N - D^T p_N = B\underline{f} \tag{2.72}$$

$$-D\underline{u}_N = 0. \tag{2.73}$$

The matrices A and B have been defined in (2.51) and (2.52). Throughout this thesis, we will not underline matrices to indicate that we deal with $\mathbf{3}KN^3 \times \mathbf{3}KN^3$ block matrices. The matrix D denotes the discrete divergence, for $\alpha, \beta, \gamma = 0, .., N-2$,

$$\begin{aligned} D\underline{u} &= \sum_{l,m,n=0}^{N} D_{\alpha\beta\gamma lmn} \left((\underline{u}_1)^k_{lmn}, (\underline{u}_2)^k_{lmn}, (\underline{u}_3)^k_{lmn} \right)^T \\ &= \sum_{k=1}^{K}{}' \sum_{l,m,n=0}^{N} \omega_\alpha \omega_\beta \omega_\gamma \left\{ \frac{l^k_y l^k_z}{4} \frac{\partial \tilde{h}_\alpha}{\partial x_1}(\xi_l) \tilde{h}_\beta(\xi_m) \tilde{h}_\gamma(\xi_n) (\underline{u}_1)^k_{lmn} \right. \\ &+ \left. \frac{l^k_x l^k_z}{4} \frac{\partial \tilde{h}_\beta}{\partial x_2}(\xi_m) \tilde{h}_\alpha(\xi_l) \tilde{h}_\gamma(\xi_n) (\underline{u}_2)^k_{lmn} + \frac{l^k_x l^k_y}{4} \frac{\partial \tilde{h}_\gamma}{\partial x_3}(\xi_n) \tilde{h}_\alpha(\xi_l) \tilde{h}_\beta(\xi_m) (\underline{u}_3)^k_{lmn} \right\}, \end{aligned} \tag{2.74}$$

and the symbol T the transpose. The set of Lagrangian interpolants on the $N-1$ point Gauss-Legendre grid is indicated by $\{\tilde{h}_i\}_{i=0}^{N-2}$, and the Gauss-Legendre weights by $\{\omega_i\}_{i=0}^{N-2}$. We remind that l^k_x, l^k_y and l^k_z represent the dimensions of the spectral element k, according to Equation (2.37) and that $\{\xi_i\}_{i=0}^{N}$ denotes the set of Gauss-Lobatto-Legendre grid points, according to the definitions in Section 2.1 and Section 2.2 (Equation (2.37)) respectively. We then multiply Equation (2.72) by DA^{-1} to obtain the following equations which are equivalent to (2.72) and (2.73)

$$Sp_N = -DA^{-1}\underline{f} \tag{2.75}$$

$$\nu A\underline{u}_N - D^T p_N = B\underline{f}, \tag{2.76}$$

with the Uzawa operator S defined as

$$S = DA^{-1}D^T. \tag{2.77}$$

We see that the computation of the pressure is completely decoupled from the computation of the velocities. Both the pressure equation (2.75) and the velocity equation (2.76) are solved by PCGM. It should be noted that each evaluation of the Uzawa operator requires the inversion of the operator A. Of course, matrix A is never inverted. Instead a PCGM is used, yielding two nested iterative methods for the computation of the pressure, which is, in most cases, rather expensive. These practical considerations will be further discussed in Chapters 4 and 5. Here, we are more interested in the theoretical aspects.

Since (2.75), (2.76) is equivalent to the original set of equations, existence and uniqueness of the solution are guaranteed by the inf-sup condition. We have for the continuous operators that $\nabla \cdot \Delta^{-1} \nabla \approx I$ and expect that the condition number of S with respect to the diagonal mass matrix on the Gauss-Legendre grid, $\tilde{B}$, is also of order unity. This suggests that P, defined by

$$P^{-1} = \tilde{B}^{-1} \tag{2.78}$$

is a good preconditioner for the steady Uzawa pressure operator. As has been recognized in for example [46], the condition number of the matrix $P^{-1}S$, κ_S, depends on the inf-sup constant β_N in the following way

$$\kappa_S = \frac{C}{\beta_N^2}, \tag{2.79}$$

with C a constant independent of N. So the number of outer pressure iterations scales like β_N^{-1}. We see that the number of outer pressure iterations is only of order unity if $\beta_N = O(1)$. For many discretization techniques this is indeed the case, but for spectral elements the inf-sup constant is not optimal. In three dimensions β_N scales like $O(N^{-1})$ and in two dimensions we have $\beta_N = O(N^{-1/2})$ (see [6], [47], and [46] for more details). Numerical tests (see [46]) indicate, however, that for values of N smaller than 16, this estimate is pessimistic and it is shown that the number of iterations for the preconditioned operator depends very slightly on N. The number of spectral elements K hardly influences β_N.

For the unsteady Stokes equations the situation is completely different, that is, more complicated. When we apply an implicit Euler backward scheme to discretize in time (see Chapter 3 for more details), we find the following spectral element formulation: Find $(\underline{u}_N, p_N) \in X_N \times M_N$ such that $\forall \underline{v}_N \in X_N$, $\forall q_N \in M_N$

$$(\frac{\underline{u}^{n+1} - \underline{u}_N^n}{\Delta t}, \underline{v}_N)_N + \nu(\nabla \underline{u}_N^{n+1}, \nabla \underline{v}_N)_N - \langle p_N^{n+1}, \text{div } \underline{v}_N \rangle_N = (\underline{f}^{n+1}, \underline{v}_N)_N \tag{2.80}$$

$$- \langle \text{div } \underline{u}_N^{n+1}, q_N \rangle_N = 0. \tag{2.81}$$

The upper index n refers to the time level. In matrix form, the equations (2.80) and (2.81) can be written as

$$S p_N^{n+1} = -DH^{-1} \underline{f}^{n+1} \tag{2.82}$$

$$H \underline{u}_N^{n+1} - D^T p_N^{n+1} = B \underline{f}^{n+1}, \tag{2.83}$$

with the unsteady Uzawa operator S and Helmholtz operator H defined as

$$S = DH^{-1}D^T, \qquad H = \Delta t^{-1} B + \nu A. \tag{2.84}$$

Note that we transferred the explicit term $\Delta t^{-1} B \underline{u}_N^n$ to the right-hand-side vector $\underline{f}^{n+1}$. We see that for large values of $\nu\Delta t$ the unsteady Uzawa operator tends to the steady operator (discarding constants) and can still be preconditioned by the inverse of the mass matrix on the Gauss-Legendre grid. For small values of $\nu\Delta t$ the situation is, however, different since S tends to the pseudo-Laplacian E, defined by

$$E = DB^{-1}D^T. \tag{2.85}$$

The preconditioning of this operator will be the subject of Chapter 6.

$N \times N \times N \times K$	$\| E(p) \|$	$\| E(\underline{u}_1) \|$
$4 \times 4 \times 4 \times 4$	$1.1_{10^{-2}}$	$1.3_{10^{-3}}$
$6 \times 6 \times 6 \times 4$	$5.2_{10^{-5}}$	$8.2_{10^{-6}}$
$8 \times 8 \times 8 \times 4$	$1.7_{10^{-7}}$	$2.5_{10^{-8}}$
$10 \times 10 \times 10 \times 4$	$3.0_{10^{-10}}$	$4.6_{10^{-11}}$
$12 \times 12 \times 12 \times 4$	$2.5_{10^{-12}}$	$6.1_{10^{-14}}$

Table III: l_2-error for Stokes problem

We will close this section by illustrating the spectral convergence for a steady Stokes problem with analytical solution

$$\underline{u}(x_1, x_2, x_3) = \left(-\cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2), \sin(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2), 0\right)^T \tag{2.86}$$

$$p(x_1, x_2, x_3) = -\pi \sin(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2) \tag{2.87}$$

which is computed by means of an unsteady solver. The steady solution is obtained in about eight ($N = 4$) to twenty-three ($N = 12$) time steps ($\nu = 1$, $\Delta t = \frac{1}{2}$). The results are given in Table III.

2.7 The discretization of the nonlinear term

The incompressible, transient Navier-Stokes equations in an open three-dimensional domain Ω are given in their nondimensional form:

$$\frac{\partial \underline{u}}{\partial t} - Re^{-1}\Delta\underline{u} + (\underline{u}\cdot\nabla)\underline{u} + \nabla p = \underline{f}, \tag{2.88}$$

$$-\operatorname{div}\underline{u} = 0 \tag{2.89}$$

and are subjected to no-slip conditions on the boundary $\partial\Omega$:

$$\underline{u} = \underline{0}. \tag{2.90}$$

The Reynolds number $Re = UL/\nu$ is based on a characteristic velocity U, a characteristic length L and the kinematic viscosity ν and describes the physical properties of the flow (i.e. the ratio between the inertia forces, through the convective term $(\underline{u}\cdot\nabla)\underline{u}$, and the viscous forces, through the diffusion term $\Delta\underline{u}$). The presence of the nonlinear convection term does not only increase the physical complexity of the fluid flow, but also complicates the numerical simulation. The discrete system becomes nonlinear and nonsymmetric, and sophisticated numerical techniques have to be applied to deal with these difficulties. A very common approach is to use explicit time advancement schemes for the nonlinear term. This both linearizes the equations and reduces the computational effort for the "inversion" of a Stokes system. In return, a restriction on the maximum allowed time step emerges (see Chapter 3).

The convection operator can be written in three different ways:

$$\nabla\cdot\underline{u}\,\underline{u} \qquad \text{conservative form} \tag{2.91}$$

$$(\underline{u}\cdot\nabla)\underline{u} \qquad \text{convective form} \tag{2.92}$$

$$\frac{1}{2}(\underline{u}\cdot\nabla)\underline{u} + \frac{1}{2}\nabla\cdot\underline{u}\,\underline{u} \qquad \text{skew-symmetric form.} \tag{2.93}$$

Although for $\underline{u}$ satisfying the incompressibility constraint, these expressions are equivalent in the continuous case, differences will appear after discretization, due to the non-exact representation of $\underline{u}$, div$\underline{u}$, and $\nabla\underline{u}$. Several numerical tests show that spectral element discretizations based on the convective form yield more accurate results than discretizations based on the skew-symmetric form, although the differences are very small.

So precision is hardly an argument to choose for any of the three representations. The distribution of the eigenvalues, however, could be decisive. Time-integration schemes for the convective operator are based on the fact that the eigenvalues are purely imaginary. If one of the three representations induces large numerical errors (i.e. large real parts for the eigenvalues), this could be a reason not to choose this form. Since it is not very common to speak about eigenvalues of nonlinear operators, we investigate the eigenvalues of the operator representing the convection of a passive scalar θ by an incompressible velocity field $\underline{u}$ in an open domain Ω. Corresponding to Expressions (2.91), (2.92), and (2.93), the respective equations are given by

$$\frac{\partial\theta}{\partial t} = -\nabla\cdot\underline{u}\theta, \qquad -\text{div}\ \underline{u} = 0 \tag{2.94}$$

$$\frac{\partial\theta}{\partial t} = -\underline{u}\cdot\nabla\theta, \qquad -\text{div}\ \underline{u} = 0, \tag{2.95}$$

$$\frac{\partial \theta}{\partial t} = -\frac{1}{2}\left(\underline{u}\cdot\nabla\theta + \nabla\cdot\underline{u}\theta\right), \qquad -\text{div}\,\underline{u} = 0, \tag{2.96}$$

with homogeneous Dirichlet boundary conditions on θ. By defining the GLL derivation operator C as $C = (C_1, C_2, C_3)^T$,

$$C\theta = \begin{pmatrix} \sum_{k=1}^{K} l^k_y l^k_z \frac{}{4} \sum_{q=0}^{N} D_{q\alpha}\rho_q\rho_\beta\rho_\gamma(\theta^k_{q\beta\gamma}) \\ \sum_{k=1}^{K} l^k_x l^k_z \frac{}{4} \sum_{q=0}^{N} D_{q\beta}\rho_q\rho_\alpha\rho_\gamma(\theta^k_{\alpha q\gamma}) \\ \sum_{k=1}^{K} l^k_x l^k_y \frac{}{4} \sum_{q=0}^{N} D_{q\gamma}\rho_q\rho_\alpha\rho_\beta(\theta^k_{\alpha\beta q}) \end{pmatrix}, \quad \alpha,\beta,\gamma = 0,..,N, \tag{2.97}$$

with

$$D_{pq} = \frac{dh_q}{dx}(\xi_p), \tag{2.98}$$

we obtain an expression for the convective terms, for instance $\underline{u}\cdot\nabla\theta \approx \underline{u}_1 C_1\theta + \underline{u}_2 C_2\theta + \underline{u}_3 C_3\theta$. We can then investigate the eigenvalues of the discrete convective operator in the conservative, convective and skew-symmetric form by solving respectively

$$-(C_1\underline{u}_1 + C_2\underline{u}_2 + C_3\underline{u}_3)\underline{x} = \lambda B\underline{x} \tag{2.99}$$

$$-(\underline{u}_1 C_1 + \underline{u}_2 C_2 + \underline{u}_3 C_3)\underline{x} = \lambda B\underline{x} \tag{2.100}$$

$$-\frac{1}{2}\left(\underline{u}_1 C_1 + \underline{u}_2 C_2 + \underline{u}_3 C_3 + C_1\underline{u}_1 + C_2\underline{u}_2 + C_3\underline{u}_3\right)\underline{x} = \lambda B\underline{x}. \tag{2.101}$$

For the sake of simplicity, we took a two-dimensional geometry ($\Omega = (-1,0)\times(-1,2)$, $K = 2$, $l^1_x = 1$, $l^2_x = 2$) and computed the eigenvalues of the convection operator pre-multiplied by the inverse of the diagonal GLL mass matrix for two different solenoidal fields $\underline{u}$; $\underline{u} = \underline{1}$ and $\underline{u} = (-\sin(x)\cos(y), \sin(x)\cos(y))^T$. Dirichlet boundary conditions are applied. The results for the constant velocity field are shown in Table IV and for the non-constant velocity field in Table V. We remark that the largest imaginary eigenvalue grows like $O(N^2)$ and that their is no difference in the conservative, convective, and skew-symmetric form for $\underline{u} = \underline{1}$. However, if the velocity can not be represented exactly, the skew-symmetric form is preferable, since the real parts of the eigenvalues are always zero (at machine accuracy). The conservative and convection forms give rise to appreciable real components with positive and negative signs.

It is interesting to study the influence of these eigenvalues that are not purely imaginary. Let us take the case $N = 10$ and suppose that a fourth-order explicit Runge

	conservative form		convective form		skew-symmetric form	
N	max real	max imag.	max real	max imag.	max real	max imag.
6	1.507E-15	20.914	1.507E-15	20.914	1.507E-15	20.914
8	9.326E-15	33.467	9.326E-15	33.467	9.326E-15	33.467
10	1.526E-13	48.654	1.526E-13	48.654	1.526E-13	48.654
12	2.621E-13	66.833	2.621E-13	66.833	2.621E-13	66.833
14	1.719E-12	88.177	1.719E-12	88.177	1.719E-12	88.177

Table IV: **Maximum value of real and imaginary part (in absolute value) of the convective operator for a constant velocity** $\underline{u} = \underline{1}$.

	conservative form		convective form		skew-symmetric form	
N	max real	max imag.	max real	max imag.	max real	max imag.
6	0.218	9.171	0.218	9.171	3.553E-15	9.208
8	0.248	15.067	0.248	15.067	6.217E-15	15.096
10	0.217	22.590	0.217	22.590	2.309E-14	22.607
12	0.292	31.791	0.292	31.791	2.132E-14	31.804
14	0.286	42.626	0.286	42.626	3.739E-14	42.635

Table V: **Maximum value of real and imaginary part (in absolute value) of the convective operator for a non-constant velocity** $\underline{u} = (-\sin(y)\cos(x), \sin(x)\cos(y))^T$.

Kutta scheme is used to discretize Equation (2.95) in time (in Chapter 3 we will comment on time schemes for the convective terms). From Table V we find that the largest imaginary eigenvalue of the convection operator is 22.590i. In practice, the time step will be chosen such that this eigenvalue is just inside the stability region of the Runge Kutta method, which intersects the imaginary axis at $2\sqrt{2}$. In Figure 2.2, the eigenvalues of the convective operator are displayed together with the stability region of the Runge Kutta method, blown up by a factor $\frac{2}{22.590}\sqrt{2}$. In this way, stability is assured for the eigenvalues along the imaginary axis, and we investigate if the same is true for eigenvalues with a nonzero real part. To this end, we zoom in on the right-half plane (see Figure 2.3). It is observed that some eigenvalues are located outside the stability zone, which may lead to instabilities when integrating over a long time span. This is a firm argument in favour of a skew-symmetric formulation. However, in practice we have never encountered any problems with the convective form. A heuristic explanation is that we always solve convection/diffusion problems in which not only the eigenvalues of the different operators, but also the stability zones of an explicit and an implicit time scheme interact. This could imply that the eigenvalues with a positive real part are shifted inside the stability region of the overall time scheme and do not impose restrictions on the maximum allowed time step. In Section 3.3 this assumption is confirmed by a stability analysis.

In this section we have only considered homogeneous Dirichlet boundary conditions. However, the eigenvalue distribution will be completely different when, for example,

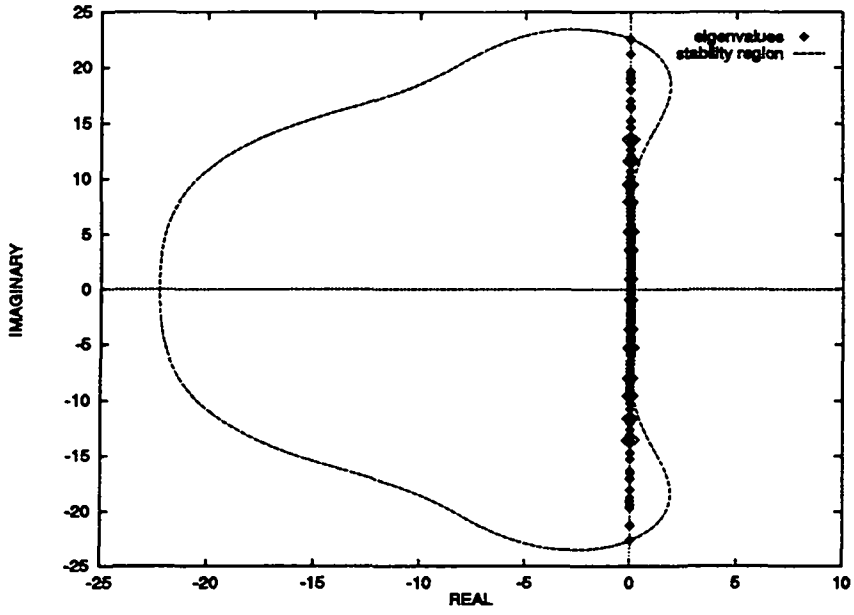

Figure 2.2: Eigenvalues of the convective operator for a non-constant velocity field ($N = 10$, $K = 2$) and the stability region of the fourth-order Runge-Kutta method, blown up by such a factor that the largest imaginary eigenvalue is just inside this region. The Runge-Kutta method is stable at the interior of the marked region.

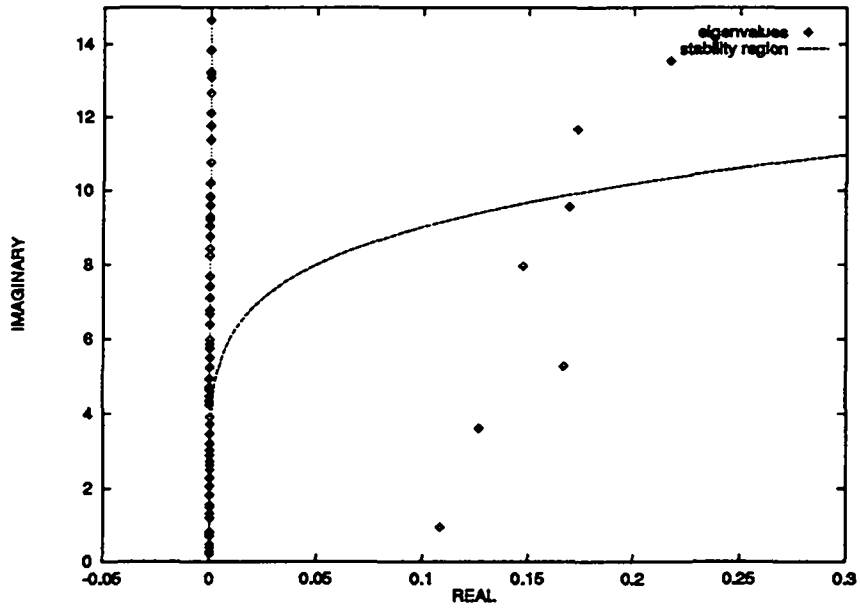

Figure 2.3: Eigenvalues of the convective operator for a non-constant velocity field ($N = 10$, $K = 2$) and the stability region of the fourth-order Runge-Kutta method, blown up by such a factor that the largest imaginary eigenvalue is just inside this region. Zoom on the right-half complex plane. The Runge-Kutta method is stable left/above the dotted line.

outflow boundary conditions are present somewhere along $\partial\Omega$. In fact, the discretization of the convection operator relies on integration by parts:

$$\int_\Omega v\underline{u}\cdot\nabla\theta dx = \int_\Omega \theta\underline{u}\cdot\nabla v dx - \int_{\partial\Omega} v\theta\underline{u}\cdot\underline{n}dS, \tag{2.102}$$

where $\underline{n}$ denotes the outward unit normal on the boundary $\partial\Omega$. The fact that the surface integral in Equation (2.102) is non-vanishing for outflow boundary conditions causes the eigenvalues to move to the left complex half plane. In general, this is not a serious problem since most time-integration schemes have large stability zones in this half plane. We observed numerically that the real parts of the eigenvalues are negative or zero for the skew-symmetric formulation. Both the conservative and the convective form, however, possess eigenvalues with positive real parts of about the same size as the ones displayed in Table IV. Like for homogeneous Dirichlet conditions, a small amount of diffusion will prevent the method to become unstable.

Chapter 3

Time discretization

High-order spatial discretization techniques are useful to obtain either accurate results or to reduce the number of grid points. The same argument can be used for high temporal accuracy: On the one hand, there exist physical problems, like the computation of the trajectory of a rocket, or, more related to our field of interest, the simulation of turbulence, that require very accurate approximations in time to perform statistics. On the other hand, for most problems that are treated in this thesis, it is of importance to maintain a certain level of precision when advancing in time with the "largest possible" time step. Since, in general, the computational cost of a first-order method is hardly any lower than for higher-order methods, the same level of temporal accuracy can much faster be obtained by a high-order method. Some attempts have been made (see e.g. Morchoisne [54]) to develop spectral methods in time, but they are seldomly used because of the large number of coupled systems of equations that have to be solved. In general, one is satisfied with third- or fourth-order accuracy in time. In this chapter we will go up to third-order methods and we will give some guidelines to increase the order.

Often, the time step is not determined by considerations of accuracy, but by a stability condition. This is for example the case for nonlinear, steady problems, where time errors should not be present in the time-converged solution. The only reason not to use an "infinite" time step is that the stability condition, due to the explicit treatment of the nonlinear terms, is violated.

We will start this chapter by introducing some basic concepts and by discussing the characteristics of some classical time-integration schemes. The necessity to split the nonlinear terms from the linear terms will be the subject of Section 3.2. There are several ways to combine a time-integration scheme for the nonlinear terms, say SN, and a scheme for the linear terms, say SL. This has to be done carefully since the overall order is not simply the minimum of the orders of SN and SL. There is also a splitting error, the order of which will be studied in this chapter. Our goal is to develop splitting schemes that do not degenerate the overall order of time accuracy. For steady problems this is unfortunately not always possible, because some splitting methods lead to non-vanishing time errors in the time-converged solution. Two splitting schemes will also be compared with respect to stability. The operator-integration-factor splitting method (see Maday et al. [48]) is especially developed to alleviate the stability constraint on

the time step. Classical stability analyses only give relevant results for the two extreme situations: purely convective and purely diffusive flows. In Section 3.3 we will propose an original way to investigate the stability for any ratio of convection to diffusion. The predicted value of the maximum allowed time step is verified by some runs of the spectral element code for different values of the Reynolds number. A similar analysis will allow us to explain that the non-negligible real parts of the eigenvalues of the convective form of the nonlinear term do not lead to unstable computations.

3.1 Basic concepts

In this section we will be concerned with the numerical approximation of the following initial boundary value problem of partial differential equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= g(u,t) \qquad t \in (t^0, T] \\ u(t^0) &= u_0. \end{aligned} \tag{3.1}$$

The right-hand-side function g is in general nonlinear and contains the spatial part of the partial differential equation. Equation (3.1) reduces to an initial value problem (IVP) for a system of ordinary differential equations by the method of lines [14]:

$$Q_N \frac{du_N}{dt} = Q_N g_N(u_N, t), \tag{3.2}$$

with u_N, typically, the spectral element approximation of u and Q_N the projection from the continuous space into the polynomial space $\mathcal{P}_{N,K}$. Equation (3.2) is often referred to as the semi-discretization of Equation (3.1). We set $y(t) = Q_N u_N(t)$, corresponding to the approximated values of $u(t)$ at the interior Gauss-Lobatto-Legendre grid points and those at the interfaces. We also introduce the function f as $f(y(t), t) = Q_N g(u_N(t), t)$. Equation (3.2) is then rewritten as

$$\frac{dy}{dt} = f(y(t), t). \tag{3.3}$$

As has been stated in the introduction to this chapter, high-order temporal discretization is essential in the context of spectral element solvers for the Navier-Stokes equations. In our opinion, the time scheme has to be of an order three or more. When we speak of the order of a method, we refer to the *global* order, i.e. the order to integrate in time over an interval $[t^0, T]$, with a time step Δt, $\Delta t \ll T - t^0$. The *local* time

error is the error at $t = t^{n+1}$, assuming that the previous solutions at $t^0 \leq t \leq t^n$ are exact. This local order is equal to the global order plus one. More precisely, the numerical time-integration scheme produces a series of approximations $\{y^n\}_{n=1}^{N_t}$ for $\{y(t^n) = y(t^0 + n\Delta t)\}_{n=1}^{N_t}$, with $N_t\Delta t = T - t^0$. The local error at $t = t^{n+1}$ is defined by

$$|y(t^{n+1}) - y^{n+1}|, \qquad \text{assuming that } y^1, y^2, .., y^n \text{ are exact.} \tag{3.4}$$

The global error is defined by

$$|y(T) - y^{N_t}|. \tag{3.5}$$

It is important to know that small changes in the initial values only produce bounded changes in the numerical approximations. We term this concept stability (see e.g. Gear [31]). In fact, time discretization is said to be stable if there exist constants δ, C, and M independent of Δt, such that, for all $T > t^0$

$$|y^n| \leq Ce^{MT}|y^0|, \tag{3.6}$$

for $t^0 \leq t^n \leq T$ and for all $0 \leq \Delta t < \delta$. In practice, the stability condition (3.6) is not restrictive enough, since it allows exponential growth of the numerical solution. Therefore, we introduce the concept of asymptotical or absolute stability [31]: A method is absolutely (asymptotically) stable for a given step size Δt and a given differential equation if the change due to a perturbation of size δ in one of the approximated values y^n is no larger than δ in all subsequent values. This definition depends on the differential equation and is therefore useless. We will define absolute stability for the scalar test equation $dy/dt = \lambda y$, with λ a complex constant. The region of absolute stability is the set of all $\lambda\Delta t$ such that $|y^{N_t}|$ is bounded as $T \to \infty$. Furthermore, a method is called A-stable if the region of absolute stability includes the complex left-half plane. A method is said to be absolutely stable for a linear system of differential equations $dy/dt = Ay$ if all the eigenvalues λ_i of A lie within the region of absolute stability of the scalar test equations $dy/dt = \lambda_i y$. Often, absolute stability is simply referred to as stability. This is because the "real" concept of stability is rarely used. In the following we will also use the term stability instead of absolute stability.

Three families of time-integration schemes are relevant within the framework of this thesis. First, we discuss the explicit Adams-Bashforth methods. The general form of an s-step Adams-Bashforth method (ABMs) is

$$y^{n+1} = y^n + \Delta t \sum_{i=0}^{s-1} \gamma_i f(y^{n+i-s+1}, t^{n+i-s+1}). \tag{3.7}$$

The method requires s start-up values $y^0, y^1, .., y^{s-1}$. The global error of ABMs is s. The coefficients of the ABMs up to order 4 are given in Table I.

As an example, we will investigate the stability of the ABM3:

$$y^{n+1} = y^n + \Delta t \left(\frac{23}{12} f(y^n, t^n) - \frac{4}{3} f(y^{n-1}, t^{n-1}) + \frac{5}{12} f(y^{n-2}, t^{n-2}) \right). \tag{3.8}$$

By applying this method to the scalar test equation $dy/dt = \lambda y$, we arrive at

$$y^{n+1} - y^n - \frac{23\Delta t\lambda}{12}y^n + \frac{4\Delta t\lambda}{3}y^{n-1} - \frac{5\Delta t\lambda}{12}y^{n-2} = 0. \tag{3.9}$$

This is a homogeneous difference equation which possesses solutions of the form $\zeta^p = y^{n-2+p}$ (note that the upper index on the left-hand side is an exponential, whereas the upper index on the right-hand side is an iteration index). Substituting in Equation (3.9) leads to

$$\zeta^3 - (1 + \frac{23\Delta t\lambda}{12})\zeta^2 + \frac{4\Delta t\lambda}{3}\zeta - \frac{5\Delta t\lambda}{12} = 0. \tag{3.10}$$

The left-hand side of Equation (3.10) defines the characteristic polynomial C of the ABM of order three;

$$C(\zeta, \lambda, \Delta t) = \zeta^3 - (1 + \frac{23\Delta t\lambda}{12})\zeta^2 + \frac{4\Delta t\lambda}{3}\zeta - \frac{5\Delta t\lambda}{12}. \tag{3.11}$$

A method is called (strongly) stable for a given $\lambda\Delta t$ if all the roots ζ_i of C are in absolute value smaller than one:

$$C(\zeta_i, \lambda, \Delta t) = 0 \Rightarrow |\zeta_i| < 1. \tag{3.12}$$

This statement is equivalent to the aforementioned definition of an absolutely stable method. Plots of the stability regions of several ABMs are given in, for example, [14] and in [31]. They are symmetric with respect to the real axis. Two parameters are of special interest: β_{imag}, the intersection of the stability region with the positive imaginary axis such that the method is stable for all imaginary values of $\Delta t\lambda$ that are smaller than β_{imag}; and β_{real}, the intersection with the negative real axis such that the method is stable for all negative, real values of $\lambda\Delta t$ larger than β_{real}. For ABM3 we have $\beta_{imag} = .723$ and $\beta_{real} = -0.545$ [14]. The values of the β_{imag} and β_{real} for the ABM up to an order of four are given in Table I.

The second family of time-integration schemes is that of the implicit backward differentiation formulas of order s (BDFs). We confine ourselves to $1 \leq s \leq 4$. Their general form is given by:

$$\beta_s y^{n+1} + \beta_{s-1}y^n + .. + \beta_0 y^{n+1-s} = \Delta t f(y^{n+1}, t^{n+1}). \tag{3.13}$$

The implicit character is obvious due to the term $f(y^{n+1}, t^{n+1})$ in the right-hand side. Note that the number of implicit relations to be solved remains the same for increasing order of the BDF. Hence, the computational cost does not augment with the order.

s	coefficients $\{\gamma_i\}_{i=0}^{s-1}$	β_{imag}	β_{real}
1	$\gamma_0 = 1$	0	-2
2	$\gamma_0 = -\frac{1}{2}, \gamma_1 = \frac{3}{2}$	0	-1
3	$\gamma_0 = \frac{5}{12}, \gamma_1 = -\frac{4}{3}, \gamma_2 = \frac{23}{12}$	.723	-.545
4	$\gamma_0 = -\frac{3}{8}, \gamma_1 = \frac{37}{24}, \gamma_2 = -\frac{59}{24}, \gamma_3 = \frac{55}{24}$	.430	-.3

Table I: **Coefficients and β_{imag} and β_{real} for Adams-Bashforth methods of order $s \leq 4$.**

Like for the family of ABM, a BDF requires start-up values. In order to be consistent, the local order of the start-up values has to be s, and they should be computed by a stable, implicit method. In this spirit, the BDF3 can be started with the initial condition y^0 and by y^1 computed by, for example, the method of Crank-Nicolson. The BDFs have $\beta_{real} = -\infty$. Moreover, the roots of the characteristic polynomial tend to zero for $|\lambda\Delta t| \to \infty$, for all $\lambda\Delta t$ that possess a nonpositive real part. The coefficients and β_{imag} can be found in Table II. The stability regions of the BDF's of order three and four do not contain the imaginary axis near the origin ($\beta_{imag} = 0$).

s	coefficients $\{\beta_i\}_{i=0}^{s}$	β_{imag}	β_{real}
1	$\beta_0 = -1, \beta_1 = 1$	∞	$-\infty$
2	$\beta_0 = \frac{1}{2}, \beta_1 = -2, \beta_2 = \frac{3}{2}$	∞	$-\infty$
3	$\beta_0 = -\frac{1}{3}, \beta_1 = \frac{3}{2}, \beta_2 = -3, \beta_3 = \frac{11}{6}$	0	$-\infty$
4	$\beta_0 = \frac{1}{4}, \beta_1 = -\frac{4}{3}, \beta_2 = 3, \beta_3 = -4, \beta_4 = \frac{25}{12}$	0	$-\infty$

Table II: **Coefficients, β_{imag}, and β_{real} for backward differentiation formulas of order $s \leq 4$.**

Finally, we introduce the second- and fourth-order explicit Runge-Kutta method (RK2 and RK4, resp.), which we will use in this chapter. These schemes do not fall within the class of multistep methods, like BDF and ABM. For the approximation of $y(t^{n+1})$, we only need y^n. The RK2 requires two function evaluations and RK4 four. We call such schemes two- and four-stage methods respectively. The RK2 scheme can be written as follows:

$$y^{n+1} = y^n + \Delta t f\left(y^n + \frac{1}{2}\Delta t f(y^n, t^n), t^n + \frac{\Delta t}{2}\right). \tag{3.14}$$

The RK4 method is given by

$$y^{n+1} = y^n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \tag{3.15}$$

with

$$
\begin{aligned}
k_1 &= f(y^n, t^n) \\
k_2 &= f(y^n + \frac{\Delta t k_1}{2}, t^n + \frac{\Delta t}{2}) \\
k_3 &= f(y^n + \frac{\Delta t k_2}{2}, t^n + \frac{\Delta t}{2}) \\
k_4 &= f(y^n + \Delta t k_3, t^n + \Delta t).
\end{aligned} \tag{3.16}
$$

The stability characteristics for RK2 are given by $\beta_{real} = -2$ and $\beta_{imag} = 0$. The stability region for RK4 is bounded by $\beta_{real} = -2.79$ and $\beta_{imag} = 2\sqrt{2}$. The RK4 method is especially interesting because of its large stability region along the imaginary axis. The associated β_{imag} is almost four times as large as β_{imag} for AB3, which is also an explicit method. In fact, the maximum value for β_{imag} that can be obtained by a four-stage Runge-Kutta method is 3, which is only slightly larger than $2\sqrt{2}$. The significance of large stability regions along the imaginary axis will become clear in the next section.

3.2 Splitting of the linear and nonlinear terms

In the previous chapter we discussed the space discretization of the linear and nonlinear terms in time. In order to point out the typical problems associated with the time advancement of the Navier-Stokes equations, we will use a simplified equation that highlights the same difficulties:

$$
B\frac{\partial \underline{u}(t)}{\partial t} = -Re^{-1}A\underline{u}(t) + N_L(\underline{u}(t), t). \tag{3.17}
$$

We have used the same notations as in Chapter 2. N_L is the nonlinear operator and does not represent a linear matrix, like B and A. Since the eigenvalues of $B^{-1}A$ grow like $O(K_1^2 N^4)$ (see [65] and Section 2.3) [1] , an implicit time-integration scheme is recommanded. It is, however, not a good idea to treat the nonlinear terms also by an implicit scheme, since a system of nonlinear equations is expensive to solve. But

[1] In Section 2.3, we have investigated the eigenvalues of A (and not of $B^{-1}A$) and determined its condition number. It is this condition number that is related to the rate of convergence. In the context of time-integration schemes, however, the eigenvalues of $B^{-1}A$ are important [65], since the term $\partial/\partial t$ is premultiplied by the diagonal mass matrix B. The latter matrix is responsible for another asymptotic behaviour of the condition number, i.e. a multiplication of $K_1 N^3$ by a factor N. Note that the eigenvalue analysis of the convection operator in Section 2.7 has already been performed with respect to B.

even after linearization, we would end up with a nonsymmetric, indefinite operator. Therefore, the nonlinear term is usually discretized by an explicit time scheme. In this way, linearization is not necessary and the eigenvalues will be imaginary or complex with small real parts, in the case of a skew-symmetric or convective formulation respectively. Let us assume that the eigenvalues are imaginary and grow (with respect to B^{-1}) as $K_1 N^2$, as has been demonstrated in Section 2.7. The fact that the eigenvalues of the nonlinear operator scale in absolute value like $O(K_1 N^2)$ and not like $O(K_1^2 N^4)$ is another argument to use an explicit scheme.

Amongst the available implicit time-integration schemes we choose the family of BDF, because of their simplicity, also for order 3 and 4, and because of their advantageous stability properties. Since they are stable for all values of $\lambda \Delta t$ along the negative real axis, there is no condition on the time step arising from the integration of the linear, diffusive terms. The choice of the time-integration scheme for the nonlinear terms, and the way in which it is combined with a BDF are less trivial and will be discussed in the following.

3.2.1 Splitting by combining multistep methods

A scheme that is often used for the nonlinear terms is the third-order Adams-Bashforth scheme (AB3). The BDF1/AB3 scheme applied to the IVP (3.17) yields

$$\begin{aligned} B\underline{u}^{n+1} &= B\tilde{\underline{u}}^n - \Delta t Re^{-1} A\underline{u}^{n+1} && (3.18)\\ B\tilde{\underline{u}}^n &= B\underline{u}^n \end{aligned}$$

$$+ \quad \Delta t \left(\frac{23}{12} N_L(\underline{u}^n, t^n) - \frac{4}{3} N_L(\underline{u}^{n-1}, t^{n-1}) + \frac{5}{12} N_L(\underline{u}^{n-2}, t^{n-2}) \right). \quad (3.19)$$

It is easily verified that the local error (LE) of this scheme is given by

$$\mathrm{LE} = \frac{1}{2}(\Delta t)^2 B^{-1} \left(\frac{dN_L(\underline{u}(t), t)}{dt} - \frac{dRe^{-1} A\underline{u}(t)}{dt} \right)_{t=t^{n+1}} + O(\Delta t)^3. \quad (3.20)$$

Hence, the BDF1/AB3 scheme is globally of order one. The extension to higher-order schemes is not trivial. If we apply BDF2 in combination with AB3, we obtain

$$\begin{aligned} 3B\underline{u}^{n+1} &= 4B\tilde{\underline{u}}^n - B\underline{u}^{n-1} - 2\Delta t Re^{-1} A\underline{u}^{n+1} && (3.21)\\ B\tilde{\underline{u}}^n &= B\underline{u}^n \end{aligned}$$

$$+ \quad \Delta t \left(\frac{23}{12} N_L(\underline{u}^n, t^n) - \frac{4}{3} N_L(\underline{u}^{n-1}, t^{n-1}) + \frac{5}{12} N_L(\underline{u}^{n-2}, t^{n-2}) \right), \quad (3.22)$$

and find a local error of order Δt. We can improve this result by replacing (3.22) by

$$B\tilde{\underline{u}}^n = B\underline{u}^n + \frac{1}{2}\Delta t\left(\frac{23}{12}N_L(\underline{u}^n,t^n) - \frac{4}{3}N_L(\underline{u}^{n-1},t^{n-1}) + \frac{5}{12}N_L(\underline{u}^{n-2},t^{n-2})\right), \quad (3.23)$$

to find a local error

$$\text{LE} = \frac{1}{3}(\Delta t)^2 B^{-1}\left(\frac{dN_L(\underline{u}(t),t)}{dt}\right)_{t=t^{n+1}} + O(\Delta t)^3. \quad (3.24)$$

This error is due to time splitting. A more systematic approach has been proposed by Ascher et al. [3].

Another, very simple method has been used by Karniadakis et al. [42] and is nothing more than an extrapolation scheme for the non-linear terms. More precisely, we apply a BDFs (here BDF3) to the linear *and* nonlinear terms:

$$B\underline{u}^{n+1} = \frac{18}{11}B\underline{u}^n - \frac{9}{11}B\underline{u}^{n-1} + \frac{2}{11}B\underline{u}^{n-2} + \frac{6}{11}\Delta t\left(-Re^{-1}A\underline{u}^{n+1} + N_L(\underline{u}^{n+1},t^{n+1})\right). \quad (3.25)$$

However, instead of solving the implicit relation for the nonlinear terms, we use an extrapolation scheme (EX3) to determine $N_L(\underline{u}^{n+1},t^{n+1})$:

$$N_L(\underline{u}^{n+1},t^{n+1}) = 3N_L(\underline{u}^n,t^n) - 3N_L(\underline{u}^{n-1},t^{n-1}) + N_L(\underline{u}^{n-2},t^{n-2}) + O(\Delta t)^3. \quad (3.26)$$

Substitution of (3.26) in (3.25) yields

$$\begin{aligned} B\underline{u}^{n+1} &= \frac{18}{11}B\underline{u}^n - \frac{9}{11}B\underline{u}^{n-1} + \frac{2}{11}B\underline{u}^{n-2} - \frac{6\Delta t}{11Re}A\underline{u}^{n+1} \\ &+ \frac{6}{11}\Delta t\left(3N_L(\underline{u}^n,t^n) - 3N_L(\underline{u}^{n-1},t^{n-1}) + N_L(\underline{u}^{n-2},t^{n-2})\right). \end{aligned} \quad (3.27)$$

Note that there is no time-splitting error. The local error of this extrapolation scheme is of order three, implying that the BDF3/EX3 scheme has a global error of order three as well, since the nonlinear terms are multiplied by Δt. A simple development in Taylor series reveals the following expression for the local error:

$$\text{LE} = \frac{3}{22}(\Delta t)^4 B^{-1}\left(\frac{d^3N_L(\underline{u}(t),t)}{dt^3} + 3Re^{-1}\frac{d^3A\underline{u}(t)}{dt^3}\right)_{t=t^{n+1}} + O(\Delta t)^5. \quad (3.28)$$

For a converged, stationary solution we have that neither $\underline{u}$, nor the operators A and N_L vary with t, yielding that the local errors (3.20), (3.24), and (3.28) vanish.

3.2.2 Operator-integration-factor splitting

Another way to construct high-order time methods is by applying the operator-integration-factor splitting as discussed by Maday et al. [48]. Timmermans [74] also applied this method in the framework of spectral-element methods for incompressible fluid flow. The advantage of this technique is that it combines any two methods for the linear and nonlinear terms. There are several interpretations of the operator-integration-factor splitting method. The one we present in this section is as a subcycling method. It is convenient to restrict our analysis to the case for which the nonlinear term N_L can be written as $-C(\underline{u}(t))\underline{u}(t)$, with $C(\underline{u}(t))$ a matrix the entries of which depend on $\underline{u}(t)$.

We start by writing Equation (3.17) in terms of an integrating factor $\mathcal{Q}_N^{(t^*,t)}$

$$\frac{\partial}{\partial t}\left(\mathcal{Q}_N^{(t^*,t)} B\underline{u}(t)\right) = -\mathcal{Q}_N^{(t^*,t)} Re^{-1} A\underline{u}(t), \tag{3.29}$$

which is defined by

$$\frac{\partial}{\partial t}\mathcal{Q}_N^{(t^*,t)} B = \mathcal{Q}_N^{(t^*,t)} C(\underline{u}(t)), \qquad \mathcal{Q}_N^{(t^*,t^*)} = I, \tag{3.30}$$

with I the identity matrix and t^* an arbitrary fixed time. The operator-integration-factor splitting proceeds by discretizing (3.29) by an appropriate time scheme. Here, we will apply a BDFs scheme. Equation (3.29) is then rewritten as

$$\beta_s B\underline{u}^{n+1} + \sum_{i=1}^{s} \beta_{s-i} \mathcal{Q}_N^{(t^{n+1},t^{n+1-i})} B\underline{u}(t^{n+1-i}) = -\frac{\Delta t}{Re} A\underline{u}^{n+1}. \tag{3.31}$$

The next step consists of the evaluation of the terms involving the operator-integration factor, which is never constructed explicitly. Instead, we define

$$\mathcal{Q}_N^{(t^{n+1},t^{n+1-i})} B\underline{u}(t^{n+1-i}) = B\tilde{\underline{u}}_i(t^{n+1}), \qquad i = 1,..,s, \tag{3.32}$$

where $\underline{\tilde{u}}_i(t^{n+1})$ is obtained by solving the following IVP :

$$B\frac{\partial \underline{\tilde{u}}_i(t)}{\partial t} = -C(\underline{\tilde{u}}_i(t))\underline{\tilde{u}}_i(t), \qquad \underline{\tilde{u}}_i(t^{n+1-i}) = \underline{u}^{n+1-i}, \qquad t^{n+1-i} \leq t \leq t^{n+1}. \quad (3.33)$$

Problem (3.33) is solved with a step size $\Delta s = \Delta t/M$; M is the number of subcycles and has an important impact on the stability of the scheme, as will be shown later. In this way, the step size for the expensive implicit part is decoupled from the cheap explicit part. Hence, the stability condition for the convective part is on Δs and not on Δt. In the next section, we will study the effect of subcycling on the stability of the complete scheme. Note that in the original paper of Maday et al. [48], the operator-integration-factor splitting method is only applied to linear and linearized operators. It can, however, also readily be applied to a nonlinear operator as is demonstrated in this section. As an example, we give the BDF1/RK4 and the BDF3/RK4 scheme. The BDF1/RK4 scheme reads as follows:

$$B\underline{u}^{n+1} = B\underline{\tilde{u}}_1 - \frac{\Delta t}{Re}A\underline{u}^{n+1}, \quad (3.34)$$

where $\underline{\tilde{u}}_1$ is computed by applying the fourth-order Runge-Kutta method to the following IVP

$$B\frac{\partial \underline{\tilde{u}}_1}{\partial t} = -C(\underline{\tilde{u}}_1(t))\underline{\tilde{u}}_1(t), \qquad t \in [t^n, t^{n+1}], \qquad \underline{\tilde{u}}_1(t^n) = \underline{u}_n. \quad (3.35)$$

The BDF3/RK4 scheme reads

$$B\underline{u}^{n+1} = \frac{18}{11}B\underline{\tilde{u}}_1 - \frac{9}{11}B\underline{\tilde{u}}_2 + \frac{2}{11}B\underline{\tilde{u}}_3 - \frac{6\Delta t}{11Re}A\underline{u}^{n+1}, \quad (3.36)$$

where the fourth-order Runge-Kutta method has been applied to (3.33)

- on the interval $[t^n, t^{n+1}]$, $\underline{\tilde{u}}_1(t^n) = \underline{u}_n$, to compute $\underline{\tilde{u}}_1$
- on the interval $[t^{n-1}, t^{n+1}]$, $\underline{\tilde{u}}_2(t^{n-1}) = \underline{u}_{n-1}$, to compute $\underline{\tilde{u}}_2$
- on the interval $[t^{n-2}, t^{n+1}]$, $\underline{\tilde{u}}_3(t^{n-2}) = \underline{u}_{n-2}$, to compute $\underline{\tilde{u}}_3$.

Again, the step size for these three problems is Δs. Unless stated otherwise, we will use $\Delta s = \Delta t$. In these examples, the fourth-order Runge-Kutta scheme is chosen

to integrate the nonlinear part because of its large stability region along the imaginary axis. Moreover, when combined with a BDFs, the overall order of the scheme is found to be $\min\{4, s\}$ (this has been shown for linearized operators [48], and is verified numerically for the nonlinear convection operator). The local splitting error for this type of methods does not vanish for stationary problems. In order to understand this phenomenon, which is not present for the BDFs/EX3 method, we will analyze the BDF1/RK2 scheme. This scheme combines the second-order Runge-Kutta method and the first-order BDF as follows:

$$B\underline{u}^{n+1} = B\tilde{\underline{u}} - \frac{\Delta t}{Re} A\underline{u}^{n+1} \tag{3.37}$$

$$B\tilde{\underline{u}} = B\underline{u}^n - \Delta t C\left(\underline{u}^n - \frac{\Delta t}{2} C(\underline{u}^n)\underline{u}^n\right)\left(\underline{u}^n - \frac{\Delta t}{2} C(\underline{u}^n)\underline{u}^n\right) \tag{3.38}$$

The local error of this method is given by the expression

$$\mathrm{LE} = \frac{(\Delta t)^2}{2Re} B^{-1} \left\{ \frac{dA\underline{u}(t)}{dt} + A\underline{u}(t) \frac{\partial C(\underline{u}(t))\underline{u}(t)}{\partial \underline{u}} \right\}_{t=t^{n+1}}. \tag{3.39}$$

The first term of (3.39) represents the local time error of the BDF scheme and vanishes for stationary problems; the second term is the splitting error and does not disappear. Remark that the error is proportional to the inverse of the Reynolds number. This is not confirmed by numerical tests applied to methods with fourth-order Runge-Kutta approximations for the nonlinear terms. Probably, the error estimates for these higher-order schemes contain more complex terms that do not depend on the Reynolds number. Because of the complex expressions for the local errors of high-order Runge-Kutta schemes, it is difficult to analyze the BDF1/RK4 or BDF3/RK4 scheme in the same way. However, by means of a numerical test, we will show that the latter scheme has a third-order splitting error that does not vanish for stationary problems. This test will be accomplished in Section 3.3, in which we will also investigate the stability characteristics of different schemes based on extrapolation and operator-integration-factor splitting. First, we will show that the latter method can be interpreted either as subcycling method, or as a method of characteristics.

3.2.3 Analogy between the subcycling method and the method of characteristics

The subcycling method is identical to the method of characteristics (see e.g. Pironneau [61] and Ho et al. [38]), as has been indicated by Maday et al. [48] and by Boukir

[10]. The exact relation between the two methods can be established by writing Equation (3.17) in its Lagrangian form

$$B\frac{\partial}{\partial t}\underline{u}(\underline{X}(\underline{x},t),t) = -\frac{1}{Re}A\underline{u}(\underline{X}(\underline{x},t),t), \tag{3.40}$$

where $\underline{X}(\underline{x},t)$ is the standard Lagrangian spatial variable. For the sake of clarity, we write the velocity field as $\underline{u}(\underline{x},t)$. The left-hand side of Equation (3.40) represents a substantial derivative, that is, a space-time derivative following a material point in the fluid:

$$B\frac{\partial}{\partial t}\underline{u}(\underline{X}(\underline{x},t),t) = B\frac{\partial}{\partial t}\underline{u}(\underline{x},t) + C(\underline{u}(\underline{x},t))\underline{u}(\underline{x},t). \tag{3.41}$$

Equation (3.40) is then discretized by a BDFs:

$$\beta_s B\underline{u}(\underline{X}(\underline{x},t^{n+1}),t^{n+1}) \quad +$$

$$\sum_{i=1}^{s}\beta_{s-i}B\underline{u}(\underline{X}(\underline{x},t^{n+1-i}),t^{n+1-i}) \quad = \quad -\frac{\Delta t}{Re}A\underline{u}(\underline{X}(\underline{x},t^{n+1}),t^{n+1}). \tag{3.42}$$

Following Pironneau [61], $\underline{u}(\underline{X}(\underline{x},t^{n+1-i}),t^{n+1-i})$ is the value of $\underline{u}$ at time t^{n+1-i} at the "foot" of the characteristic the "head" of which at time t^{n+1} is $\underline{x}$. In fact, $\underline{X}(\underline{x},t^{n+1-i})$ is defined by the backward problem

$$\frac{\partial \underline{X}(\underline{x},s)}{\partial s} \quad = \quad \underline{u}(\underline{X}(\underline{x},s),s) \qquad s \in \left[t^{n+1-i},t^{n+1}\right) \tag{3.43}$$

$$\underline{X}(\underline{x},t^{n+1}) \quad = \quad \underline{x}. \tag{3.44}$$

By virtue of the "end-point" condition (3.44), Equation (3.42) simplifies to

$$\beta_s B\underline{u}^{n+1} + \sum_{i=1}^{s}\beta_{s-i}B\underline{u}(\underline{X}(\underline{x},t^{n+1-i}),t^{n+1-i}) = -\frac{\Delta t}{Re}A\underline{u}^{n+1}. \tag{3.45}$$

The foot characteristic values can be determined by solving Equations (3.43) and (3.44) by an appropriate time scheme, for instance RK4, and time step. The equivalence with

the subcycling method is established by integrating Equation (3.41) in time

$$\int_{t^{n+1-i}}^{t^{n+1}} B\frac{\partial}{\partial t}\underline{u}(\underline{X}(\underline{x},t),t)dt = \int_{t^{n+1-i}}^{t^{n+1}} B\frac{\partial}{\partial t}\underline{u}(\underline{x},t)dt + \int_{t^{n+1-i}}^{t^{n+1}} C(\underline{u}(\underline{x},t))\underline{u}(\underline{x},t)dt \quad i=1,..,s, \quad (3.46)$$

which gives

$$B\underline{u}(\underline{X}(\underline{x},t^{n+1-i}),t^{n+1-i}) = B\underline{u}^{n+1-i} - \int_{t^{n+1-i}}^{t^{n+1}} C(\underline{u}(\underline{x},t))\underline{u}(\underline{x},t)dt \quad i=1,..,s. \quad (3.47)$$

By virtue of Equation (3.32), we can set

$$B\underline{u}(\underline{X}(\underline{x},t^{n+1-i}),t^{n+1-i}) = \mathcal{Q}_N^{(t^{n+1},t^{n+1-i})} B\underline{u}(t^{n+1-i}), \quad (3.48)$$

and Equation (3.31) is equivalent to Equation (3.45).

3.3 Accuracy and stability of splitting schemes

In this section we will try to make a comparison between the splitting schemes à la Maday et al. [48] and those that are based on extrapolation. We will both be concerned with accuracy and stability. In particular, we will examine four methods, BDF1/RK4 and BDF1/EX3, as examples of low-order methods, and BDF3/RK4, BDF3/EX3, for the simulation of problems for which high-order temporal accuracy is essential.

3.3.1 Accuracy

We start by comparing the accuracy of three of the aforementioned methods on a steady problem. Let us focus on the maximum errors in the $\underline{u}_1$ velocity (see Figure 3.1) for a three-dimensional Navier-Stokes problem in the cube $[0,1]^3$ with a hole of dimension $[0.4,0.6]\times[0,1]\times[0,4,0.6]$ ($N=5$, $K=8$, $Re=10$), see also Section 4.2, Figure 4.6. The analytical solution of the stationary problem is given by

$$\underline{u}(x_1,x_2,x_3) = \left(-\cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2), \sin(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2), 0\right)^T \quad (3.49)$$

$$p(x_1,x_2,x_3) = 0. \quad (3.50)$$

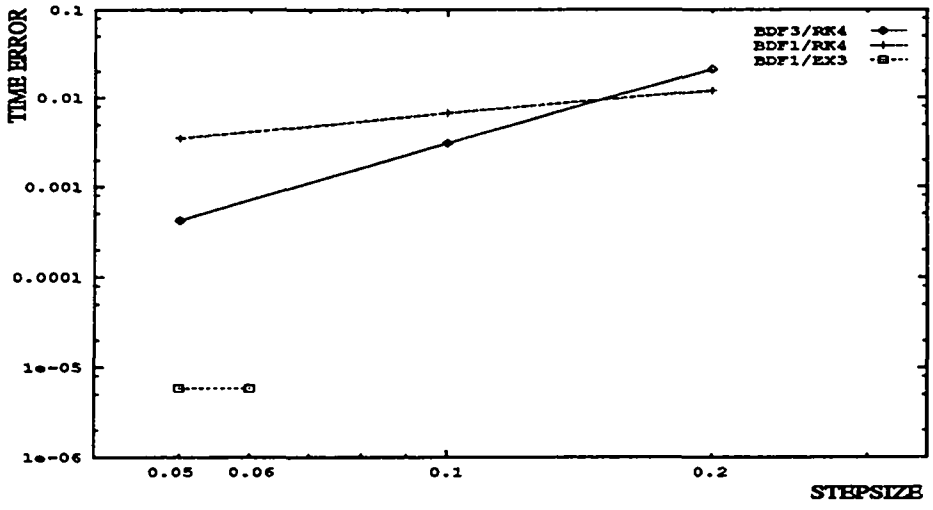

Figure 3.1: **Maximum (time-splitting error) in $\underline{u}_1$ velocity as a function of the step size. Stationary problem. The BDF3/RK4 result at $\Delta t = 0.2$ has been obtained by setting $M = 3$. The BDF1/EX3 has no time-dependent error for stationary problems, but is unstable for $\Delta t > 0.06$.**

We see in Figure 3.1 that the BDF1/RK4 scheme is indeed of order one and that the BDF3/RK4 scheme is of order three. The BDF1/EX3 method has no time-dependent error for stationary problems. This confirms the analytical results of the previous section. The BDF3/EX3 method will show the same behaviour as its first-order analogue BDF1/EX3. Apparently, the third-order method is not behaving optimally due to a very large error constant. This makes the curves for the BDF1/RK4 and BDF3/RK4 schemes intersect for relatively large values of Δt. A confirmation of the large error constant for BDF3/RK4 is found when we make a comparison with BDF3/EX3 on the test problem (3.49), (3.50), where the velocity field is multiplied by $\sin(\pi t)$; $N = 5$, $Re = 100$, $\Delta t = 0.05$. The solution is no longer stationary, so both methods will show time errors. Comparing the maximum errors in the velocity we found 4.6E-4 for BDF3/EX3 and 5.8E-3 for BDF3/RK4. Another disadvantage of the BDF3/RK4 method is that it requires a considerable number of evaluations of the convection operator ($24M$ per time step, versus one for BDF3/EX3). And, indeed, we see a considerable difference in cpu time (more than a factor two for this problem) between the two schemes. This difference is believed to be reduced when the polynomial degree N increases, but is still an argument in favour of the extrapolation scheme. The number of subcycles M has been found to have no effect on the precision.

3.3.2 Stability

The operator-integration-factor splitting method is designed to circumvent the stability condition by using a smaller time step, Δs, for the explicit part. We should not forget, however, that we do not solve two separated problems, allowing an independent analysis of the convective and the diffusive part, but one global problem. Hence, the role of this so-called subcycling method can only be investigated, by also taking the diffusive part into account. In order to analyze the stability for any ratio of convection to diffusion, the classical linear test equation $dy/dt = \lambda y$ is too simple. Therefore, we

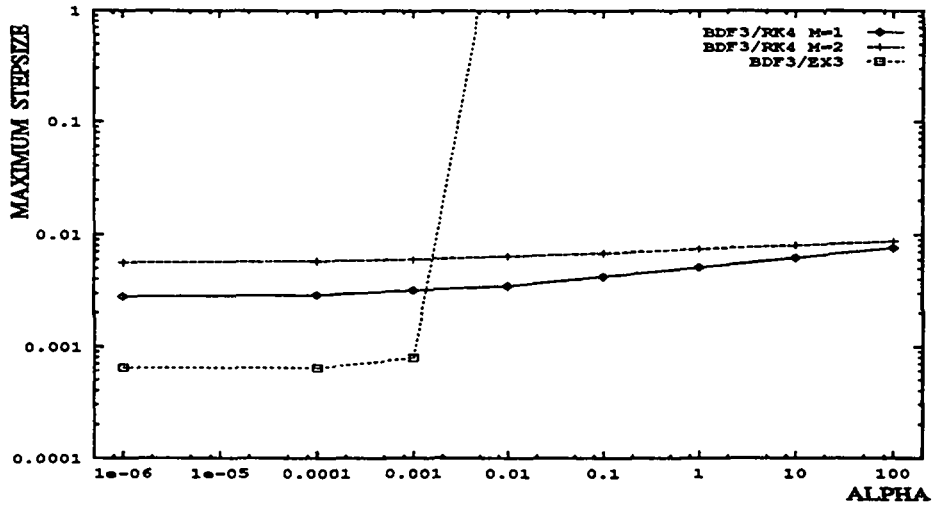

Figure 3.2: Maximum allowed time step Δt as a function of α; $\lambda_1 = -10^6\alpha$ and $\lambda_2 = 10^3 i$, corresponding to $K_1 = 10$ and $N = 10$.

introduce a new scalar test equation

$$\frac{dy}{dt} = \lambda_1 y + \lambda_2 y, \tag{3.51}$$

where λ_1 represents the eigenvalue spectrum of the diffusion operator and λ_2 represents the eigenvalue spectrum of the convection operator. The characteristic polynomials are derived in the same manner as in Section 3.1 and are listed for various methods in the Appendix A. The only difference with a classical stability analysis is that the characteristic polynomials depend on four variables $(\zeta, \lambda_1, \lambda_2, \Delta t)$ instead of three. We recall that for an s-step method, ζ is defined as $\zeta^p = y^{n+1-s+p}$. By definition, a scheme is (strongly) stable for a given λ_1, λ_2 and Δt if all the roots ζ_i of C are in absolute value smaller than one:

$$C(\zeta_i, \lambda_1, \lambda_2, \Delta t) = 0 \Longrightarrow |\zeta_i| < 1. \tag{3.52}$$

In order to perform a relevant analysis, we have to determine estimates for the parameters λ_1 and λ_2. From the previous chapter and from Rønquist [65], we learn that the maximum eigenvalue of the convective operator grows like $O(K_1 N^2 i)$ and that the largest eigenvalue of the diffusive operator grows like $O(K_1^2 N^4)$. Hence, in order to investigate the stability of a spectral element discretization with $N = 10$, $K_1 = 10$, we have to take $\lambda_1 = -10^6$ and $\lambda_2 = 10^3 i$. Then we multiply λ_1 by a factor α to monitor the amount of diffusion in the flow. Large values of α correspond to a Stokes flow and small values of α correspond to a flow at a high Reynolds number.

In Figure 3.2, we give the maximum time step as a function of α for some third-order time-integration schemes. The results have been obtained by evaluating the characteristic polynomials (see Appendix A) using the symbolic-manipulation program Mathematica. We see that, for small values of α, the BDF3/RK4 is significantly

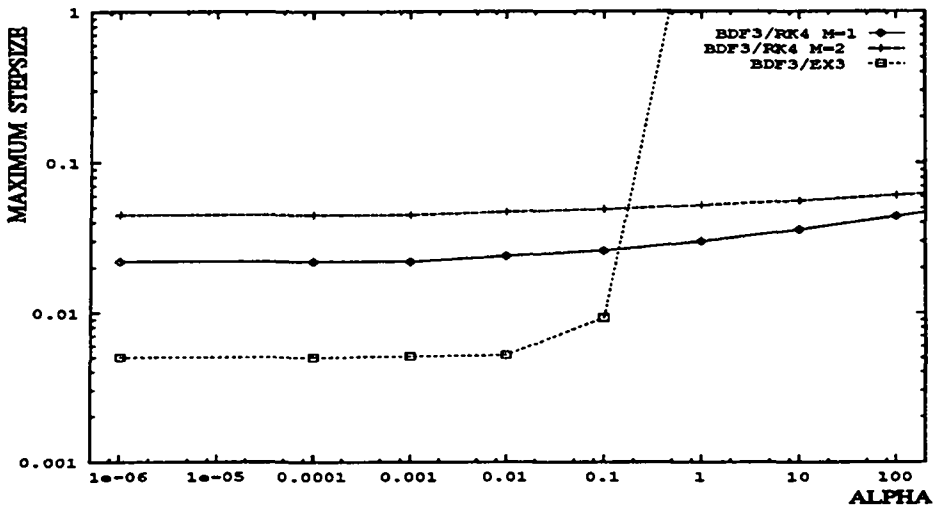

Figure 3.3: **Maximum allowed time step Δt as a function of α; $\lambda_1 = -3125\alpha$ and $\lambda_2 = 125i$, corresponding to $K_1 = 5$ and $N = 5$.**

more stable than the BDF3/EX3, and that the maximum allowed time step for the BDF3/RK4 method doubles when two subcycles are used. The positive effect thanks to multiple subcycles vanishes for larger values of α. This is explained by the fact that for these values of α, the stability is increasingly determined by the BDF. This also accounts for the unconditionally stable behaviour of the BDF3/EX3 scheme once a "critical α" is attained. We remark that we could not get reliable results for $M > 2$, which is not surprising by taking a closer look at the corresponding characteristic polynomials (Appendix A, Equations (A.5) and (A.9)). The BDF3/RK4 ($M = 3$) polynomial for example (Appendix A, Equation (A.9)), requires the computation of $(\lambda_2 \Delta t)^{36}$ and the divisor is of order 10^{26}. For the same reason, we give the results for large α with some precautions.

In Figure 3.3, we give the results of the same test, with a different set of parameters: $\lambda_1 = -3125\alpha$ and $\lambda_2 = 125i$, corresponding to $N = 5$, $K_1 = 5$. Apart from a larger allowed time step, we see the same behaviour as in Figure 3.2. In Figure 3.4, we give the results for the first-order schemes. We have no explanation for the superior behaviour of the $M = 1$ scheme to the $M = 2$ scheme for $\alpha > 1$, nor for the fact that for small values of α, the Δt_{max} for the BDF3/EX3 scheme is 3.5 times as large as the Δt_{max} for the BDF1/EX3 scheme (compare Figures 3.3 and 3.4).

It is important to note that the numbers presented in Figures 3.2, 3.3, and 3.4 are only an approximation of the real situation, where constants in front of the operators and the presence of the pressure might give some different results. However, the following test for the stability of the schemes which is applied to the full Navier-Stokes equations will confirm the trend indicated by the figures.

We have investigated the stability for the Navier-Stokes equations in the cube geometry with a hole ($K = 8$, $N = 5$ analytical solution given by (3.49), (3.50), multiplied by $\sin(\pi t)$) for different values of the Reynolds number. In order to define the maximum allowed time step Δt_{max}, we integrated the problem over a large time interval $[0, 32]$. In Table III, we give the results for $Re = 100$. We see that subcycling im-

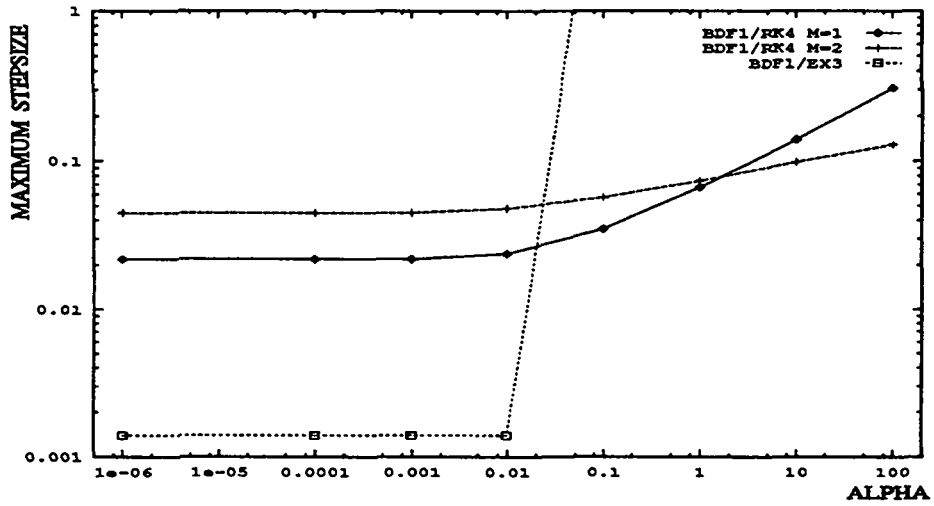

Figure 3.4: **Maximum allowed time step Δt as a function of α; $\lambda_1 = -3125\alpha$ and $\lambda_2 = 125i$, corresponding to $K_1 = 5$ and $N = 5$.**

BDF1/RK4			BDF3/RK4			BDF1/EX3	BDF3/EX3
M=1	M=3	M=5	M=1	M=3	M=5		
$[\frac{32}{140}, \frac{32}{130}]$	$[\frac{32}{70}, \frac{32}{65}]$	$[\frac{32}{60}, \frac{32}{55}]$	$[\frac{32}{225}, \frac{32}{200}]$	$[\frac{32}{165}, \frac{32}{150}]$	$[\frac{32}{165}, \frac{32}{150}]$	$[\frac{32}{925}, \frac{32}{850}]$	$[\frac{32}{425}, \frac{32}{400}]$

Table III: **Interval containing Δt_{max} for $N = 5$, $K = 8$, $Re = 100$, and $T = 32$**

proves considerably the stability and that the extrapolation schemes are inferior to the operator-integration-factor splitting schemes. This could be expected from Figure 3.3 and Figure 3.4, considering low values of α. What can not be explained from these figures is the difference between the BDF1/RK4 and the BDF3/RK4 methods.

In Table IV, we give the results for $Re = 5$, which show a remarkable resemblance with the situation at $\alpha \approx 1$ in Figures 3.3 and 3.4: An increasing value of M has an improving effect on the BDF3/RK4 scheme, but does hardly help the BDF1/RK4 scheme to become more stable; The figures also explain that the first-order schemes are slightly more stable and that the BDF1/EX3 and BDF3/EX3 methods are stable for any value of Δt. Finally, we remark that for $Re = 100$, the BDF3/EX3 scheme allows a much larger time step than the BDF1/EX3 scheme. This phenomenon is also observed by comparing Figures 3.3 and 3.4.

BDF1/RK4			BDF3/RK4			BDF1/EX3	BDF3/EX3
M=1	M=3	M=5	M=1	M=3	M=5		
$[\frac{32}{70}, \frac{32}{65}]$	$[\frac{32}{65}, \frac{32}{60}]$	$[\frac{32}{50}, \frac{32}{45}]$	$[\frac{32}{200}, \frac{32}{175}]$	$[\frac{32}{125}, \frac{32}{120}]$	$[\frac{32}{120}, \frac{32}{110}]$	$[\frac{32}{25}, \infty)$	$[\frac{32}{25}, \infty)$

Table IV: **Interval containing Δt_{max} for $N = 5$, $K = 8$, $Re = 5$, and $T = 32$**

3.3.3 Closing remarks

In the previous tests, the nonlinear term has been discretized in its convective form. As has been demonstrated in Section 2.7, this might lead to some appreciable real parts in the eigenvalues for a purely convective problem. We will now make clear that this does not lead to instabilities if (very small) diffusive effects are present in the flow. To this end, we return to the stability analysis performed in the previous section. We fix $\alpha = 10^{-3}$ and look for the corresponding Δt_{max} for BDF1/RK4 and BDF1/EX3 in Figure 3.4. We then took $\lambda_1 = -3125 \cdot 10^{-3}$ and replaced $\lambda_2 = 125i$ by $\lambda_2 = \beta i + 1$. In other words, the stability is investigated for the case where the eigenvalues of the convective operator are not imaginary, but complex with a small real part. We have found that both schemes are still stable for $0 \leq \beta < 122$, which is very close to $0 \leq \beta < 125$ for the unperturbed case. It is important that no instabilities have been found for small values of β. Similar tests with different values of the parameters α, λ_1, λ_2, and a different size of the perturbation confirmed the conclusion that the convection form can be used without any danger, provided that the flow is not purely convective.

By choosing for the convective form, the evaluation of the nonlinear term is still not fixed: One can either linearize this term or not. Since the operator is evaluated in an explicit manner, there is no *need* to linearize. Nevertheless, it is suggested in [48] to do so. Let us illustrate this phenomenon by the computation of

$$\underline{k}_2 = -C(\underline{u}^n + \frac{\Delta t \underline{k}_1}{2})(\underline{u}^n + \frac{\Delta t \underline{k}_1}{2}), \tag{3.53}$$

as occurs in the RK4 method (3.16). If we would have linearized $C(\underline{u})\underline{u}$, Equation (3.53) has a different form:

$$\underline{k}_2 = -C\left(\underline{u}(t^n + \frac{\Delta t}{2})\right)(\underline{u}^n + \frac{\Delta t \underline{k}_1}{2}), \tag{3.54}$$

where $\underline{u}(t^n + \Delta t/2)$ is approximated by an extrapolation formula of a suitable order, that uses previously computed values of $\underline{u}$, for example:

$$\underline{u}(t^n + \tau\Delta t) = \underline{u}^n + O(\tau\Delta t) \qquad \text{or} \tag{3.55}$$

$$\underline{u}(t^n + \tau\Delta t) = (1+\tau)\underline{u}^n - \tau\underline{u}^{n-1} + O(\Delta t\tau)^2 \qquad \text{or} \tag{3.56}$$

$$\underline{u}(t^n + \tau\Delta t) = \frac{1}{2}(\tau^2 + 3\tau + 2)\underline{u}^n - (\tau^2 + 2\tau)\underline{u}^{n-1} + \frac{1}{2}(\tau^2 + \tau)\underline{u}^{n-2} + O(\Delta t\tau)^3. \tag{3.57}$$

A priori, there is no obvious reason to choose for one of the two options, although the non-linearized form (3.53) seems more logical. All the tests in this chapter have been performed by using (3.53).

Let us investigate if the second option (3.54) is more stable. First, we remark that when equation (3.57) is used for linearization of the BDFs/EX3 method, we obtain an equivalent scheme to the non-linearized one. (The extrapolation procedure requires only values at the step points.) For the BDFs/RK4 schemes, however, we obtain different formulations. In Table V, we show the results for the linearized method BDF3/RK4 ($N = 5$, $K = 8$, $Re = 100$). As compared to the results of the non-

M=1	M=3	M=5
$[\frac{32}{250}, \frac{32}{225}]$	$[\frac{32}{100}, \frac{32}{90}]$	$[\frac{32}{100}, \frac{32}{90}]$

Table V: **Interval containing Δt_{max} for $N = 5$, $K = 8$, $Re = 100$, and $T = 32$. BDF3/RK4 method, where the nonlinear term is first linearized, and then evaluated.**

linearized formulation (see Table III), we see that the scheme based on the linearized convection operator is more stable for $M > 1$.

The numerical tests show that the value of the maximum allowed time step does not grow any more for $M > M_0$, with M_0 typically equal to 5. This seems to correspond to the analysis of time integration by the method of characteristics, which is another interpretation of the operator-integration-factor splitting scheme. This study has been performed by Boukir [10] who showed that a second stability condition emerges which leads to an additional restriction on Δt_{max}.

Summarizing, we can say that the choice between a splitting scheme based on extrapolation and subcycling is in fact a choice between accuracy and stability, respectively. The former method has the advantage that it does not possess a time error for steady problems. Moreover, when the problem is unsteady it leads to much more accurate results. The subcycling method has very attractive stability properties. By augmenting the number of subcycles, the maximum time step can be increased, reducing the number of implicit relations to solve in a given time interval. The stability region can even be augmented by linearizing the nonlinear operator.

In this chapter, we developed time-integration methods up to an order of accuracy of three. The extension to fourth-order accurate methods is not difficult: A fourth-order extrapolation scheme can easily be combined with a BDF4. The operator-integration-factor splitting can be made fourth-order by combining a BDF4 with a RK4. The stability regions of these methods are difficult to predict, but can be determined by the technique which is proposed in this section.

Chapter 4

Fast Helmholtz solvers

In this chapter we focus on the fast solution of the Helmholtz problem

$$Hu = Bf, \tag{4.1}$$

with

$$H = \nu A + \Delta t^{-1} B. \tag{4.2}$$

We refer to Chapter 2 for notations. When solving an unsteady, 3D Navier-Stokes problem, a Helmholtz equation has to be solved for each velocity component. If the Uzawa decoupling method (see Section 2.6) has been used, three Helmholtz equations have also to be inverted in each pressure iteration. Hence, the cpu time required for a (Navier-)Stokes computation depends linearly on the speed of the Helmholtz solver.

When an iterative method is used to solve the Helmholtz problem, preconditioning is essential. Common preconditioners are the diagonal, incomplete LU and finite-elements. The latter method is very effective when the finite element system can be solved by a direct method. Unfortunately, this is often not possible in three dimensions. The incomplete LU preconditioner does not parallelize well. Rønquist [64] proposed recently a preconditioning technique based on deflation. This technique will be further discussed in the context of the preconditioning of the pressure operator (see Chapter 6 or Rønquist [63]). Let us investigate the efficiency of the diagonal preconditioner by introducing $x_l = \nu^{-1}\Delta t^{-1}$. It is clear that for large values of x_l (as is the case for Navier-Stokes equations at moderate or large Reynolds numbers), the Helmholtz operator (4.2) is dominated by the diagonal part B. Consequently, the preconditioner P, $P^{-1} = \text{diag}(H)^{-1}$, is expected to be efficient. This is verified in Figure 4.1 and Figure 4.2 for one-dimensional operators. The condition number is displayed as a function of the polynomial degree N and the number of spectral elements K. Two values of x_l have been chosen, $x_l = 0$ and $x_l = 1000$. It can be observed that the diagonal preconditioner is much more effective for the latter case. Note also that for $x_l = 0$ (not preconditioned) the condition number grows more or less like K^2N^3. So especially for small values of x_l (i.e. when a steady Stokes problem is solved by an unsteady solver), an alternative technique has to be developed. First, we will propose a fast, fully direct method which is limited to parallelepipedic geometries (or: "boxes"). Then, in Section 4.2, a semi-direct Schur complement method is introduced for geometries that are decomposable in boxes. Finally, a general method is discussed

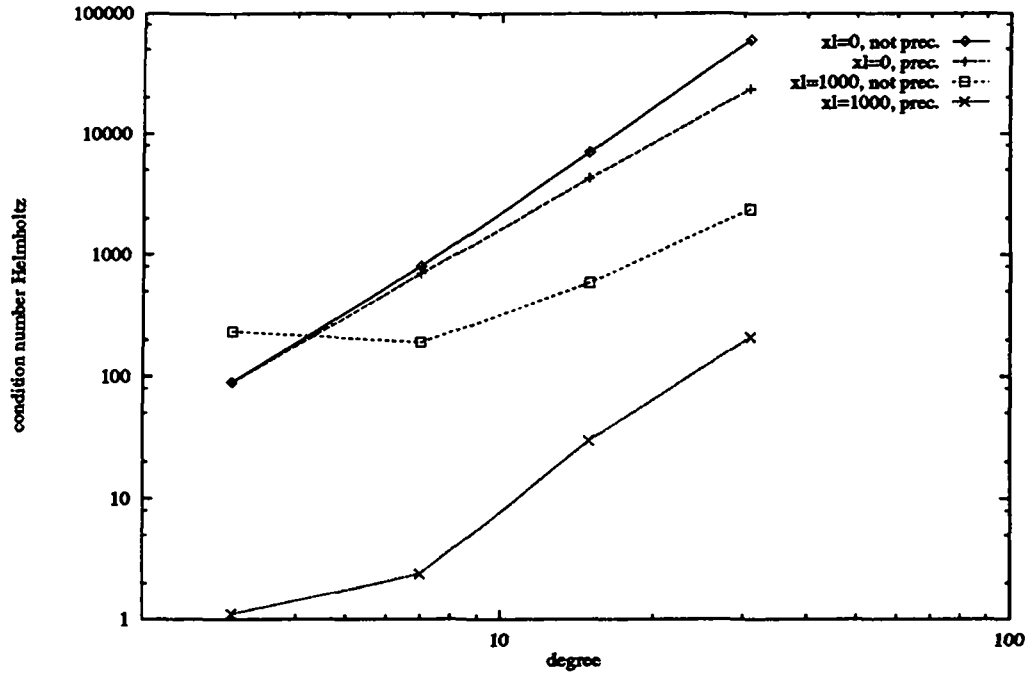

Figure 4.1: Log-log plot of the condition number of the Helmholtz operator as a function of the polynomial degree N. The number of spectral elements is fixed at $K = 8$ and the value for x_l is either $x_l = 0$ or $x_l = 1000$. Both the condition numbers of the preconditioned and the non-preconditioned operators are given.

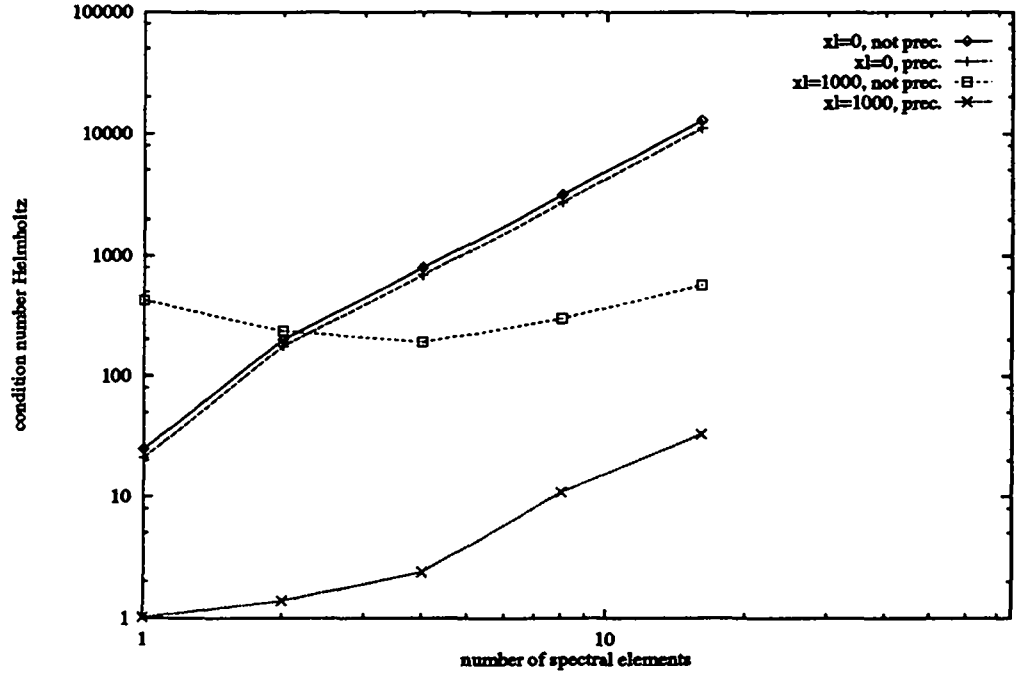

Figure 4.2: Log-log plot of the condition number of the Helmholtz operator as a function of number of spectral elements. The number of polynomial degree N is fixed at $N = 7$ and value for x_l is either $x_l = 0$ or $x_l = 1000$. Both the condition numbers of the preconditioned and the non-preconditioned operators are given.

in Section 4.3, based on preconditioning of the Helmholtz operator by the "incomplete Schur" operator.

4.1 Fast solvers for parallelepipedic geometries

We consider a parallelepipedic, three-dimensional geometry Ω decomposed in parallelepipedic spectral elements $\{\Omega_k\}_{k=1}^K$. For this case, a fast, direct Helmholtz solver has been developed by Couzy and Deville [18] and by Magère [51]. The two methods rely on the same principle and are almost identical. For pedagogic reasons we will here explain the inversion algorithm along the same lines as in [51]. To this end, we introduce the following definition. *A three-dimensional operator L is separable if and only if it can be written in the following form:*

$$L = I \otimes I \otimes L_x + I \otimes L_y \otimes I + L_z \otimes I \otimes I, \tag{4.3}$$

with $A \otimes B \otimes C$ the tensor product of the one-dimensional matrices A, B, and C, and I the one-dimensional identity matrix. Lynch et al. [44] showed that the inverse of an inversible, separable operator L can be written as

$$L^{-1} = P_z \otimes P_y \otimes P_x (I \otimes I \otimes \Lambda_x + I \otimes \Lambda_y \otimes I + \Lambda_z \otimes I \otimes I)^{-1} P_z^{-1} \otimes P_y^{-1} \otimes P_x^{-1}. \tag{4.4}$$

The matrices P_x, P_y and P_z are the matrices involving the eigenvector decomposition of the one-dimensional operators L_x, L_y and L_z, such that

$$P_x^{-1} L_x P_x = \Lambda_x, \qquad P_y^{-1} L_y P_y = \Lambda_y, \qquad P_z^{-1} L_z P_z = \Lambda_z, \tag{4.5}$$

with Λ_x, Λ_y and Λ_z diagonal matrices with as entries the eigenvalues of the corresponding operators. The interest of this method, which is often called the fast diagonalization method (FDM), becomes clear when the cost of an evaluation of $L^{-1}v$ is compared to the matrix-vector multiplication Lv, the basic ingredient of any iterative method. Using (4.4), $L^{-1}v$ requires $6N^4 + N^3$ multiplications, whereas Lv takes $3N^4$ multiplications. So the inverse is computed at the price of two matrix-vector products. Many authors have used this fast diagonalization technique in the context of spectral methods, e.g. [37], [36], and [71]. All these papers are based on mono-element computations. The Helmholtz equation (4.1) is not immediately separable, due to the presence of the weights and the geometric variables l_x, l_y, and l_z: By studying Equations (2.51) and (2.52), it is clear that the derivative with respect to, for example, x depends on y and z via the weights and l_y and l_z. This prohibits the application of (4.4). Fortunately, there is a simple way to overcome this problem. Let us define the mono-dimensional matrices D_{xx}^k, D_{yy}^k, D_{zz}^k, B_x^k, B_y^k, and B_z^k on an arbitrary element Ω_k as

$$(D_{xx})_{\alpha,l}^k = \frac{2\nu}{l_x^k} \sum_{q=0}^{N} D_{q\alpha} \rho_q D_{ql} + \frac{l_x^k}{2\Delta t} \rho_l \quad \alpha, l = 0, .., N$$

$$(D_{yy})_{\beta,m}^k = \frac{2\nu}{l_y^k} \sum_{q=0}^{N} D_{q\beta} \rho_q D_{qm} + \frac{l_y^k}{2\Delta t} \rho_m \quad \beta, m = 0, .., N$$

$$
\begin{aligned}
(D_{zz})^k_{\gamma,n} &= \frac{2\nu}{l^k_z}\sum_{q=0}^{N} D_{q\gamma}\rho_q D_{qn} + \frac{l^k_z}{2\Delta t}\rho_n \quad \gamma,n = 0,..,N \\
(B_x)^k_{\alpha,l} &= \delta_{\alpha l}\rho_l \frac{l^k_x}{2} \quad \alpha,l = 0,..,N \\
(B_y)^k_{\beta,m} &= \delta_{\beta m}\rho_m \frac{l^k_y}{2} \quad \beta,m = 0,..,N \\
(B_z)^k_{\gamma,n} &= \delta_{\gamma n}\rho_n \frac{l^k_z}{2} \quad \gamma,n = 0,..,N.
\end{aligned} \tag{4.6}
$$

In this way, we find on a arbitrary element Ω_k that the Helmholtz operator H^k and the mass matrix B^k can be written as

$$
\begin{aligned}
H^k &= B^k_z \otimes B^k_y \otimes D^k_{xx} + B^k_z \otimes D^k_{yy} \otimes B^k_z + D^k_{zz} \otimes B^k_y \otimes B^k_z \\
B^k &= B^k_z \otimes B^k_y \otimes B^k_z.
\end{aligned} \tag{4.7}
$$

The Helmholtz equation (4.1) can then be written as

$$
(B_z \otimes B_y \otimes D_{xx} + B_z \otimes D_{yy} \otimes B_x + D_{zz} \otimes B_y \otimes B_x)u = B_z \otimes B_y \otimes B_x f. \tag{4.8}
$$

Two remarks have to be made with respect to (4.8). First, boundary equations have been eliminated and direct stiffness is applied to the mono-dimensional operators. In order to simplify notations, we dropped the summation sign $\sum_{k=1}^{K}{}'$ and the superscript k in Equation (4.8). Second, redundant equations, which are present in the original formulation due to direct stiffness (and do not harm an iterative method), have been removed. This induces the necessity of a transformation to a global enumeration of the velocity variables. More precisely, interface variables that indicate the same physical point are eliminated, except one. We proceed by multiplying both sides of Equation (4.8) by $B_z^{-1} \otimes B_y^{-1} \otimes B_x^{-1}$ to arrive at

$$
(I \otimes I \otimes B_x^{-1}D_{xx} + I \otimes B_y^{-1}D_{yy} \otimes I + B_z^{-1}D_{zz} \otimes I \otimes I)u = f. \tag{4.9}
$$

It is clear that the operator is now inversible and separable and can be inverted by the FDM.

We now investigate the cpu time to compute a steady Stokes flow with solution (2.86), (2.87) for two geometries displayed in Figure 4.3 and Figure 4.4. After twenty time steps ($\Delta t = 0.25$) a steady solution has been obtained. Table I and II show the computation times for the two different methods, measured on a Data General Aviion machine (performance of 1 Mflops on a LINPACK benchmark problem).

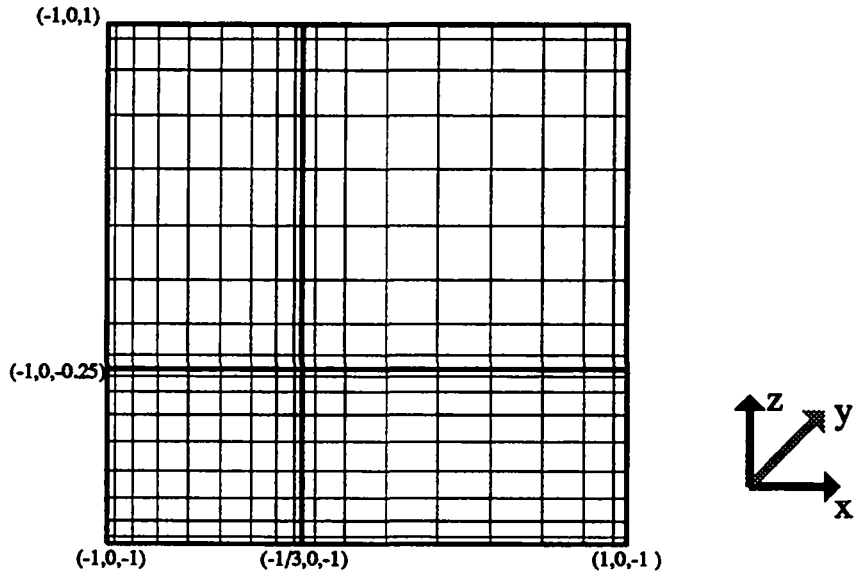

Figure 4.3: **Geometry consisting of four spectral elements. Projection in y-direction, y between -1 and 1.**

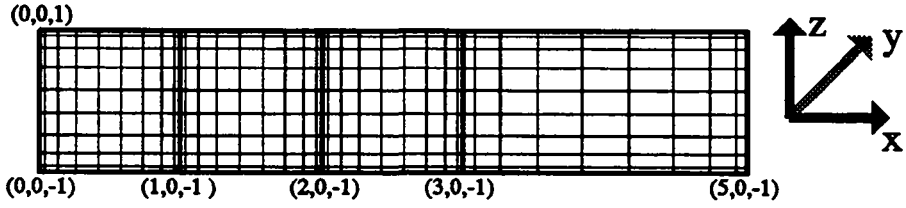

Figure 4.4: **Geometry consisting of four spectral elements. Projection in y-direction, y between -1 and 1.**

	Iterative Helmholtz			Helmholtz by FDM		
N	4	6	9	4	6	9
Seconds	437	2328	12080	54	205	945

Table I: **Geometry as in Figure 4.3. Timings in seconds for 20 time steps ($\nu = 1$, $\Delta t = 0.25$, tolerance for the iterative Helmholtz solver is 10^{-12}). Increasing degree N of polynomial approximation.**

	Iterative Helmholtz			Helmholtz by FDM		
N	4	6	9	4	6	9
Seconds	427	2236	12145	73	275	1181

Table II: **Geometry as in Figure 4.4. Timings in seconds for 20 time steps ($\nu = 1$, $\Delta t = 0.25$, tolerance for the iterative Helmholtz solver is 10^{-12}). Increasing degree N of polynomial approximation.**

We remark that the initialization time for the FDM, i.e. the time to compute the eigenvalue/eigenvector decomposition, is always negligible (less than one second). A comparison of Table I and Table II shows that the current implementation of the FDM is faster for Geometry 4.3 than for Geometry 4.4. Comparing the new algorithm to the iterative one (preconditioned by the diagonal of the Helmholtz operator), an impressive acceleration is obtained. As expected, this speed-up increases for large N.

As will be shown in the next section, the acceleration is less important for small values of $\nu\Delta t$, because of the increasing efficiency of the diagonal preconditioner. We recall that the application of the FDM method is confined to parallelepipedic geometries because of the condition that the Helmholtz operator should be separable. There are, however, some very interesting problems, like turbulence in square cavities or ducts that fall within this class. Often, mono-domain spectral methods using FDM are applied to solve these flows. This might lead to over-accurate results in some parts of the domain. The extension of FDM to spectral element discretizations could alleviate this problem by local refinements and lead to more acceptable values of the condition numbers for the pressure operator. In the next section the extension of the FDM to geometries that are decomposable in parallelepipeds is presented.

The following section appeared as a paper in the Journal of Computational Physics, Vol. 116, 1995, pp 135-142.

4.2 A fast Schur complement method for the spectral element discretization of the incompressible Navier-Stokes equations

W. Couzy and M.O. Deville
Applied Mechanics
Catholic University of Louvain
Louvain-la-Neuve, Belgium.

Classification: 65N30, 76D05

Abstract: The weak formulation of the incompressible Navier-Stokes equations in three space dimensions is discretized with spectral element approximations and Gauss-Lobatto-Legendre quadratures. The Uzawa algorithm is applied to decouple the velocities from the pressure. The equation that results for the pressure is solved by an iterative method. Within each pressure iteration, a Helmholtz operator has to be inverted. This can efficiently be done by separating the equations for the interior nodes from the equations at the interfaces, according to the Schur method. Fast diagonalization techniques are applied to the interior variables of the spectral elements. Several ways to deal with the resulting interface problem are discussed. Finally, a comparison is made with a more classical method.

4.2.1 Introduction

In the past years, the spectral element discretization of the 3-D Navier-Stokes equations has received considerable attention [29, 26, 42, 47, 65]. The advantages of this method are numerous. The high degrees of the approximating polynomials combined with high-order quadrature rules yield accurate solutions. In comparison with more classical discretization methods, a low number of degrees of freedom is needed for a prescribed level of accuracy. The clustering of the grid points close to the boundary, which is typical for many spectral methods, makes the method attractive for flows dominated by boundary-layer dynamics. The decomposition of the domain into several subdomains (the spectral elements) ensures geometrical flexibility and a natural implementation on parallel computers [29, 26].

There are many ways to uncouple the velocities from the pressure. Karniadakis et al. [42] proposed a high-order splitting method, where the pressure is computed by a Poisson equation with compatible boundary conditions. Another way of dealing with this problem is the Uzawa technique [2] which is in fact a Gaussian elimination method by block. An advantage of this approach is that the resulting system, which consists of four positive (semi-) definite symmetric systems (one for the pressure and three for the velocities), is equivalent to the original coupled set of equations. Hence, the system is determined by velocity boundary conditions only and no additional conditions for the pressure are needed. Usually, the four systems are solved by the Preconditioned Conjugate Gradient Method (PCGM), an efficient iterative method for symmetric systems of equations. One of the attractive properties of the PCGM is that the matrix system, which would take $O(K_e N^6)$ memory positions, is never built up explicitly (K_e corresponds to the number of subdomains and N is the polynomial degree in one space dimension). Moreover, tensor products reduce the bulk of the work, which consists in the computation of matrix-vector products, to $O(K_e N^4)$. A disadvantage of the Uzawa method, however, is the high cost required to compute the pressure. Since the pressure matrix contains the inverse of a Helmholtz operator, a classical implementation requires two *nested* PCGMs, resulting in large computation times. One way to deal with this problem is operator splitting [48]. This method seems to work very well in practice, although until now the consistency has not been proven for increasing order of the time scheme.

This paper deals with another approach, first proposed by Patera [58], that reduces rigorously the time to invert the Helmholtz operator and hence the time to compute the pressure. To this end, the Schur complement method is used to separate the Helmholtz equations at the interior nodes of each element from those at the inter-element interfaces. This leads to a set of independent subproblems which is, in general, easier to solve than the original global problem: Iterative methods tend to converge faster due to the locally reduced number of variables and the absence of inter-element coupling. Direct methods can also be considered to invert the local Helmholtz operators. In this paper, we will restrict ourselves to geometries consisting of non-deformed spectral elements. In this case, a fast direct method [44] based on tensor products of the eigenvalue/eigenvector decomposition of the one-dimensional operators is applied to the interior nodes. This direct method, which we will often refer to as the diagonalization method, is very fast; the inverse is computed at the price of two PCGM iterations.

It is not surprising that the fast diagonalization method (FDM) is often used in the context of spectral methods (see for example [58, 71]), where iterative methods tend to converge slowly, due to a large value of N and ill-conditioned matrices. The dimension of the corresponding Schur matrix is less by one than the dimension of the original system, since it involves the interface variables only. Therefore, we might consider classical direct methods as well as iterative methods. In the latter case, the conjugate gradient method can be efficiently preconditioned by block Jacobi. Moreover, an impressive improvement on vector computers can be obtained when the Schur matrix is constructed explicitly. In this way, we avoid the evaluation of tensor products, which is inefficient in terms of vectorization. Independently of the solution method for the Schur matrix, we found that the new algorithm is an order of magnitude faster than the classical one.

The outline of this paper is as follows. First, in section 4.2.2, we briefly present the spectral discretization of the Navier-Stokes equations. In section 4.2.3 we discuss the FDM and in section 4.2.4 we treat the Schur method for a particular problem. Section 4.2.5 will deal with two test problems and comment on the parallelization of the method.

4.2.2 Derivation of the discrete equations

The 3-D incompressible Navier-Stokes equations are discretized by the spectral element method. For more details about the material of this section, we refer the reader to the article of Maday and Patera [47] and to the lecture series by Rønquist [65]. The Navier-Stokes problem is formulated as follows: Find velocities $\underline{u}$ and pressure p in a domain $\Omega \subset \mathcal{R}^3$ such that

$$\frac{\partial \underline{u}}{\partial t} - Re^{-1}\Delta \underline{u} + \underline{u} \cdot \nabla \underline{u} + \nabla p = \underline{b}, \tag{4.10}$$

$$-\mathrm{div}\underline{u} = 0, \tag{4.11}$$

with $t \in [0, T_{\mathrm{end}}]$. At the boundary $\partial\Omega$ of the domain Ω, we impose Dirichlet boundary conditions for the velocity:

$$\underline{u} = \underline{g} \qquad \text{on } \partial\Omega. \tag{4.12}$$

Here, $Re = UL/\nu$ is the Reynolds number based on a characteristic velocity, length and kinematic viscosity. Furthermore, $\underline{b}$ is a force vector and $\underline{g}$ contains the Dirichlet boundary conditions. The extension to Neumann or mixed boundary conditions is straightforward. As a starting point for the spectral element discretization, we use the variational equivalences of (4.10 - 4.11):
Find $(\underline{u}, p)$ in $X_g \times M$ such that $\forall \underline{w} \in X_0, \forall q \in M$

$$\left(\frac{\partial \underline{u}}{\partial t}, \underline{w}\right) + Re^{-1}\,(\nabla \underline{u}, \nabla \underline{w}) + (\underline{u} \cdot \nabla \underline{u}, \underline{w}) - (p, \mathrm{div}\underline{w}) \;=\; (\underline{b}, \underline{w}) \tag{4.13}$$

$$-\,(q, \mathrm{div}\underline{u}) \;=\; 0, \tag{4.14}$$

where

$$\forall \phi, \psi \in \mathcal{L}^2(\Omega) \qquad (\phi, \psi) = \int_\Omega \phi(\underline{x})\psi(\underline{x})d\underline{x}, \qquad \underline{x} \in \Omega. \tag{4.15}$$

The space $\mathcal{L}^2(\Omega)$ is the space of all square integrable functions over Ω. The space X_g for the velocity, the space X_0 for the test functions and the space M for the pressure are defined as follows:

$$X_g \;=\; \{\mathrm{v} \in |\mathcal{H}^1(\overline{\Omega})|^3,\ \mathrm{v\ satisfies\ boundary\ conditions}\} \tag{4.16}$$
$$X_0 \;=\; \{\mathrm{v} \in |\mathcal{H}^1(\overline{\Omega})|^3,\ \mathrm{v\ vanishes\ at}\ \partial\Omega\} \tag{4.17}$$

$$M \;=\; \mathcal{L}^2_0(\Omega) = \{\phi \in \mathcal{L}^2(\Omega);\quad \int_\Omega \phi(\underline{x})dx = 0\} \tag{4.18}$$

Expression (4.18) should be interpreted as an averaging procedure for the pressure. $\mathcal{H}^1(\Omega)$ is the space of all square integrable functions whose first-order derivatives are also square integrable over Ω.

The first step in the discretization process is to subdivide the domain $\overline{\Omega} = \partial\Omega \cup \Omega$ into K_e non-overlapping rectilinear elements $\overline{\Omega}_k$ such that the intersection of two or more neighboring elements is either a face, an edge or a vertex. In order to simplify the notation, we assume that the number of nodes N is equal in each direction and on each element. Of course, this does not affect the general concept. Next, we have to define the discrete polynomial subspaces $X_{g,h} \subset X_g$ and $M_h \subset M$, in which the velocities and pressure will be approximated respectively. In order to avoid spurious pressure modes, Maday and Patera [47] and Bernardi and Maday [7] proposed the use of the following subspaces:

$$X_{g,h} \;=\; X_g \cap \mathcal{P}^3_{N,K_e}(\Omega) \tag{4.19}$$
$$M_h \;=\; M \cap \mathcal{P}_{N-2,K_e}(\Omega), \tag{4.20}$$

with $\mathcal{P}_{N,K_e} = \{\phi \in \mathcal{L}^2(\Omega);\, \phi_{|\Omega_k}$ is a polynomial of degree less than or equal to $N\}$. Consequently, the space $X_{0,h}$ is defined as

$$X_{0,h} \;=\; X_0 \cap \mathcal{P}^3_{N,K_e}(\Omega). \tag{4.21}$$

The choice for the spaces (4.19-4.20) implies the introduction of staggered grids. In our case, the velocities will be approximated on a Gauss-Lobatto-Legendre grid, whereas the pressure will be approximated on a Gauss-Legendre grid. Furthermore, the velocities are continuous along the element boundaries (while the pressure is *not* necessarily continuous).

The spectral element discretization proceeds by the application over each subdomain of two Gaussian integration rules, corresponding to the two grids mentioned above. We should remark, that the three-dimensional rules are obtained by tensor products of the one-dimensional formulas.

The final step consists in the discretization in time. In this paper we confine ourselves to a simple time scheme. We choose Backward Euler for the viscous term, where the convective terms at time t_{n+1} are approximated by an explicit time-integration method. The third-order explicit Adams-Bashforth scheme has been applied for its advantageous stability characteristics; the overall order is one. We write the discrete equations immediately in matrix notation, where we use the same symbols as in [47] and [65]

$$\frac{1}{\Delta t} B \underline{u}_i^{n+1} + Re^{-1} A \underline{u}^{n+1} - D_i^T p^{n+1} = B \underline{f}_i^{n+1} \tag{4.22}$$

$$- D_i \underline{u}_i^{n+1} = 0, \qquad i = 1,2,3. \tag{4.23}$$

Here, B is the diagonal mass matrix, A is the discrete Laplace operator, D_i is the discrete divergence operator and the superscript T indicates the transpose. The right-hand-side vector $\underline{f}^{n+1}$ contains the volume force $\underline{b}^{n+1}$ and the explicit terms.

The Uzawa algorithm is applied to uncouple the velocities from the pressure. This technique was originally designed for finite element computations, but is nowadays used in spectral element computations as well (see [47],[65]). The attractive property of the Uzawa method is that the uncoupled system, which consists of four positive (semi-) definite, symmetric sets of equations, is equivalent to the original system. Starting from the discrete equations (4.22-4.23), the Uzawa algorithm is obtained by multiplication of the momentum equations by $D_i H^{-1}$, with

$$H = (Re^{-1} A + \Delta t^{-1} B). \tag{4.24}$$

We obtain

$$- D_i H^{-1} D_i^T p^{n+1} = D_i H^{-1} B \underline{f}_i^{n+1} \tag{4.25}$$

$$H \underline{u}_i^{n+1} = D_i^T p^{n+1} + B \underline{f}_i^{n+1} \qquad i = 1,2,3. \tag{4.26}$$

Equation (4.25) is solved by the PCGM. Clearly, the Helmholtz operator H has to be inverted within each PCGM-iteration. This can efficiently be done by the Schur

complement method in combination with the FDM, which will be discussed in the following sections.

4.2.3 The fast diagonalization method

For a tensorizable and separable operator it is possible to explicitly construct an inverse having a similar tensor product structure. Under certain conditions, which we will discuss later, H is such an operator. According to Lynch et al. [44], we can write

$$H^{-1} = P_z \otimes P_y \otimes P_x (I \otimes I \otimes \Lambda_x + I \otimes \Lambda_y \otimes I + \Lambda_z \otimes I \otimes I)^{-1} P_z^{-1} \otimes P_y^{-1} \otimes P_x^{-1}. \quad (4.27)$$

Here, $A \otimes B$ denotes the tensor product of A and B, and Λ_x, Λ_y and Λ_z are the diagonal matrices of eigenvalues arising from the (1-D) generalized eigenvalue problems $Hu = \lambda u$ in x-, y- and z-direction. The matrices P_x, P_y and P_z are the corresponding matrices containing the eigenvectors. The interest of this method becomes clear when the cost of an evaluation of $H^{-1}\underline{v}$ is compared to the matrix-vector multiplication $H\underline{v}$, the basic ingredient of any iterative method. Using (4.27), $H^{-1}\underline{v}$ requires $6N^4$ multiplications, whereas $H\underline{v}$ takes $3N^4$ multiplications. So the inverse is computed at the price of two matrix-vector products.

Many authors have used this fast diagonalization technique in the context of spectral methods, e.g. [36], [37]. Streett and Hussaini [71] describe the application of this method to the Uzawa technique. All these papers are based on mono-domain computations. On the first hand this restriction seems obvious, since the condition that the operator should be separable, does not allow non-rectangular geometries. If the interface and boundary variables, however, are eliminated, we can use the FDM for the interior nodes of each spectral element, *provided that these elements are rectangular.* Note that, in fact, H is not a separable operator due to a multiplication by the weights. Therefore, we solve the equivalent problem $(HB^{-1})B\underline{u} = \underline{f}$ instead. To avoid complex notation, this diagonal shift is not explicitly represented in this paper. In the next paragraph we will describe the Schur complement method which separates the interior variables from the interface variables. In this way, we can take full advantage of the FDM.

4.2.4 Schur complement method

Construction of the Schur complement

The use of the Schur method is a common practice in modern numerical mechanics. The reduction of the problem to a set of subproblems often leads to memory savings and faster algorithms. Moreover, these subproblems can be solved independently, leading to a high degree of parallelism. Another argument is that different solvers can be applied to the interior and to the interface variables. In our case, the latter reason is the most important, although we will also discuss a parallel implementation in paragraph 4.2.6.

In order to explain the Schur method, we consider a domain as depicted in Figure 4.5. The parallelepipedic domain $\overline{\Omega}$ is decomposed in four spectral elements $\overline{\Omega}_1, \overline{\Omega}_2, \overline{\Omega}_3$

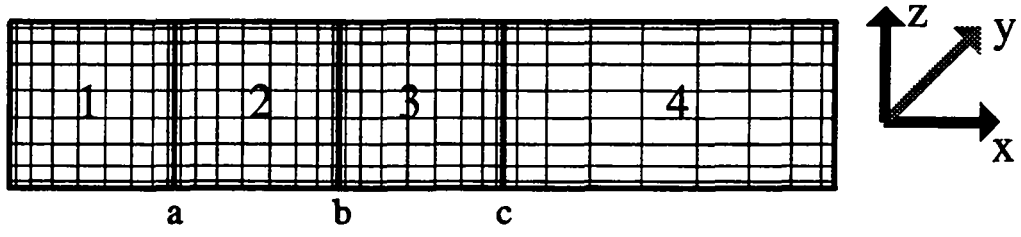

Figure 4.5: **Geometry consisting of four spectral elements. Projection in y-direction (y=1).**

and $\overline{\Omega}_4$. The first three elements are of size $[0,1] \times [0,2] \times [0,1]$, but the fourth element is two times as large in the x-direction. The interfaces Γ_a, Γ_b and Γ_c are defined as $\Gamma_a = \overline{\Omega}_1 \cap \overline{\Omega}_2, \Gamma_b = \overline{\Omega}_2 \cap \overline{\Omega}_3$ and $\Gamma_c = \overline{\Omega}_3 \cap \overline{\Omega}_4$. Boundary variables are assumed to be eliminated. We introduce the following notations for the unknowns u and the right-hand side f: $f_1, u_1 \in \Omega_1, f_2, u_2 \in \Omega_2, f_3, u_3 \in \Omega_3, f_4, u_4 \in \Omega_4, f_a, u_a \in \Gamma_a, f_b, u_b \in \Gamma_b$ and $f_c, u_c \in \Gamma_c$. The Helmholtz equation $Hu = f$ can be written as

$$\begin{pmatrix} H_{11} & 0 & 0 & 0 & H_{1a} & 0 & 0 \\ 0 & H_{22} & 0 & 0 & H_{2a} & H_{2b} & 0 \\ 0 & 0 & H_{33} & 0 & 0 & H_{3b} & H_{3c} \\ 0 & 0 & 0 & H_{44} & 0 & 0 & H_{4c} \\ H_{a1} & H_{a2} & 0 & 0 & H_{aa} & H_{ab} & 0 \\ 0 & H_{b2} & H_{b3} & 0 & H_{ba} & H_{bb} & H_{bc} \\ 0 & 0 & H_{c3} & H_{c4} & 0 & H_{cb} & H_{cc} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_a \\ u_b \\ u_c \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_a \\ f_b \\ f_c \end{pmatrix}. \tag{4.28}$$

The following notations have been used:

H_{ii}	discretization of the Helmholtz operator on subdomain i (internal nodes)
$H_{i\alpha}$	coupling between the unknowns for subdomain i and interface α
$H_{\alpha i}$	coupling between the unknowns for interface α and subdomain i
$H_{\alpha\beta}$	coupling between the unknowns for interface α and interface β.

We have that $H_{ii} \in \mathcal{R}^{N^3 \times N^3}, H_{\alpha i} \in \mathcal{R}^{N^2 \times N^3}$ and $H_{\alpha\beta} \in \mathcal{R}^{N^2 \times N^2}$, $i \in \{1,2,3,4\}$, $\alpha, \beta \in \{a,b,c\}$. Moreover, it can be shown that $H_{i\alpha} = H_{\alpha i}^T$ and $H_{\alpha\beta} = H_{\beta\alpha}^T$. Elimination of the variables at the interior nodes leads to the following system, often called the Schur complement:

$$
\begin{array}{rl}
(H_{aa} - H_{a1}H_{11}^{-1}H_{1a} - H_{a2}H_{22}^{-1}H_{2a})u_a + & (H_{ab} - H_{a2}H_{22}^{-1}H_{2b})u_b = \\
 & f_a - H_{a1}H_{11}^{-1}f_1 - H_{a2}H_{22}^{-1}f_2 \\
(H_{bb} - H_{b2}H_{22}^{-1}H_{2b} - H_{b3}H_{33}^{-1}H_{3b})u_b + & (H_{ba} - H_{b2}H_{22}^{-1}H_{2a})u_a + \\
(H_{bc} - H_{b3}H_{33}^{-1}H_{3c})u_c = & f_b - H_{b2}H_{22}^{-1}f_2 - H_{b3}H_{33}^{-1}f_3 \\
(H_{cc} - H_{c3}H_{33}^{-1}H_{3c} - H_{c4}H_{44}^{-1}H_{4c})u_c + & (H_{cb} - H_{c3}H_{33}^{-1}H_{3b})u_b = \\
 & f_c - H_{c3}H_{33}^{-1}f_3 - H_{c4}H_{44}^{-1}f_4
\end{array} . \tag{4.29}
$$

System (4.29) shows that the unknowns of the three interfaces, although they do not have any node in common, are coupled in a direct way. Moreover, the appearance of the matrices $H_{\alpha\beta}$ $(\alpha \neq \beta)$ is due to the high-order approximations and is absent in classical discretization methods, as the finite element or the finite difference technique. Schematically, the Schur complement of our example has the following form

$$
\begin{pmatrix} \square & \square & 0 \\ \square & \square & \square \\ 0 & \square & \square \end{pmatrix} \begin{pmatrix} u_a \\ u_b \\ u_c \end{pmatrix} = \text{r.h.s.} \ , \tag{4.30}
$$

where the square box $\square$ represents an $N^2 \times N^2$ full block matrix. The FDM is used to evaluate the expressions $H_{ii}^{-1}f_i$ in the right-hand side of (4.29) and to compute the variables at the interior nodes:

$$
\begin{array}{rcl}
H_{11}u_1 & = & -H_{1a}u_a + f_1 \\
H_{22}u_2 & = & -H_{2a}u_a - H_{2b}u_b + f_2 \\
H_{33}u_3 & = & -H_{3b}u_b - H_{3c}u_c + f_3 \\
H_{44}u_4 & = & -H_{4c}u_c + f_4
\end{array} . \tag{4.31}
$$

In case of a more general (curvy) geometry, the Schur complement method can still be useful. Although the FDM does not apply anymore, the inverse of the Helmholtz operators can be computed by a classical direct method or by an iterative method preconditioned by finite elements [24]. The advantage is that these computations are decoupled per element.

The Schur complement problem (4.29), involving the interface variables, can be solved either by a direct or by an iterative method. In the case of a direct method, it is clear that the implicit, tensorized form in which the Schur matrix (4.29) is written, should be replaced by an explicit formulation. It is also interesting to construct the matrix explicitly when an iterative method, like PCGM, is used. This becomes clear when we take a closer look at the block-matrices that form the Schur matrix. Using

expression (4.27) to evaluate H_{11}^{-1} and H_{22}^{-1}, the number of operations to multiply one of the block-matrices, for example $(H_{aa} - H_{a1}H_{11}^{-1}H_{1a} - H_{a2}H_{22}^{-1}H_{2a})$, is $17N^4$. When this block is constructed explicitly, the operation count of a block-vector multiplication is reduced to N^4. Moreover, since the explicit formulation does not contain tensor products, block-vector products can efficiently be computed on vector computers. It can be shown that the price to construct these blocks is $O(N^5)$. This computation is done once and for all in a preprocessing stage, so that its cost will be amortized.

Preconditioning

Let us first discuss the preconditioning of the original Helmholtz operator H. The condition number of this operator depends on Δt. For small values of Δt, the operator $H = (Re^{-1}A + \Delta t^{-1}B)$ tends to the diagonal matrix B, containing the Gauss-Lobatto-Legendre weights. For larger values of Δt, the Laplacian A becomes dominant. The matrix $B^{-1}A$ is ill-conditioned, since its condition number is proportional to N^4 [65]. A preconditioner that works well for small and large values of Δt is the diagonal of the Helmholtz matrix H. This preconditioner is used when we compare the new method to the classical one. Alternatives could be a preconditioner based on the incomplete Choleski [47] or the finite element method.

In this paper, however, we are not concerned with the preconditioning of the Helmholtz operator, but with the associated Schur complement. Although the condition number of the Schur matrix is smaller than that of the original system, preconditioning is still essential. Many preconditioners have been proposed in the literature, e.g. [8], [16]. In this paper, we will consider two preconditioners; the inverse of the diagonal of (4.29) and the block diagonal matrix, which has as entries the inverse of the diagonal blocks of (4.30). The first preconditioner is easy to construct and its cost per multiplication is negligible, but its efficiency is low, as will be shown later. The effect of the block diagonal preconditioner, also called block Jacobi, is impressive. A question, however, is whether the price to construct this preconditioner (K_e matrices of dimension $N^2 \times N^2$ have to be inverted) is not too high, especially when compared to the inversion of the complete system (4.29). For this simple, four-element problem this might indeed be the case, but it is interesting to compare the cost for a larger number of interfaces. Let us define the number of interfaces by K_i. Taking into account that the block matrices on the diagonal are symmetric, computing the preconditioner requires $K_i N^6/6$ operations, whereas the inversion of the complete Schur complement requires $K_i^3 N^6/6$ operations. Hence, the iterative method becomes attractive for larger K_i. The construction of the preconditioner is performed once and for all and the cost is spread out over hundreds of pressure iterations.

In order to give some heuristics about the spectrum of the Schur complement and of the Schur complement premultiplied by the diagonal and the block-diagonal preconditioners, we computed the condition numbers of these three operators for different values of K_i, N and $\Delta t Re^{-1}$. Table III represents the results for a geometry with dimensions $[0,1] \times [0,4] \times [0,1]$, with respectively 4, 8 and 16 elements in y-direction and one in the remaining directions. For this particular mesh, the Schur complement is tridiagonal by block. We remark that the results representing a Navier-Stokes sim-

		$\Delta t = 0.25$, Stokes			$\Delta t = 0.01$, $Re = 10$		
K_i	N	CS	$D^{-1}CS$	$(BD)^{-1}CS$	CS	$D^{-1}CS$	$(BD)^{-1}CS$
3	5	2.66	2.45	1.02	1.10	1.09	1.00
3	7	4.61	3.67	1.02	1.29	1.22	1.00
3	9	7.36	4.92	1.02	1.57	1.36	1.00
3	11	10.94	6.19	1.02	1.98	1.54	1.00
7	5	2.88	2.68	1.38	1.08	1.08	1.00
7	7	5.13	4.14	1.38	1.21	1.16	1.00
7	9	8.27	5.63	1.38	1.47	1.30	1.00
7	11	12.34	7.14	1.38	1.89	1.50	1.00
15	5	4.84	4.60	3.29	1.07	1.06	1.00
15	7	7.93	6.50	3.29	1.19	1.15	1.00
15	9	12.68	8.67	3.29	1.46	1.30	1.00

Table III: **Condition number of the Schur complement (CS), the Schur complement premultiplied by the diagonal preconditioner ($D^{-1}CS$) and the Schur complement premultiplied by the block-diagonal preconditioner ($(BD)^{-1}CS$) for different values of K_i and N.**

ulation ($\Delta t = 0.01$, $Re = 10$) give rise to condition numbers close to one. Moreover, they seem to be more or less independent of the number of interfaces. For $\Delta t = 0.25$, $Re = 1$, we see that the condition number of the Schur complement, for a fixed value of K_i, is of order $O(N^2)$. The diagonal preconditioner seems to reduce the condition number to $O(N)$ and the block-diagonal preconditioner yields a condition number that is independent of N. There is no obvious relation between the spectrum and K_i, other than that the condition number grows as K_i increases.

Although the construction of an efficient preconditioner for the pressure operator (4.25) is beyond the scope of this paper, we make the following remarks: for large values of Δt, this operator is well-conditioned and can be preconditioned by a diagonal matrix containing the Gauss-Legendre weights. The problem is much more difficult for small values of Δt. In a recent paper, Rønquist [63] proposes a preconditioner based on the decomposition of the pressure system into two pressure systems. Here, we confine ourselves to the simple diagonal preconditioner based on the Gauss-Legendre weights.

4.2.5 Comparison of different methods

Numerical results for a simple test problem

The new algorithm, based on the Schur method and the direct inversion is compared to the classical one, where the complete Helmholtz equation is solved by the PCGM. The difference between a direct and an iterative method in solving the Schur complement is examined. Moreover, in the case of an iterative solver, the performance of the preconditioners is investigated. The tests have been run on a Convex C3820 vector computer. Every program has been compiled with and without vector optimization.

The routines that perform the multiplication by the Schur complement or by the inverse of the Schur complement (in the case of an iterative or direct method respectively) are written in BLAS.

As a test problem, we have taken the Stokes flow, in which the non-linear terms in (4.10) are neglected. The geometry is given in Figure 4.5. The degree of the approximating polynomials is equal to ten for the velocities and eight for the pressure, $K_e = 4$ and $K_i = 3$. The analytical solution is given by

$$\begin{aligned} \underline{u}(x_1, x_2, x_3) &= \left(-cos(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_3), 0, sin(\frac{\pi}{2}x_1)cos(\frac{\pi}{2}x_3)\right)^T \\ p(x_1, x_2, x_3) &= -\pi sin(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_3). \end{aligned} \tag{4.32}$$

At $t = 0$, the fluid is at rest and the boundary conditions match the analytical solution. After ten time steps ($\Delta t = 0.5$) a steady solution is obtained. Four different methods are compared: The classical iterative (CI) method, the direct Schur complement (DS) method, the iterative Schur complement method preconditioned by the diagonal (ISD) and the iterative Schur complement method preconditioned by the block-diagonal (ISB). Table IV shows the results of the runs without vector optimization. The accu-

Method	CI	DS	ISD	ISB
Seconds	2637	191	330	223

Table IV: **Timings in seconds after ten time steps for the test problem without vector optimization**

racy of the four methods is of the same order. We found a maximum error of $3 \cdot 10^{-5}$ for the pressure and of $8 \cdot 10^{-7}$ for the velocities. The speed of the three methods which are based on the Schur method (DS, ISD and ISB) is much higher than that of the classical method. Furthermore, we notice that the direct inversion of the Schur complement (DS) is faster than in the iterative methods (ISB and ISD). Finally, preconditioning with the block diagonal matrix is preferred to the diagonal preconditioner. We found that, for any Schur method, the preprocessing time (construction of the Schur complement and preconditioner, eigenvalue decomposition of mono-dimensional operators necessary for fast diagonalization) is less than two seconds.

The results for the vectorized programs can be found in Table V . We notice that, as expected, the Schur methods benefit more from vectorization than the classical method. The difference between the methods DS, ISD and ISB has almost completely disappeared. This can be explained as follows: The time for the computation of the interfaces has become negligible with respect to the computation of the interior nodes, which, due to the tensor products, does not vectorize well. An analysis of the distribution of the cpu time over the different subroutines reveals that for this particular

Method	CI	DS	ISD	ISB
Seconds	1565	105	114	111

Table V: **Timings in seconds after ten time steps for the test problem with vector optimization**

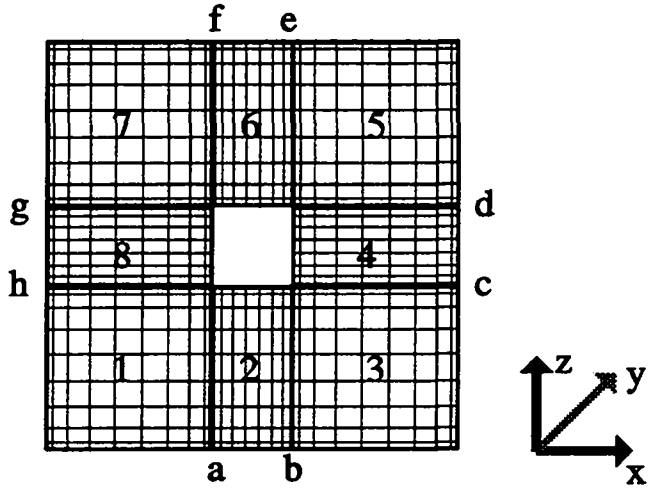

Figure 4.6: **Geometry consisting of eight spectral elements. Projection in y-direction (y=0.5). The sets of interior points are enumerated 1..8; the interfaces a..h.**

problem, the ISB method spent a considerable amount of time (about 47%) to compute the interior nodes by FDM, whereas the interface routines took only 5%.

Numerical results on an eight-element geometry

In this paragraph we consider a more complex geometry, consisting of the cube $[0,1]^3$ with a hole in the center $[0.4,0.6] \times [0,1] \times [0.4,0.6]$. The number of spectral elements is eight ($K_i = K_e = 8$) and the polynomial degree is ten. Figure 4.6 shows a projection of the cube in the y-direction. Homogeneous Dirichlet boundary conditions are applied everywhere, except at the top plane ($z = 1$), where $\underline{u}_1 = 16xy(x-1)(y-1)$. The Schur complement that results after elimination of the interior nodes is somewhat more complex than that of the four-element matrix (4.29). We will refrain from giving the full system and give, as an example, the equations at the interface a

$$(H_{aa} - H_{a1}H_{11}^{-1}H_{a1}^T - H_{a2}H_{22}^{-1}H_{a2}^T)u_a + (H_{ab} - H_{a2}H_{22}^{-1}H_{b2}^T)u_b \; - \\ H_{a1}H_{11}^{-1}H_{h1}^T u_h = f_a - H_{a1}H_{11}^{-1}f_1 - H_{a2}H_{22}^{-1}f_2 \quad . \qquad (4.33)$$

The equations at the other interfaces are given by similar expressions, since the geometry is symmetric with respect to the interfaces. Schematically, the Schur complement can be written in the following form:

$$\begin{pmatrix} \square & \square & 0 & 0 & 0 & 0 & 0 & \square \\ \square & \square & \square & 0 & 0 & 0 & 0 & 0 \\ 0 & \square & \square & \square & 0 & 0 & 0 & 0 \\ 0 & 0 & \square & \square & \square & 0 & 0 & 0 \\ 0 & 0 & 0 & \square & \square & \square & 0 & 0 \\ 0 & 0 & 0 & 0 & \square & \square & \square & 0 \\ 0 & 0 & 0 & 0 & 0 & \square & \square & \square \\ \square & 0 & 0 & 0 & 0 & 0 & \square & \square \end{pmatrix} \begin{pmatrix} u_a \\ u_b \\ u_c \\ u_d \\ u_e \\ u_f \\ u_g \\ u_h \end{pmatrix} = \text{r.h.s.} \ , \tag{4.34}$$

System (4.31) illustrates that for large K_i direct inversion of the Schur complement is not a good idea (see discussion in paragraph 4.2.4). Therefore, an iterative method (ISB) is preferred.

Problem	Stokes $\Delta t = 0.5$		Navier-Stokes $Re = 100, \Delta t = 0.005$	
Method	CI	ISB	CI	ISB
Seconds	702	40	670	140
Average number of iterations	87	18	17	6

Table VI: **Timings in seconds after one time step for the eight-element problem on a Convex 3820 using vector optimization. The average number of iterations (tolerance 10^{-12}) is given per Helmholtz solve (CI) or per solve of the Schur complement (ISB).**

Table VI compares the cpu time of the ISB and CI methods for the first time step of a Stokes flow and a Navier-Stokes flow ($Re = 100$, where the characteristic velocity U equals one). Again, the new algorithm is much faster than the previous one. The large differences between the computation times for the Stokes and Navier-Stokes problem can be explained by the fact that for small values of Δt more iterations will be needed to compute the pressure, whereas the Helmholtz operator is well conditioned, resulting in a low number of internal iterations, as is illustrated by the last line of Table VI.

4.2.6 Closing remarks

In Fischer et al. [29] and in Fischer and Patera [26], the parallelization of the classical CI method is discussed. Large computational kernels, like the computation of gradients and the multiplication by the Helmholtz operator H can be performed in parallel on each spectral element.

The parallel efficiency of our method will not differ very much from that of the original method, at least when computers based on shared memory are considered: The additional computational kernels can naturally be parallelized. The work to construct the Schur complement method and its preconditioner can be divided over different

processors. Clearly, the computation of the interior nodes by FDM can be done independently on each element. Finally, when an iterative method is used to solve the Schur complement system, the matrix-vector multiplication is split in block matrix-vector operations. In table VII, we give the results on the parallel Alliant FX/8 for the Stokes and Navier-Stokes problems, defined in the previous paragraph. The parallel efficiency is about 85% for the Stokes and 80% for the Navier-Stokes problem.

Problem	Stokes $\Delta t = 0.5$		Navier-Stokes $Re = 100, \Delta t = 0.005$	
Mode	vec	vec+par	vec	vec+par
# proc.	1	4	1	4
Seconds	544	160	1694	532

Table VII: **Timings in seconds of the first time step for the test problem on an Alliant FX/8. The method used is ISB.**

On distributed memory computers, where communication is an issue, the Schur method might have a disadvantage compared to the original CI method. According to Fischer et al. [29], the latter method only requires the communication of scalars (to assemble the dot product) and points at the interfaces (for the direct stiffness). The interfaces are only exchanged between neighboring elements. In our case, the FDM and the construction of the preconditioner are communication free. The construction of the Schur complement matrix requires some communication, but this is done only once in a preprocessing stage. The difficulty is related to the implementation of the iterative method to solve the Schur complement. Since the Schur matrix is in general not block diagonal (see for example (4.30,4.34)), the interface variables have to be exchanged between processors. In the case of system (4.34) for instance, u_b and u_h have to be sent to the processor that computes the first row. The time to send these $O(N^2)$ messages can be relatively large with respect to the fast block matrix-vector multiplication, which is of order $O(N^4)$. Preliminary results show that the performance of the iterative solver depends heavily on the architecture of the parallel distributed memory computer.

The algorithm we investigated in this paper leads to an important acceleration of the Uzawa algorithm applied to the discrete Navier-Stokes equations. Since the conditions for the FDM are very restrictive, the present method can only be applied to geometries consisting of non-deformed spectral elements. This is a serious limitation. However, since the Schur method decouples the global Helmholtz operator H in K_e local operators H_{ii}, classical direct methods can also be considered. Once the inverses H_{ii} (or the LL^T decomposition) have been computed, the Schur complement matrix can be constructed, yielding no extra computational effort to solve the Schur complement problem, apart from a higher preprocessing cost. The decoupled problems for the interior nodes ($H_{ii}u_i$ = r.h.s.) can be solved either by multiplication by the local inverse (or by back substitution) or by an iterative method preconditioned by finite elements [24]. Fischer and Rønquist [28] already showed in the context of preconditioning of the pressure operator that the construction of local inverses by a classical direct method

is feasible and advantageous in terms of computation times. Both the parallelization on distributed memory machines and the implementation of deformed geometries will be the subject of future research.

Acknowledgement. The above text presents research results of the Belgian Incentive Program "Information Technology - Computer Science of the Future", initiated by the SPPS (Services du Premier Ministre. Programmation de la Politique Scientifique). The scientific responsibility is assumed by the authors.

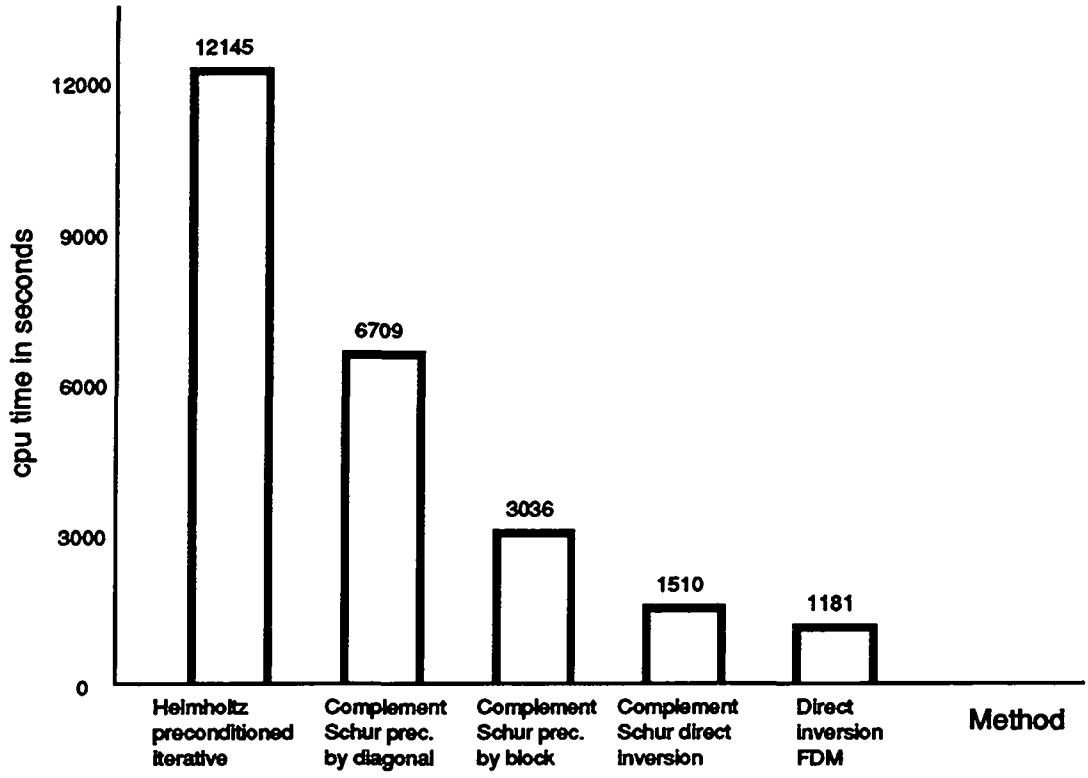

Figure 4.7: Comparison of the methods presented in the previous two sections with respect to their cpu time (on a Data General workstation) to compute steady Stokes solution. $K = 4$, $N = 9$, $\Delta t = 0.25$, $\nu = 1$.

4.3 Incomplete Schur preconditioning

Before discussing some of the disadvantages of the Schur complement method and some suggestions to precondition the Helmholtz operator on generally deformed geometries, we will first compare several techniques to solve the analytical problem presented in Section 4.1. For the geometry displayed in Figure 4.4, we computed a steady Stokes flow with analytical solution (2.86), (2.87) and $\nu = 1$, $\Delta t = 0.25$, $N = 9$. We compare the cpu time for the iterative Helmholtz solver preconditioned by the diagonal of H, the Schur complement method solved by an iterative (with resp. diagonal and block diagonal preconditioners) or direct technique and the FDM applied to the entire geometry (see Section 4.1). As expected, the cpu times for the Schur complement methods lie in between the two extreme; the fully iterative Helmholtz solver preconditioned by the diagonal and the FDM applied to the entire domain. Furthermore, since the problem is rather small, a direct inversion of the Schur matrix works out well.

The solution of the Helmholtz equation by the Schur complement method requires respectively the FDM for the computation of the right-hand side ($H_{ii}^{-1} f_i$, according to Equation 4.29), the inversion of the Schur complement matrix, preferably by an iterative method, and a second application of the FDM to compute the velocities at the interior nodes (c.f. Equation 4.31). This process is shown in Figure 4.8.

The Schur complement method has been tested on a number of relatively small test problems. The potentially very complex structure of the Schur matrix refrained us from

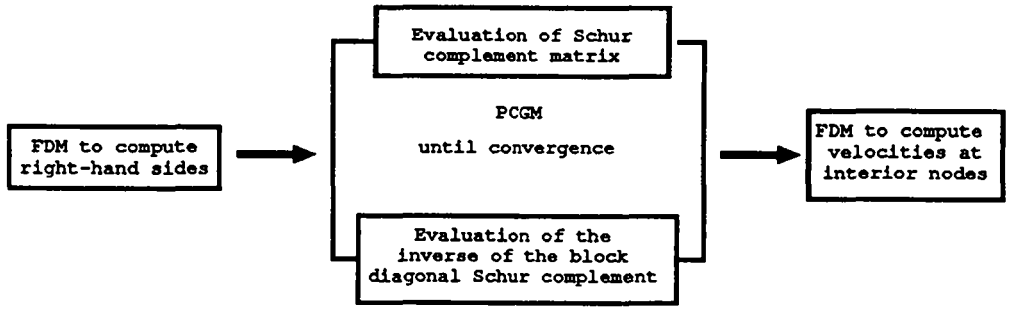

Figure 4.8: **Solution of the Helmholtz problem by block-diagonally preconditioned Schur complement method.**

treating large problems. Because of the high-order coupling, the number of entries (i.e. block matrices) is much higher than for interface operators based on, for example, finite element discretizations. A solution to this problem could be to divide the geometry in the smallest possible number of boxes. Inside each box consisting of multiple spectral elements the FDM can be applied and the size of the interface matrix will be small. In this way, the 3D backward-facing step geometry, for example, can be decomposed in two "superboxes" which have only one interface in common. Almost any mesh allows such a decomposition by large blocks. Unfortunately, this approach is in contradiction with our goal to implement the solver on a parallel computer, which requires decomposition in many, small parts, instead of a few, large parts. Nevertheless, this idea of reducing the size and complexity of the Schur problem remains attractive for scalar computers. Besides the doubtful efficiency on parallel computers, the aforementioned method can still not deal with deformed geometries.

It is not unusual to precondition operators on deformed geometries by matrices that are constructed on nondeformed meshes. In Section 2.4, we have showed that the number of iterations depends on the rate of deformation. In the following, we will investigate this strategy for the Helmholtz operator. To this end, let us consider a geometry Ω decomposed in K deformed spectral elements $\{\Omega_k\}_{k=1}^K$. We then define the dimensions of the corresponding parallelepipedic spectral elements $\tilde{\Omega}_k$ as the average difference in the x, y, resp. z coordinates of the eight corners of Ω_k, as has been illustrated in Figure 2.1, Section 2.4 (2D case). The Helmholtz operator can then be preconditioned by the Schur complement method applied to the abstract geometry consisting of the set of parallelepipedic elements $\{\tilde{\Omega}_k\}_{k=1}^K$. Then, we still have to deal with the complex structure of the Schur matrix. In the previous section it has been shown that the block diagonal is an excellent preconditioner for the Schur matrix. Therefore, it is a logical idea to replace the (complicated) Schur matrix by its block diagonal. Heuristically, we expect that this will not dramatically increase the number of iterations when the geometry is already deformed. We will call this technique the incomplete Schur method. It is schematically displayed in Figure 4.9.

We have tested the incomplete Schur method on a box geometry ($[0,4] \times [-1,1] \times$

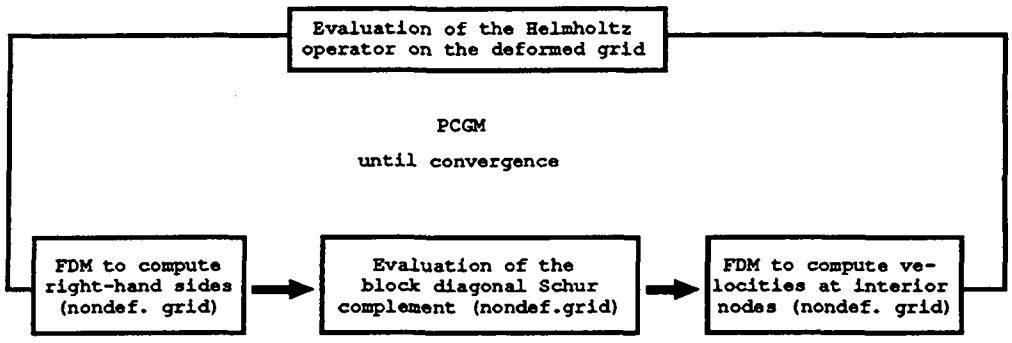

Figure 4.9: **Solution of the Helmholtz problem on deformed geometries by incomplete Schur preconditioning.**

$[0,1]$) consisting of $4 \times 2 \times 2$ spectral elements. The elements in the x-direction are deformed by the function $\alpha \sin(\pi y) \sin(\pi z)$, with α a parameter that determines the rate of deformation. We solved a Poiseuille flow (Stokes, $\nu = 1$), using the preconditioned pressure correction method (see Chapter 5 and Chapter 6). The tolerance for convergence is set to 10^{-9} for the velocities and 10^{-6} for the pressure. The results

	$\alpha = 0$			$\alpha = 0.15$		
Preconditioner	N=6	N=8	N=10	N=6	N=8	N=10
Inc. Schur	417; 8'	470; 25'	521; 61'	1481; 20'	943; 50'	1162; 134'
Diagonal	2425;19'	2496; 49'	3096; 118'	2829; 24'	2987; 70'	3509; 175'

Table VIII: **Number of Helmholtz iterations and cpu time in minutes (R3000 Silicon Graphics) to perform 10 time steps ($\Delta t = 0.1$) to compute a Poiseuille flow. The interfaces in x-direction are deformed by the function $\alpha \sin(\pi y) \sin(\pi z)$.**

Preconditioner	N=6	N=8	N=10
Inc. Schur	1819; 39'	2227; 109'	2830; 293'
Diagonal	6245; 60'	8157; 173'	10249; 814'

Table IX: **Number of Helmholtz iterations and cpu time in minutes (R3000 Silicon Graphics) to perform 100 time steps ($\Delta t = 0.01$) to compute a Poiseuille flow. No deformation.**

have been depicted in Table VIII ($\Delta t = 0.1$) and in Table IX ($\Delta t = 0.01$). In general, a good acceleration of the convergence rate is obtained. We should note that the results of this section are encouraging, but preliminary: We can not explain the high number of iterations for the incomplete Schur algorithm ($N = 6$, $\alpha = 0.15$, Table VIII).

Chapter 5

Decoupling methods

In this chapter we focus on techniques to decouple the pressure from the velocity field. This induces a time error the order of which should be at least the same as the order of the overall time scheme. A convenient way to analyze such decoupling (projection) methods is by writing them as generalized block LU decompositions, according to the paper of Blair Perot [9]. A family of high-order projection methods will be derived. Several arguments are given to apply the decoupling procedure to the space-discrete equations and not to the space-continuous equations. The methods are validated by numerical tests.

5.1 Projection methods applied to space-discrete and -continuous Navier-Stokes equations

Let us recall the spectral element discretization of the incompressible Navier-Stokes (NS) equations using the notations of Chapter 2 and Chapter 3:

$$\frac{\beta_s}{\Delta t} B\underline{u}^{n+1} + Re^{-1} A\underline{u}^{n+1} - D^T p^{n+1} = B\underline{f}^{n+1} \tag{5.1}$$

$$-D\underline{u}^{n+1} = 0. \tag{5.2}$$

Boundary conditions are incorporated in the operators. The right-hand-side vector $\underline{f}$ contains all explicit terms, resulting from a BDF scheme of order s for the linear terms and an explicit time discretization for the nonlinear terms by either the extrapolation method or an operator-integration-factor splitting method, as discussed in Chapter 3. The constant β_s depends on the order s of the BDF. The velocity belongs to X_N and the pressure to M_N, defined in Equation (2.69). We can write (5.1) and (5.2) as

$$\begin{pmatrix} H & -D^T \\ -D & 0 \end{pmatrix} \begin{pmatrix} \underline{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} B\underline{f}^{n+1} \\ 0 \end{pmatrix}, \tag{5.3}$$

with $H = \frac{\beta_s}{\Delta t}B + Re^{-1}A$. Problems arise if we apply the conjugate gradient method directly to (5.3): The zero block in the continuity equation often leads to no, or very slow convergence of the iterative method. Some solutions using fill-in exist (see e.g. Dahl and Wille [22]), but it is common practice to derive separate equations from (5.3) for the three velocity components and the pressure. As an example, we already proposed the Uzawa method in Section 2.6.

The simplest, extensively studied decoupling method is the fractional step method, which goes back to Chorin [17] and Teman [73]. This method introduces a first-order time error which can be improved to second order by computing a correction to the pressure (see e.g. Van Kan [75]). However, these projection methods are applied to the space-continuous (time-discrete) NS equations and require an additional boundary condition for the pressure. This additional condition has to be approximated at an order which is at least the same as the decoupling error (see Karniadakis et al. [42]). This is quite a difficult task and it is not easy to analyze these schemes.

In a recent article of Blair Perot [9], which is excellent because of its simplicity, it is proposed to apply the decoupling method to the discrete (in time and in space) set of equations (5.3). Boundary conditions have already been incorporated and the projection error can easily be studied. This allows not only to find in retrospect the correct pressure boundary condition for the fractional-step method applied to the *space-continuous* equations, but also to find a general formulation of projection methods of *any* order. This is done by applying so-called generalized block LU decompositions to

$$\begin{pmatrix} H & -HQD^T \\ -D & 0 \end{pmatrix} \begin{pmatrix} \underline{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} B\underline{f}^{n+1} \\ 0 \end{pmatrix}, \tag{5.4}$$

with Q an arbitrary matrix that determines the projection method and is to be defined later. We close this section by referring to the article of Maday et al. [48] who provide an operator-integration-factor splitting technique which is a very good alternative for the projection methods discussed in this chapter.

5.2 Generalized block LU decompositions

The solution of the block LU decomposition of (5.4) takes place in two steps:

$$\begin{pmatrix} H & 0 \\ -D & -DQD^T \end{pmatrix} \begin{pmatrix} \underline{u}^{*} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} B\underline{f}^{n+1} \\ 0 \end{pmatrix} \tag{5.5}$$

and

$$\begin{pmatrix} I & -QD^T \\ 0 & I \end{pmatrix} \begin{pmatrix} \underline{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} \underline{u}^* \\ p^{n+1} \end{pmatrix}. \tag{5.6}$$

Here, $\underline{u}^*$ is an intermediate velocity field that is not necessarily divergence-free. The second step (5.6) can be seen as the projection of the non divergence-free $\underline{u}^*$ on the divergence-free field $\underline{u}^{n+1}$. For this reason, the above method is often called the projection method. Note that since boundary conditions have already been applied to the discrete operators, no problems arise for the solution of the pressure p^{n+1}, as would have been the case when the continuous operators are subjected to a block LU decomposition. The power of this approach is that the system (5.5), (5.6) covers a large class of projection methods, depending on the choice for Q.

The first, most obvious choice for Q is

$$Q = H^{-1}. \tag{5.7}$$

In this way, Equations (5.5) and (5.6) are equivalent to the original system (5.3) and no decoupling error has been introduced. This method is often referred to as the Uzawa method and has as a disadvantage that the inverse of H appears in the computation of the pressure. The performance of the Uzawa method will depend on the efficiency this inversion is computed with.

A second choice is

$$Q = \frac{\Delta t}{\beta_s} B^{-1}, \tag{5.8}$$

which leads to a method we will refer to as the fractional-step (FS) method, although this term was originally introduced for space-continuous splitting. By means of (5.5), (5.6), it is easily verified that the decoupling error, introduced by the fact that Q is only an *approximation* of H^{-1} relies on the following relation

$$-D^T p^{n+1} - (-HQD^T p^{n+1}) = \frac{\Delta t}{\beta_s Re} AB^{-1} D^T p^{n+1} = O(\Delta t). \tag{5.9}$$

In the momentum equations (5.1) there is a factor Δt^{-1} in front of the velocity which implies together with (5.9) that the local time error is of order two. Consequently, the FS method is first-order accurate in time. Furthermore, it possesses a non-vanishing decoupling error, even when the problem is stationary. Note that the decoupling error is not determined by a rigorous error analysis, but by simply computing the order of the perturbation with respect to the original matrix. The results are verified by numerical tests in the next section. Note also that we speak here about the order of accuracy of

the velocities, denoted by π_v. The order of the pressure is a source of confusion and contradictory statements can be found in literature. Here, we confine ourselves to the remark that we numerically found out that the order of accuracy π_p of the pressure respects $\pi_v - 1 \leq \pi_p \leq \pi_v$. This holds for all the time schemes we tested in this chapter.

The first-order accurate result (5.8) can be improved by choosing a better approximation for H^{-1} (see [9]), for example

$$Q = \frac{\Delta t}{\beta_s} B^{-1} - \left(\frac{\Delta t}{\beta_s}\right)^2 Re^{-1} B^{-1} A B^{-1}, \tag{5.10}$$

which is nothing more than a Taylor-series approximation of H^{-1}. This leads to a second-order decoupling error based on the following relation:

$$-D^T p^{n+1} - (-HQD^T p^{n+1}) = \left(\frac{\Delta t}{\beta_s Re}\right)^2 (AB^{-1})^2 D^T p^{n+1} = O(\Delta t)^2. \tag{5.11}$$

In practice, this method does not work well, because (5.10) is positive definite only for very small values of $\Delta t Re^{-1}$. Its third-order analogue

$$Q = \frac{\Delta t}{\beta_s} B^{-1} - \left(\frac{\Delta t}{\beta_s}\right)^2 Re^{-1} B^{-1} A B^{-1} + \left(\frac{\Delta t}{\beta_s}\right)^3 Re^{-2} (B^{-1}A)^2 B^{-1} \tag{5.12}$$

is always positive definite and yields a decoupling error based on

$$-D^T p^{n+1} - (-HQD^T p^{n+1}) = \left(\frac{\Delta t}{\beta_s Re}\right)^3 (AB^{-1})^3 D^T p^{n+1} = O(\Delta t)^3. \tag{5.13}$$

We will call this method BP3. Note that all the decoupling errors (5.9), (5.11), and (5.13) are non-vanishing for steady problems and that different choices of Q lead to different pressure operators, requiring each different preconditioning techniques. It turned out that especially the pressure operator for (5.12) is ill conditioned. This will be discussed in Chapter 6.

Another way of improving the decoupling error of the fractional-step method consists in modifying (5.4) in

$$\begin{pmatrix} H & -HQD^T \\ -D & 0 \end{pmatrix} \begin{pmatrix} \underline{u}^{n+1} \\ p^{n+1} - p^n \end{pmatrix} = \begin{pmatrix} B\underline{f}^{n+1} + D^T p^n \\ 0 \end{pmatrix}. \tag{5.14}$$

Applying the block LU decomposition to (5.14), with

$$Q = \frac{\Delta t}{\beta_s} B^{-1}, \tag{5.15}$$

leads to the pressure-correction (PC) method [75], which is a second-order scheme. The decoupling error is determined with the aid of

$$-D^T p^{n+1} - (-HQD^T(p^{n+1} - p^n)) - (-D^T p^n) \;=$$

$$\frac{\Delta t}{\beta_s Re} AB^{-1} D^T (p^{n+1} - p^n) \;=\; O(\Delta t)^2. \tag{5.16}$$

This decoupling error vanishes when the problem is steady (i.e. $p^{n+1} - p^n = 0$). The same technique (i.e. adding $D^T p^n$ to the right-hand-side vector $\underline{f}^{n+1}$ and computing a correction term for the pressure) can be applied to the BP3 scheme [45]. In this way, we obtain what we will call the BP3PC scheme with a fourth-order decoupling error, based on the following estimate

$$\left(\frac{\Delta t}{\beta_s Re}\right)^3 (AB^{-1})^3 D^T (p^{n+1} - p^n) = O(\Delta t)^4, \tag{5.17}$$

which vanishes for steady problems.

Following the above described strategy, projection methods of fifth order, sixth order, etc. can be be derived. The practical interest of such methods is, however, limited, for the following reason: The overall order of the time schemes based on the different choices for Q is only the minimum of the order s of the BDF, the order of the time scheme used for the evaluation of the nonlinear terms, the order of the implicit/explicit splitting error and the order of the decoupling error.

5.3 Numerical tests

In this section we will numerically verify the order of five of the aforementioned decoupling methods, and we will compare the cpu time for each of these methods to obtain a certain level of accuracy. The test problem consists of an eight-element geometry $\Omega = (0,1)^3$ with a hole $[0.4; 0.6] \times [0,1] \times [0.4; 0.6]$, as has been used in Chapter 3 and in Chapter 4. The analytical solution is given by

$$\underline{u}(x_1, x_2, x_3) \;=\; \sin(\pi t) \left(-\cos(\frac{\pi}{2} x_1) \sin(\frac{\pi}{2} x_2), \sin(\frac{\pi}{2} x_1) \cos(\frac{\pi}{2} x_2), 0 \right)^T \tag{5.18}$$

$$p(x_1, x_2, x_3) \;=\; -\pi \sin(\pi t) \sin(\frac{\pi}{2} x_1) \sin(\frac{\pi}{2} x_2), \tag{5.19}$$

for $t \in [0, T]$. Throughout this section, the initial condition is zero velocity and pressure.

5.3.1 Accuracy of decoupling methods

As a first test, we take $Re = 10$, $T = 0.75$ and $N = 6$. The order of the BDF is three and the nonlinear terms are discretized by a third-order extrapolation scheme. There is no implicit/explicit splitting error. For this small value of the Reynolds number, we expect large decoupling errors. Table I reveals that, as far as the velocity

Method	$\Delta t = \frac{1}{40}$	$\Delta t = \frac{1}{80}$	$\Delta t = \frac{1}{160}$	$\Delta t = \frac{1}{320}$
Frac. Step	0.66; 1.55	0.85; 1.81	1.07; 2.09	1.31; 2.38
PC	1.71; 2.17	2.06; 3.22	2.64; 3.82	3.18; 4.42
BP3	0.75; 0.71	1.14; 1.55	1.68; 2.53	2.45; 3.48
BP3PC	1.66; 1.72	2.11; 3.07	3.23; 4.30	4.29; 5.36
Uzawa	3.03; 4.82	3.92; 5.63	4.82; 5.82	5.42; 5.71

Table I: **Number of correct decimal digits (-log(max error)) for the pressure resp. $\underline{u}_1$ velocity component ($T = 0.75$, $Re = 10$).**

is concerned, three schemes show the order of accuracy we expect: The fractional-step method is first-order, the pressure-correction method is second-order and the BP3 method is third-order accurate, although the large error constant of the latter scheme yields somewhat disappointing results. The Uzawa method is third-order accurate, but for small values of Δt, the spatial error becomes dominant. (The presentation of the results by the number of correct decimal digits is often used in the context of the analysis of time-integration schemes. It allows for a quick verification of the order of the method: Since $-\log(0.5) \approx 0.301$, we find that by dividing the step size by two, a gain of 0.3 digits corresponds to order 1, a gain of 0.6 digits to order 2, etc.) Note the important improvement of BP3 by computing a pressure *correction* (BP3PC). The latter scheme is formally of order three, due to the BDF3 and the extrapolation for the nonlinear terms, but shows a fourth-order behaviour. This is explained by the fact that the decoupling error is, apparently, dominant.

The order of accuracy of the pressure is much less documented in literature. Van Kan [75] mentions that the order of the pressure for his pressure-correction scheme is less than two. Blair Perot [9] states that the order for the pressure is always inferior to the order of the velocity. Our results show that, except for Uzawa of course, the order of the pressure is somewhere in between the order of the velocity and the order of the velocity minus one.

We repeated aforementioned problem, but with a Reynolds number of $Re = 50$ this time. The results are presented in Table II and show that the pressure-correction

	Frac. Step	PC	BP3	BP3PC	Uzawa
$\Delta t = \frac{1}{200}$	1.85; 2.09	3.49; 4.13	3.89; 4.21	4.96; 5.13	5.10; 5.13

Table II: **Number of correct decimal digits (-log(max error)) for the pressure resp. $\underline{u}_1$ velocity component ($T = 0.75$, $Re = 50$).**

method, BP3, and especially the BP3PC method are very close to the "optimal" Uzawa scheme. This is due to the fact that the decoupling error is proportional to Re^{-1}, Re^{-3}, and Re^{-3}, respectively. For $Re > 50$ the difference between the five methods will even be smaller.

5.3.2 Performance of decoupling methods

Performance depends on the speed the Helmholtz equation is solved with and on the efficiency of the pressure preconditioner. In this section we will present results with and without the fast semi-direct inversion method for the Helmholtz equation which has been discussed in Chapter 4. The preconditioning of the pressure operator is presented in the next chapter. All the results of this section are obtained with the most efficient preconditioning technique in terms of cpu time that can be found in Chapter 6.

The problem is the same as in Section 5.3.1, with $Re = 50$, $T = 0.3$. The Helmholtz equation is inverted by the solver based on the Schur complement method and fast elemental solves. We will compare the Uzawa method, PC, and BP3PC as regards the cpu required to obtain a certain level of accuracy, say 10^{-5}, for the $\underline{u}_1$-velocity component. This implies that the Uzawa method proceeds in time with a time step of $\Delta t = 0.02$. The corresponding error in the pressure is $4.9_{10^{-4}}$. To get the same precision for the velocity, BP3PC requires a step size $\Delta t = 0.01$ (the corresponding error in the pressure is $6.3_{10^{-5}}$). The cpu time to get this typical accuracy is 1.6 times larger for BP3PC than for Uzawa. There are three reasons that BP3PC can not win the race. First, BP3PC needs more time steps to attain the same precision. Second, the pressure preconditioner is somewhat more efficient for Uzawa than for BP3PC. Finally, the most important reason is that a very fast algorithm is available to invert H. This inversion is done at almost the same computational cost as an evaluation of Q (5.12). The PC method is a low-order method and a step size of about $\Delta t = 0.0015$ is needed to obtain the required accuracy. The computation took more than three times as much cpu time as for Uzawa.

Next, we consider a geometry $[-\frac{1}{4}, \frac{1}{4}] \times [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{4}, \frac{1}{4}]$ consisting of four spectral elements in y-direction, with solution (5.18), (5.19). We take $N = 7$, $T = 0.5$ and a Reynolds number of 50 (based on a characteristic length $L = 0.5$) and compare the number of time steps for Uzawa, PC and BP3PC to obtain a $1.6_{10^{-5}}$ precision in the velocity. This time, the Helmholtz equations are solved by a PCGM with a diagonal preconditioner. The total cpu time is normalized with respect to the PC method. We assume that spatial errors are negligible with respect to temporal errors. The order of the BDF is three. As expected (see Table III), Uzawa is the most accurate

Method	cpu time	Max. Δt
Uzawa	3.5	0.045
PC	1.0	0.017
BP3PC	5.2	0.025

Table III: **Maximum time step and corresponding cpu time to obtain a 1.6E-5 precision in the velocity.** $N = 7$, $K = 4$, $Re = 50$, $T = 0.5$. **The cpu time is normalized with respect to the preconditioned PC method.**

method, followed by BP3PC. On first sight, the results for the BP3PC method seem disappointing, but they can be explained by the fact that $Re^{-1}\Delta t$ is not small enough. This works out negatively in two ways: The decoupling error is large, leading to a relatively small Δt to obtain the demanded accuracy and the preconditioner is not optimal for $Re^{-1}\Delta t$ in this range.

Finally, we define a problem on four deformed spectral elements in a row ($N = 6$, $\Omega = [-\frac{1}{4}, \frac{1}{4}] \times [0, 2] \times [-\frac{1}{4}, \frac{1}{4}]$), with a *steady* solution. This solution corresponds to (5.18) and (5.19), without the time dependent term $\sin(\pi t)$. The three interfaces are deformed by a sine function. Since there is no reason to use high-order time schemes for steady problems, the BDF3 is replaced by BDF1 (yielding $\beta_s = 1$). In this case, no fast Helmholtz solver is used. We take $Re = 50$, $\Delta t = 0.015$ and compare the number of time steps and cpu time to obtain a 2_{10-4} precision for the $\underline{u}_1$-velocity component. The results for Uzawa have been obtained with the two-step preconditioning technique (see Chapter 6). In Table IV, we give the number of time steps to obtain the aforementioned precision, the cpu time (measured on a small workstation) and the error in the stationary solution at $t = 10$. We remark that after 600 time steps the two methods that are based on the computation of a correction to the pressure (PC and BP3PC) still iterate to compute the pressure, although the solution does hardly change. Uzawa and BP3 "recognize" that the solution is stationary. Apparently, the pressure correction takes a long time (if not forever!) to damp out. The results for Uzawa, PC and

	PC	BP3	BP3PC	Uzawa
time steps	78	83	78	78
cpu time	1	2.75	2.21	4.82
error in steady flow	8.8_{10-5}	1.5_{10-4}	8.8_{10-5}	8.8_{10-5}

Table IV: **Number of time steps, cpu time (normalized with respect to PC) to obtain 2_{10-4} precision for $\underline{u}_1$-velocity component and error in $\underline{u}_1$-velocity component (stationary flow). Four deformed spectral elements,** $N = 6$, $Re = 50$, $\Delta t = 0.015$, **tolerance of** 10^{-11} **and** 10^{-14} **on velocity and pressure resp.**

BP3PC are exactly the same, except for cpu time. It was expected that these three methods would show the same error for the steady solution, since no decoupling errors occur. It is somewhat surprising, however, that the error of 2_{10-4} is obtained after an identical number of iterations. The BP3 method does have a decoupling error for

steady problems. It is clear that PC is the fastest method, followed by BP3PC and BP3. In comparison with the previous test, the performance of the Uzawa method has dropped, due to the absence of fast Helmholtz solvers.

5.4 Conclusions and recommendations

In this chapter we have presented a class of projection methods which have been applied to the spectral-element discretization of the incompressible Navier-Stokes equations in three dimensions. By writing these methods as generalized block LU decompositions, the order of accuracy can be determined and new methods can be derived. In general, the decoupling error does not vanish in the case of steady problems. This can be cured by computing a correction to the pression. Numerical tests showed, however, that this correction damps out very slowly. A combination of a correction-based method (PC or BP3PC) for the first part of the time-integration interval and Uzawa or BP3 for the final part could be an attractive solution.

The decoupling error is proportional to $Re^{-1}\Delta t$ to a certain power, implying that the most interesting application is the computation of flows with Reynolds numbers that are moderate or high (typically $Re > 75$). For Stokes problems, the Uzawa method remains the only serious candidate. Not only accuracy, but also performance depends on $Re^{-1}\Delta t$. In Section 5.3.2 a number of projection methods have been compared as regards the cpu time required to obtain a certain level of accuracy. The Reynolds number for these tests was chosen to be fifty. For higher values, it is no longer accuracy that determines the maximum time step, but stability. In fact, for $Re > 100$, we will see that simulations using the maximum Δt obtained from stability considerations lead to negligible differences in the time accuracy of PC, Uzawa, BP3 and BP3PC.

Finally, we give some recommendations which method to use for some typical situations (see Table V). It concerns heuristic guidelines rather than scientifically established rules.

$Re\Delta t^{-1}$	steady (s) unsteady (u)	accuracy important	fast Helmh. solver	recommended method(s)
small	s+u	yes+no	yes+no	Uzawa
large	s	yes+no	yes+no	PC (BP3PC)
large	u	yes	yes	Uzawa, BP3PC, BP3
large	u	yes	no	BP3PC, BP3, (Uzawa ?)
large	u	no	yes+no	PC (BP3PC)

Table V: **Recommended method as a function of the Reynolds number divided by the time step, the time dependence of the flow, the need of accurate solutions (in time) and the existence of a fast Helmholtz solver.**

Chapter 6

Preconditioning of the pressure operator

In this chapter we focus on the iterative solution of the pressure operator for small values of $Re^{-1}\Delta t$. This corresponds to the simulation of Navier-Stokes flows at moderate or high Reynolds numbers which requires small time steps because of the stability condition due to the explicit treatment of the nonlinear terms. We recall that for this range of values of $Re^{-1}\Delta t$, a simple preconditioner works well for the Helmholtz problem ($P^{-1} = (\mathrm{diag}H)^{-1}$). The diagonal preconditioner for the pressure operator, $P^{-1} = \tilde{B}^{-1}$, is, however, only useful for large values of $Re^{-1}\Delta t$. In the following we will assume that $Re^{-1}\Delta t \ll 1$.

6.1 Introduction

In the previous chapter, we discussed several decoupling methods leading to different pressure operators. We mention S_u,

$$S_u = DH^{-1}D^T, \qquad H = \frac{1}{Re}A + \frac{\beta_s}{\Delta t}B, \tag{6.1}$$

the Uzawa pressure operator; S_{pc}, also referred to as E,

$$S_{pc} = \frac{\Delta t}{\beta_s}E = \frac{\Delta t}{\beta_s}DB^{-1}D^T, \tag{6.2}$$

the pressure-correction or fractional-step operator and, finally, S_{bp},

$$S_{bp} = \frac{\Delta t}{\beta_s} D B^{-1} \left(I - \frac{\Delta t}{\beta_s Re} A B^{-1} + \left(\frac{\Delta t}{\beta_s Re} A B^{-1} \right)^2 \right) D^T, \tag{6.3}$$

the operator related to BP3 and BP3PC, depending on whether the pressure or a pressure correction is computed.

Several remarks have to be made with respect to these three operators. First, S_u, S_{pc}, and S_{bp} are the product of at least three matrices that can not be assembled in a preprocessing stage: Each evaluation requires at least three matrix-vector multiplications, using tensor-product factorizations. Furthermore, S_u, S_{pc}, and S_{bp} are defined on the Gauss-Legendre grid whereas H, B, and A belong to the Gauss-Lobatto-Legendre grid. Hence, D does not only represent derivation, but also interpolation. Since, H, B and A are defined on the velocity grid, they account for boundary conditions and direct stiffness. In fact, they are responsible for the inter-elemental coupling of the pressure operator. Unlike the velocity, there are no pressure nodes shared between two adjacent interfaces, and continuity conditions are absent. It is postulated in [63] that this is the reason that the pressure operator is more difficult to solve than the velocity operator.

The preconditioning of the three pressure operators is related. For the continuous operators, Cahouet and Chabard [12] showed that

$$-\nabla \cdot (\Delta t^{-1} I - Re^{-1} \Delta)^{-1} \nabla = \left(Re^{-1} I^{-1} - \Delta t^{-1} (\nabla^2)^{-1} \right)^{-1}, \tag{6.4}$$

see also Zhou [76]. The same is not true for the discrete operators. Nevertheless, it follows from Equation (6.4) that $P^{-1} S_u \approx I$, with

$$P^{-1} = Re^{-1} \tilde{B}^{-1} + \frac{\beta_s}{\Delta t} (D B^{-1} D^T)^{-1} = Re^{-1} \tilde{B}^{-1} + \frac{\beta_s}{\Delta t} E^{-1}, \tag{6.5}$$

suggesting that P^{-1} is a good preconditioner for S_u. From Chapter 5 it follows that, using an informal notation,

$$S_u = S_{bp} + O(\frac{\Delta t}{Re})^3. \tag{6.6}$$

Hence, by means of Equations (6.2), (6.5), and (6.6) the following relation between S_u,

S_{bp} and S_{pc} emerges:

$$S_u = S_{bp} + O(\frac{\Delta t}{Re})^3 \approx (Re^{-1}\tilde{B}^{-1} + S_{pc}^{-1})^{-1}. \tag{6.7}$$

We conclude that the operators are equivalent for $Re \rightarrow \infty$. In Section 6.2, the preconditioning of S_u will be discussed and extended to S_{pc} and S_{bp} in Section 6.3.

In Timmermans [74], some impressive results are obtained for a finite element preconditioner of the Poisson operator. It is shown that when the linear finite elements are constructed on the Gauss-Lobatto-Legendre mesh, the convergence rate ($K = 1$) is independent of N. By discussing the differences between the approach of Timmermans and ours, we will illustrate that finite element preconditioning is not a good idea in our case. First, we notice that Timmermans uses the pressure-correction decoupling method applied to the space-continuous operators. This implies not only that the pressure can be discretized on the same grid as the velocities, but also that an additional pressure boundary condition is needed (c.f. Chapter 5). The linear finite element operator is then nothing more than a banded matrix with nine (twenty-seven) diagonals in two (resp. three) dimensions, taking into account the pressure boundary condition. The fact that the finite element matrix is spectrally close to the spectral matrix means that the former operator inherits all the difficult modes from the latter. Consequently, iterative methods will converge almost as slow for the finite element as for the spectral element system. Direct methods are therefore to be used to solve the finite element preconditioner. Efficient band solvers exist for the 2D and 3D linear finite element Poisson operator. The inverse is computed in a preprocessing stage, diminishing the cost per pressure iteration to the evaluation of the pressure operator and the (banded) inverse. Their application to large 3D problems (typically $K \gg 1$ and $7 \leq N \leq 12$) is, however, not yet feasible or, at least, very cpu demanding. Furthermore, when decoupling is applied to the space-discrete operators, it is logical to first construct D_{FE} and B_{FE}, the finite element equivalences of D and B respectively, and then to perform the multiplication $S_{FE} = D_{FE}B_{FE}^{-1}D_{FE}^T$. The structure of this product matrix is much more complex than in the case of continuous decoupling, mainly (but not only) due to the inverse of the *non*-diagonal finite element mass matrix. The bandwidth is larger, there are more diagonals involved and sparsity patterns are difficult to recognize.

Summarizing, it is the author's opinion that the only way in which finite element preconditioning of the 3D pressure matrix resulting from a discrete decoupling can be helpful, is to discretize directly the Poisson operator on a coarse grid (say $N = 2$ or $N = 3$). This will yield the loss of the independence of the condition number of N. Also from the parallel-computing viewpoint, finite element preconditioners are not efficient when the granularity imposed by the spectral element discretization is maintained. Moreover, the relation between the condition number of the preconditioned operator and the number of spectral elements K (which is not addressed in [74]), is more important than the relation with N.

The following section appeared as a paper in the Journal of Scientific Computing, Vol. 9, No. 2, 1994, pp 107-122.

6.2 Spectral-Element Preconditioners for the Uzawa Pressure Operator Applied to Incompressible Flows

W. Couzy [1] and M.O. Deville [2]

Abstract: Seven preconditioners for the Uzawa pressure operator are discussed in the context of the spectral-element discretization of unsteady, incompressible flows. An already existing algorithm, based on decomposition of the pressure operator, is modified in order to reduce the number of iterations and cpu time. A two-stage preconditioning technique is set up to solve efficiently problems at high Reynold numbers. The performance of the preconditioners is tested on a number of Stokes and Navier-Stokes problems in three dimensions.

Keywords: Navier-Stokes equations, Spectral-element discretization, Iterative Methods, Preconditioning, Uzawa decoupling

[1]CESAME, Université Catholique de Louvain, Louvain-la-Neuve, Belgium
[2]IMHEF, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

6.2.1 Introduction

In this paper, we focus on the solution of the large algebraic systems that result from the spectral-element discretization [57] of the unsteady, incompressible Navier-Stokes equations in three dimensions. Direct methods are not attractive in this context for various reasons: for example, the inverse is expensive to compute and has to be kept in memory. Moreover, the explicit form of the matrix which is to be inverted is not always known and sparsity patterns are difficult to localize. Finally, the cost to compute an evaluation of the original matrix can be kept low by using tensor products, while this is not the case for the inverse matrix. For these reasons, iterative methods are preferred.

The number of iterations to converge is related to the condition number of the matrix, which, on its turn, depends on the number of spectral elements and on the degree of the approximating polynomials. In order to reduce the number of iterations, an effective preconditioning technique is essential. Here, we define an efficient preconditioner as a matrix for which the condition number of the product of its inverse (which should be cheap to construct) and the original matrix is close to unity. The ultimate goal is to develop a scalable preconditioner, implying that the convergence rate is independent of the number of spectral elements.

Our spectral-element solver for the Navier-Stokes equations is based on the decoupling of the momentum and continuity equations by the Uzawa method [2]. An advantage of this method is that no rediscretization of the original problem is required and that, hence, no extra boundary conditions for the pressure are needed. The non-linear terms are computed by an (explicit) extrapolation scheme, inducing the solution of one system of symmetric, elliptic equations for each velocity component. The solution of the pressure system, which is also symmetric, is much more difficult. This pressure matrix is known to be ill conditioned, especially for high Reynolds numbers [46].

In a recent article, Rønquist [63] introduces a technique to precondition elliptic boundary-value problems. This method is the starting point for this paper and is based on the decomposition of the pressure space into a "global-coarse" and a "fine" subspace. The coarse subspace contains the pressure modes that account for the long-range interactions, and the associated coarse-grid operator is inverted by a direct method. The fine-grid operator is inverted by an iterative method, preconditioned by a local, elemental operator. In order to reduce the number of operations, we propose to construct this preconditioner on a grid consisting of non-deformed spectral elements, even though the original grid can be generally deformed. In this way, we are able to use an efficient algorithm based on the fast diagonalization method (FDM) [44] to invert the local preconditioner. Furthermore, we will study different types of boundary conditions for this preconditioner and show the effect on the rate of convergence. Finally, a framework of two-stage preconditioning accelerates the method for higher Reynolds numbers. All different algorithms have been tested on a series of test problems.

6.2.2 Discrete formulation

The spectral element method was first proposed by Patera [57] and has been elaborated in e.g. Maday and Patera [47], Rønquist [65] and Maday et al [46]. Here, we adopt the

same notation as in the above mentioned papers.

The Navier-Stokes equations are given on a domain Ω by

$$\frac{\partial \underline{u}}{\partial t} - Re^{-1}\Delta \underline{u} + \underline{u} \cdot \nabla \underline{u} + \nabla p = \underline{b}, \tag{6.8}$$

$$-\operatorname{div}\underline{u} = 0 \tag{6.9}$$

and are subjected to no-slip conditions on the boundary $\partial\Omega$:

$$\underline{u} = \underline{0}. \tag{6.10}$$

Here, $\underline{u}$ is the velocity, p is the pressure and $\underline{b}$ is a body force. The Reynolds number $Re = UL/\nu$ is based on a characteristic velocity U, a characteristic length L and the kinematic viscosity ν. When we will speak of a Stokes flow, we leave out the convection term $\underline{u} \cdot \nabla \underline{u}$ and take $Re = 1$ in (6.8). The governing equations are written in their variational form: Find $(\underline{u}, p) \in X_g \times M$ such that $\forall \underline{w} \in X_g, \forall q \in M$

$$\left(\frac{\partial \underline{u}}{\partial t}, \underline{w}\right) + Re^{-1}\left(\nabla \underline{u}, \nabla \underline{w}\right) + \left(\underline{u} \cdot \nabla \underline{u}, \underline{w}\right) - \left(p, \operatorname{div}\underline{w}\right) = \left(\underline{b}, \underline{w}\right) \tag{6.11}$$

$$-\left(q, \operatorname{div}\underline{u}\right) = 0, \tag{6.12}$$

where

$$\forall \phi, \psi \in \mathcal{L}^2(\Omega) \qquad (\phi, \psi) = \int_\Omega \phi(\underline{x})\psi(\underline{x})d\underline{x} \qquad \underline{x} = (x_1, x_2, x_3) \in \Omega. \tag{6.13}$$

The space X_g for the velocity and the space M for the pressure are given by

$$X_g = \{v \in |\mathcal{H}^1(\Omega \cup \partial\Omega)|^3, v \text{ vanishes on } \partial\Omega\} \tag{6.14}$$

$$M = \mathcal{L}_0^2(\Omega) = \{\phi \in \mathcal{L}^2(\Omega) \quad \int_\Omega \phi(\underline{x})dx = 0\}. \tag{6.15}$$

$\mathcal{H}^1(\Omega)$ is the space of all functions that are square integrable and whose first-order derivatives are also square integrable over Ω. The space $\mathcal{L}^2(\Omega)$ is the space of all square integrable functions over Ω.

The domain Ω is broken up into K spectral elements Ω_k on which $\underline{u}$ and p are expanded in Nth-order (resp. $(N-2)$th-order) tensor-product Lagrangian bases. Tensor-product numerical quadrature rules based on the Gauss-Lobatto-Legendre grid are used

to evaluate the integrals in (6.11,6.12), except for $(p, \mathrm{div}\underline{w})$ and $(q, \mathrm{div}\underline{u})$, which are computed by tensor-product quadrature rules on the Gauss-Legendre grid. The chosen time scheme is Euler Backward, where the non-linear terms at $t = t_{n+1}$ are approximated by third-order extrapolation in time. These terms are added, together with $(\Delta t)^{-1}\underline{u}^{n+1}$, and the body force $\underline{b}$ to become the new right-hand side vector $\underline{f}^{n+1}$. The discrete formulation of (6.11,6.12) reads:

$$\frac{1}{\Delta t} B\underline{u}_i^{n+1} + Re^{-1} A\underline{u}_i^{n+1} - D_i^T p^{n+1} = B\underline{f}_i^{n+1} \qquad i = 1,2,3 \tag{6.16}$$

$$- D_i \underline{u}_i^{n+1} = 0. \tag{6.17}$$

Here, B is the diagonal mass matrix, A is the discrete Laplace operator, D_i is the discrete divergence operator and the superscript T indicates the transpose. The velocities are in $X_{g,h}$ and the pressure is in M_h (according to [7]), defined as

$$X_{g,h} = X_g \cap \mathcal{P}^3_{N,K}(\Omega) \tag{6.18}$$

$$M_h = M \cap \mathcal{P}_{N-2,K}(\Omega), \tag{6.19}$$

with $\mathcal{P}_{N,K} = \{\phi \in \mathcal{L}^2(\Omega); \phi_{|\Omega_k} \text{ is a polynomial of degree less than or equal to } N\}$. The Gauss-Lobatto-Legendre/Gauss-Legendre discretization for velocities/pressure guarantees the absence of spurious pressure modes [7].

Starting from the discrete equations (6.16,6.17), the Uzawa algorithm [2] is obtained by multiplication of the momentum equations by $D_i H^{-1}$, with

$$H := (Re^{-1} A + \Delta t^{-1} B). \tag{6.20}$$

We obtain

$$Sp^{n+1} = -D_i H^{-1} B\underline{f}_i^{n+1} \tag{6.21}$$

$$H\underline{u}_i^{n+1} = D_i^T p^{n+1} + B\underline{f}_i^{n+1} \qquad i = 1,2,3. \tag{6.22}$$

where

$$S := D_i H^{-1} D_i^T. \tag{6.23}$$

Tensor products reduce the cost of matrix-vector multiplications to $O(KN^4)$. Equation (6.21) is solved by the preconditioned conjugate gradient method (PCGM). Clearly, the Helmholtz operator H has to be inverted within each PCGM-iteration. This is done by a PCGM, preconditioned by the diagonal of H.

6.2.3 Pressure preconditioners

For large values of $Re^{-1}\Delta t$, the pressure operator $\tilde{B}^{-1}S$ is identity like, where $\tilde{B}$ is the diagonal mass matrix defined on the Gauss-Legendre grid [46]. This suggests that $\tilde{B}$

is a good candidate for a preconditioner. Therefore, we define our first preconditioner $P1$ as

$$P1^{-1} = \tilde{B}^{-1}. \tag{6.24}$$

Throughout this report, we will introduce preconditioners by defining their inverse matrices, because the PCGM requires at each iteration a multiplication by the inverse of the preconditioner. This matrix-vector multiplication, together with the multiplication by the original matrix, are the basic computational units of any iterative method. In this paper, we will often use the term "evaluation" when we speak about these matrix-vector multiplications. For small values of $Re^{-1}\Delta t$, $\tilde{B}^{-1}S$ is no longer identity like. In this case, Maday et al [46] found that S is spectrally close to $D_i B^{-1} D_i^T$. We introduce the symbol E to represent this pseudo-Laplace operator:

$$E := D_i B^{-1} D_i^T. \tag{6.25}$$

Note that B^{-1} is defined on the Gauss-Lobatto-Legendre grid and has zeroes at entries corresponding to Dirichlet boundary conditions. A preconditioner P that works for both cases is defined by (see e.g. [46])

$$P^{-1} = Re^{-1}\tilde{B}^{-1} + \Delta t^{-1} E^{-1}. \tag{6.26}$$

The difficulty introduced by this preconditioner is that E itself is also ill conditioned, implying that the problem in solving S is transferred to the iterative method to solve E.

Another important remark is made by Rønquist [63], who pointed out that the pressure Poisson operator is more difficult to invert than the standard Laplace operator, since the first one is in $\mathcal{L}^2$ and the latter in $\mathcal{H}^1$. Therefore, Rønquist proposes to solve the pressure in two different pressure subspaces. The first set of equations serves to build up a global "skeleton" for the pressure. This coarse, global system contains the constant pressure levels inside each element. The second subspace represents the local variations of the pressure with respect to the global frame. Applied to our case, this method consists in decomposing M_h into two disjoint parts $M_{h,0}$ and $M_{h,N}$

$$M_{h,0} = \mathcal{L}_0^2(\Omega) \cap \mathcal{P}_{0,K}(\Omega), \tag{6.27}$$

$$M_{h,N} = \mathcal{L}_{0,K}^2(\Omega) \cap \mathcal{P}_{N-2,K}(\Omega), \tag{6.28}$$

where

$$\mathcal{L}_{0,K}^2(\Omega) = \{\phi \in \mathcal{L}_0^2(\Omega_k), k = 1, ..., K\}, \tag{6.29}$$

such that

$$p^{n+1} = (Jp_0 + p_N) \in M_h = M_{h,0} \oplus M_{h,N}. \tag{6.30}$$

Here, p_0 accounts for the "skeleton" and p_N for the local variation. The operator J maps the local constant pressure level on the space M_h. Instead of solving (6.21), we now derive from

$$S(Jp_0 + p_N) = -D_i H^{-1} B \underline{f}_i^{n+1} \tag{6.31}$$

two equations to compute p_0 and p_N. To this end, we multiply the left- and right-hand side of (6.31) by J^T and solve for p_0:

$$p_0 = (J^T S J)^{-1} J^T (-D_i H^{-1} B \underline{f}_i^{n+1} - S p_N). \tag{6.32}$$

We substitute (6.32) in (6.31) to find the following expression for p_N

$$(I - SJ(J^T S J)^{-1} J^T) S p_N = -(I - SJ(J^T S J)^{-1} J^T) D_i H^{-1} B \underline{f}_i^{n+1} \tag{6.33}$$

We introduce the coarse-grid operator S_0 and the fine-grid operator S_N as:

$$S_0 = J^T S J, \quad S_N = (I - S J S_0^{-1} J^T) S \tag{6.34}$$

and rewrite (6.33,6.32) as

$$S_N p_N = -(I - S J S_0^{-1} J^T) D_i H^{-1} B \underline{f}_i^{n+1} \tag{6.35}$$

$$S_0 p_0 = J^T (-D_i H^{-1} B \underline{f}_i^{n+1} - S p_N). \tag{6.36}$$

Since (6.36) contains only K unknowns, a direct solution method is preferred. On the other hand, an iterative method (i.e. the conjugate gradient method) is used to solve (6.35). The key point in this algorithm is that (6.35) can be efficiently preconditioned by $block(S)$, corresponding to the elemental local S-matrices, where homogeneous Dirichlet boundary conditions are imposed (to H^{-1}) on all boundaries and interfaces. The advantages of this approach is that the preconditioner respects

$$ker(block(S)) = ker(S_N) \tag{6.37}$$

and is completely decoupled per element. The latter aspect is important for the implementation on parallel computers (see [28]).

In practice, $block(S)$ is hard to construct. Therefore, we replace $block(S)$ by $(\Delta t^{-1} block(E)^{-1} + Re^{-1}\tilde{B})^{-1}$, which is, according to (6.26), spectrally close. We remark that we do no longer have a compatibility between the null spaces of the preconditioner and S_N, but this is not believed to prohibit or slow down the convergence of the iterative method. In Rønquist [63], in a slightly different context, the LL^T decomposition of $block(E)$ is computed on each element by a direct method. We remark that the cost to compute this decomposition is of order $O(K(N-2)^9)$ and that the cost of each back substitution is of order $O(K(N-2)^6)$. In order to reduce the cost of a back substitution to an order of $O(K(N-2)^5)$, Fischer and Rønquist [28] construct a $block(E)$ matrix on a finite-element mesh. Here, we propose to use the FDM to bring the cost back to $O(K(N-2)^4)$. A second advantage of the FDM is that the inverse matrix (or LL^T decomposition) does not have to be kept in memory.

6.2.4 Preconditioners based on FDM

The FDM , which we will use to construct our preconditioner, goes back to Lynch et al [44], who gave explicit formulas for the inverse of a tensorizable and separable operator,

say L. In particular, they showed that when such an operator L is written as

$$L = (I \otimes I \otimes L_x + I \otimes L_y \otimes I + L_z \otimes I \otimes I), \tag{6.38}$$

its inverse is given by

$$\begin{aligned} L^{-1} &= (P_z \otimes P_y \otimes P_x)\Lambda^{-1}(P_z^{-1} \otimes P_y^{-1} \otimes P_x^{-1}), \\ \Lambda &= I \otimes I \otimes \Lambda_x + I \otimes \Lambda_y \otimes I + \Lambda_z \otimes I \otimes I. \end{aligned} \tag{6.39}$$

Here, $A \otimes B$ denotes the tensor product of A and B. The matrices P_x, P_y and P_z represent the eigenvector decomposition of the one-dimensional operators L_x, L_y and L_z, such that

$$P_x^{-1} L_x P_x = \Lambda_x, \qquad P_y^{-1} L_y P_y = \Lambda_y, \qquad P_z^{-1} L_z P_z = \Lambda_z, \tag{6.40}$$

with Λ_x, Λ_y and Λ_z diagonal matrices having as entries the eigenvalues of the corresponding operators. The interest of this method becomes clear when the cost of an evaluation of $L^{-1}\underline{v}$ is compared to the matrix-vector multiplication $L\underline{v}$, the basic ingredient of any iterative method. Using (6.39), $L^{-1}\underline{v}$ needs twice as many operations as the matrix-vector product. So the inverse is computed at the price of two matrix-vector products.

In the following, we will show how the FDM can be used to invert $block(E)$, i.e. how to solve a system like $block(E)x = g$ on an arbitrary, non-deformed spectral element. If we denote by J_x^T, J_y^T, J_z^T the one-dimensional interpolation operators from the GLL to the GL grid; by w_x, w_y, w_z the weights on the GL grid; by $\tilde{w}_x, \tilde{w}_y, \tilde{w}_z$ the inverses of the weights on the GLL grid with zeroes at the first and Nth position, corresponding to homogeneous Dirichlet conditions; by D_x, D_y, D_z the one-dimensional derivation operators on the GLL grid, based on information on the GL grid, then we can write $block(E)x = g$ on a nondeformed spectral element with dimensions $a \times b \times c$ as

$$\begin{aligned} (w_z \otimes w_y \otimes w_x)\{&\frac{bc}{2a}(J_z \otimes J_y \otimes D_x)(\tilde{w}_z \otimes \tilde{w}_y \otimes \tilde{w}_x)(J_z^T \otimes J_y^T \otimes D_x^T) \quad + \\ &\frac{ac}{2b}(J_z \otimes D_y \otimes J_x)(\tilde{w}_z \otimes \tilde{w}_y \otimes \tilde{w}_x)(J_z^T \otimes D_y^T \otimes J_x^T) \quad + \\ &\frac{ab}{2c}(D_z \otimes J_y \otimes J_x)(\tilde{w}_z \otimes \tilde{w}_y \otimes \tilde{w}_x)(D_z^T \otimes J_y^T \otimes J_x^T)\}(w_z \otimes w_y \otimes w_x)x \quad = \quad g. \end{aligned} \tag{6.41}$$

This expression can be rewritten as

$$\begin{aligned} \{I \otimes I \otimes (a^2 J_x \tilde{w}_x J_x^T)^{-1} D_x \tilde{w}_x D_x^T + I \otimes (b^2 J_y \tilde{w}_y J_y^T)^{-1} D_y \tilde{w}_y D_y^T \otimes I \quad + \\ (c^2 J_z \tilde{w}_z J_z^T)^{-1} D_z \tilde{w}_z D_z^T \otimes I \otimes I\}(w_z \otimes w_y \otimes w_x)x \quad = \end{aligned}$$

$$\frac{2}{abc}(w_z J_z \tilde{w}_z J_z^T)^{-1} \otimes (w_y J_y \tilde{w}_y J_y^T)^{-1} \otimes (w_x J_x \tilde{w}_x J_x^T)^{-1} g. \qquad (6.42)$$

Clearly, the term between braces is written under the form (6.38) and can be inverted by the FDM. The other factors consisting of tensor products of the form $(A \otimes B \otimes C)^{-1}$ are trivially inverted as $A^{-1} \otimes B^{-1} \otimes C^{-1}$. After rearrangement of the terms, the cost to evaluate the inverse is expressed by a polynomial in N, the leading term of which is $6K(N-2)^4$.

At this point, two remarks have to be made. First, the matrix $block(E)$ can only be inverted by the FDM on non-deformed grids. Otherwise, the operator $block(E)$ is not separable. In the case of deformed geometries, we construct therefore $block(E)$ on the corresponding non-deformed spectral elements. Numerical results, which we will present in Section 6.2.6, show that this operator is still a good preconditioner, although it does not take the deformation of the mesh into account. Second, the unit vector belongs to the kernel on each spectral element, resulting in a zero eigenvalue in the eigenvalue/eigenvector decomposition of each one-dimensional operator [63]. This is fixed by replacing the zero eigenvalue in one direction by 10^{-2}. Empirically, we found that the number of PCGM iterations to solve (6.35) is practically the same when $block(E)$ is inverted by FDM (with modified eigenvalue) as when $block(E)$ is, on its turn, inverted by an iterative method.

The above constructed preconditioner is called P2 (see Figure 6.1). Instead of imposing homogeneous Dirichlet boundary conditions to $block(E)$ (putting zeroes at entries of B^{-1} corresponding to positions at the boundaries and interfaces), we propose to impose nothing at interfaces, as is the case for Neumann boundary conditions of the type $\frac{\partial u}{\partial n} = 0$. For $K > 1$, this strategy automatically solves the problem of the zero eigenvalue. In many cases, less iterations are required for this preconditioner which we will call P3. When P3 is directly applied to precondition S (instead of S_N), we speak of P4 (see Figure 6.1). The advantage of P4 is that only one S-evaluation is needed for every PCGM iteration, whereas P2 and P3 require two S-evaluations (see (6.34,6.35)).

6.2.5 Two-stage preconditioners

In the previous paragraph, three preconditioners have been proposed that require at each iteration one or two evaluations of the matrix S. These evaluations are very expensive since a Helmholtz equation has to be solved (see (6.21)). Therefore, we propose a *two-stage* preconditioning technique which is especially interesting for small values of $Re^{-1}\Delta t$. The first stage consists in preconditioning the operator S by (6.26). We recall that when this preconditioner is used, few iterations are needed to converge ($Re^{-1}\Delta t \ll 1$). In order to invert the preconditioner, a second preconditioning stage is introduced, applying the same technique to E as has been applied to S in the previous paragraphs. More precisely, the evaluation of (6.26) requires the solution of systems like Eq =rhs. This equation is rewritten as $E(Jq_0 + q_N)$ =rhs, with $q_0 \in M_{h,0}$ and $q_N \in M_{h,N}$. Analogously to (6.34), we introduce the coarse-grid operator E_0 and the

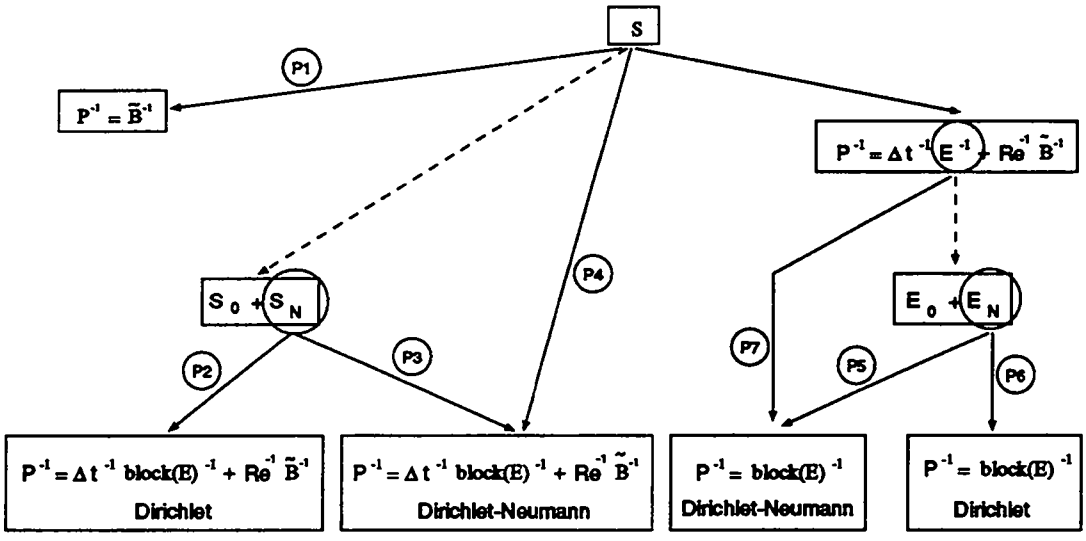

Figure 6.1: **Overview of the seven preconditioners, indicated by P1,..,P7. The dashed arrows point to splitting of the pressure space in a coarse-global and a fine subspace. The straight arrows represent the preconditioners. The one-stage preconditioners (P1,P2,P3 and P4) can be found at the left; the two-stage preconditioners at the right. The last line indicates the type of boundary conditions that are applied to solve** $block(E)$.

fine-grid operator E_N as:

$$E_0 = J^T E J, \quad E_N = (I - EJE_0^{-1}J^T)E \tag{6.43}$$

and solve

$$E_N q_N = (I - EJE_0^{-1}J^T)\text{rhs} \tag{6.44}$$
$$E_0 q_0 = J^T(\text{rhs} - Eq_N). \tag{6.45}$$

In this way we evaluate S only a few times (first stage), at the price of many, but cheap evaluations of E_N. The matrix E_N, on its turn, is preconditioned by $block(E)$; the second stage. Again, we can apply homogeneous Dirichlet (P6) or Neumann-Dirichlet (P5) boundary conditions to $block(E)$. If we apply P5 directly to E, instead of to E_N, we speak of P7. All preconditioners are schematically represented in Figure 6.1.

So the two-stage preconditioning technique consists in the introduction of an intermediate preconditioner, with the aim to reduce the number of expensive S-evaluations.

6.2.6 Numerical results

In order to test the performance of the six preconditioners and P1, which we defined previously as $(P1)^{-1} = \tilde{B}^{-1}$ (6.24), we solve a problem with analytical solution

$$\underline{u}(x_1,x_2,x_3) = \left(-cos(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_2), sin(\frac{\pi}{2}x_1)cos(\frac{\pi}{2}x_2), 0\right)^T \quad (6.46)$$

$$p(x_1,x_2,x_3) = -\pi sin(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_2) \quad (6.47)$$

on the domain $[-1,1] \times [0,16] \times [-1,1]$, for a Navier-Stokes flow with $Re = 100$, $\Delta t = 0.01$. Various values of K have been tested. In Table I, we compare

$K_x \times K_y$	preconditioner	♯ iterations	cpu time
1×8	P1	132	2'12"
1×8	P2	19	42"
1×8	P3	21	44"
1×8	P4	22	27"
1×8	P5	4(54)	28"
1×8	P6	4(56)	28"
1×8	P7	4(54)	20"
2×8	P1	223	8'38"
2×8	P2	43	3'46"
2×8	P3	25	2'26"
2×8	P4	23	50"
2×8	P5	5(60)	53"
2×8	P6	5(128)	1'24"
2×8	P7	5(59)	32"
3×12	P1	323	35'36"
3×12	P2	60	15'39"
3×12	P3	50	13'07"
3×12	P4	69	6'08"
3×12	P5	7(133)	3'38"
3×12	P6	6(212)	4'59"
3×12	P7	7(179)	2'33"

Table I: **Number of iterations and cpu time for first time step of the Navier-Stokes problem; $Re^{-1}\Delta t = 10^{-4}$, N=9, tolerance for convergence is set to 10^{-8}. K_x denotes the number of elements in x-direction, K_y denotes the number of elements in y-direction. The number of elements in z-direction is always one. The number of iterations corresponds to the number of S-evaluations (P1,P4,P5,P6,P7) or S_N-evaluations (P2,P3). For P5 and P6, the number of E_N-evaluations is indicated between parentheses. For P7, the number of E-evaluations is indicated between parentheses.**

the number of iterations for the pressure residual to converge to 10^{-8} for the first time step. It should be noted, that this residual is precisely the discrete divergence $-D_i\underline{u}_i$.

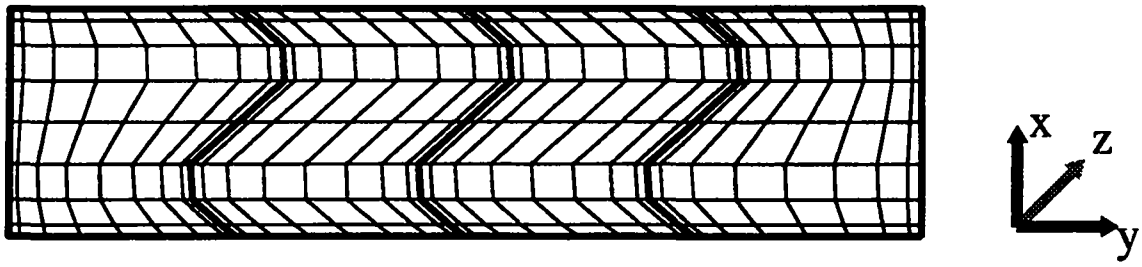

Figure 6.2: A projection ($z = 0$) of the deformed mesh ($\alpha = 0.5$). The geometry is contained in the box $[-1,1] \times [0,8] \times [-1,1]$

In addition, we give the cpu time measured on one processor of the Convex 3820 vector computer. Although it is difficult to draw general conclusions from Table I, we remark that in terms of cpu time, the two-stage preconditioning scheme is always advantageous. Furthermore, the application of Dirichlet *and* Neumann boundary conditions reduces the number of iterations. We also conclude that direct preconditioning of the operator (P4 and P7) is faster than via S_N, respectively E_N. Finally, we remark that the rate of convergence depends on the number of spectral elements K. Since this test considers only the first time step, start-up time (to compute eigenvalue/eigenvector decomposition and S_0 or E_0) takes a non-negligible amount of the total cpu time. However, the same test extended to more time steps shows the same pattern.

Next, we test the influence of the deformation of the grid on the rate of convergence. We take the previous test problem (6.46,6.47) on a geometry $[-1,1] \times [0,8] \times [-1,1]$, decomposed in $1 \times 4 \times 1$ spectral elements. The interfaces are deformed in y-direction by the function $\alpha \sin(\pi x) \sin(\pi z)$. Table II displays the results for $\alpha = 0, \alpha = 0.24$ and $\alpha = 0.5$.

	Preconditioner						
α	P1	P2	P3	P4	P5	P6	P7
0	90; 45"	18; 16"	15; 14"	15; 8"	4(43); 10"	4(55); 11"	4(43); 6"
0.24	121; 59"	37; 32"	31; 26"	34; 15"	4(76); 14"	5(102); 16"	4(76); 8"
0.5	168; 1'35"	55; 54"	51; 43"	51; 26"	5(128); 21"	5(164); 25"	5(126); 12"

Table II: Number of iterations and cpu time for the first time step of the Navier-Stokes problem; $Re^{-1}\Delta t = 10^{-4}$, N=9, tolerance for convergence is set to 10^{-8}. Four deformed spectral elements. α indicates the rate of deformation. The number of iterations corresponds to the number of S-evaluations (P1,P4,P5,P6,P7) or S_N-evaluations (P2,P3). For P5 and P6, the number of E_N-evaluations is indicated between parentheses. For P7, the number of E-evaluations is indicated between parentheses.

For the latter value of α, a projection of the grid ($z = 0$) can be found in Figure 6.2. The results confirm more or less the conclusions of the previous test. The cpu time for the preconditioners based on the two-stage preconditioning technique does not suffer

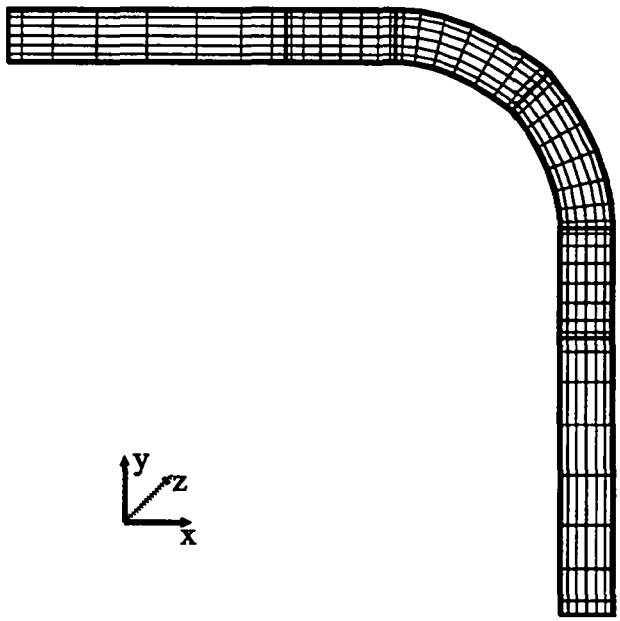

Figure 6.3: **A projection in z-direction of the curvy channel. The geometry is contained in the box** $[0,11] \times [0,11] \times [0,1]$.

much from the increasing deformation. This is explained by the fact that the number of expensive S-evaluations only grows by one.

The third test is a Poiseuille flow in a curved channel, $K = 6, N = 9$. Boundary conditions at $z = 0$ and $z = 1$ are $\frac{\partial \underline{u}_1}{\partial \underline{n}} = \frac{\partial \underline{u}_2}{\partial \underline{n}} = \underline{u}_3 = 0$. At the inflow, a parabolic

velocity profile is imposed; at the outflow we impose the outflow conditions $\underline{u}_1 = \frac{\partial \underline{u}_2}{\partial \underline{n}} = \underline{u}_3 = 0$. The initial solution at $t = 0$ is given by $\underline{u} = \underline{0}$ and $p = 0$. The grid is displayed in Figure 6.3. The results for a Stokes flow at $t = 0.5$ ($\Delta t = 0.5$) can be found in Table III.

preconditioner	no prec	P1	P2	P4	P6	P7
iterations	200	73	55	52	26(996)	26(1257)
cpu time	1h 28'49"	58'37"	1h 12'54"	38'22"	19'55"	20'12"

Table III: **Number of iterations and cpu time to compute one time step. Stokes flow through a curved channel.** $K = 6, N = 9, \Delta t = 0.5$. **Tolerance for convergence is set to** 10^{-8}. **The number of iterations corresponds to the number of** S**-evaluations (P1,P4,P6,P7) or** S_N**-evaluations (P2). For P6, the number of** E_N**-evaluations is indicated between parentheses. For P7, the number of** E**-evaluations is indicated between parentheses.**

As expected from the discussion in paragraph 3, the simple preconditioner P1 performs rather well. The two-stage preconditioners need less cpu time than the one-stage preconditioners. This is explained by the fact for large values of Δt, the Helmholtz equation H is difficult to solve.

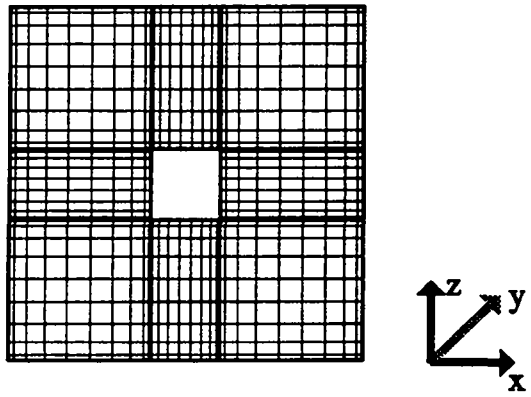

Figure 6.4: A projection in y-direction of regularized driven cavity with a hole. The geometry is contained in the box $[-1,1]^3$. The dimensions of the hole are $[0.4,0.6] \times [0,1] \times [0.4,0.6]$.

The converged Stokes solution is used as the initial guess for the Navier-Stokes problem $Re = 100, \Delta t = 0.0075, K = 6, N = 9$. In Table IV, we give cpu timings for the first ten time steps. The interest of the two-stage technique is clearly shown.

preconditioner	no prec	P2	P4	P6	P7
cpu time	35'59"	5'3"	4'	1'32"	1'21"

Table IV: Cpu time to compute solution at $t = 0.075$. Navier-Stokes flow through a curved channel; with Stokes flow as initial solution. $K = 6, N = 9, \Delta t = 0.0075$. Tolerance for convergence is set to 10^{-8}.

The last test consists of the computation of a Stokes and a Navier-Stokes flow in a regularized driven cavity with a hole. The dimensions of the cube are $[0,1]^3$ with the hole in the center $[0.4,0.6] \times [0,1] \times [0.4,0.6]$. The number of spectral elements is eight and the polynomial degree is ten. Figure 6.4 shows a projection of the cube in the y-direction. Homogeneous Dirichlet boundary conditions are applied everywhere, except at the top plane ($z = 1$), where $\underline{u}_1 = 16xy(x-1)(y-1)$. The Helmholtz problem is solved by a combination of a Schur complement method and FDM, as discussed in Couzy and Deville [19]. For the current Stokes problem, this implies that the Helmholtz problem is solved about 15 times faster than with the standard diagonal preconditioner. For the Navier-Stokes problem, speed ups of about a factor four have been reported. The results can be found in Tables V and Table VI. Table V clearly shows that, for a Stokes flow, the two-stage preconditioners (P6,P7) do not work well. This can be explained by the fact that, because of the fast Helmholtz solver, an S-evaluation is almost as expensive as an E-evaluation. For both the Stokes and the Navier-Stokes flow, we observe a rather poor performance of the preconditioners based on homogeneous boundary conditions (P2,P6).

preconditioner	no prec	P1	P2	P4	P6	P7
iterations	123	24	87	24	24	25
cpu time	5'54"	1'12"	8'33"	1'29"	8'24"	6'10"

Table V: **Number of iterations and cpu time to compute solution at $t = 1$. Stokes flow in a driven cavity with hole. $K = 8, N = 10, \Delta t = 0.5$. Tolerance for convergence is set to 10^{-8}. The number of iterations corresponds to the number of S-evaluations (no prec.,P1,P4,P6,P7) or S_N-evaluations (P2).**

preconditioner	no prec	P1	P2	P4	P6	P7
iterations	1998	1597	443	393	48	47
cpu time	46'35"	39'31"	18'7"	9'58"	13'03"	7'34"

Table VI: **Number of iterations and cpu time to compute solution at $t = 0.035$. Navier-Stokes flow in a driven cavity with hole. Initial solution is Stokes flow. $K = 8, N = 10, \Delta t = 0.0035$. Tolerance for convergence is set to 10^{-8}. The number of iterations corresponds to the number of S-evaluations (no prec.,P1,P4,P6,P7) or S_N-evaluations (P2).**

6.2.7 Conclusions

In the previous paragraphs, we have studied a number of ways to precondition the pressure operator that results when the Uzawa technique is applied to the three-dimensional, unsteady, incompressible Navier-Stokes equations. The fast diagonalization technique is used to reduce the cost of the construction and evaluation of the preconditioner. Since the FDM can only be applied on parallelepipedic spectral elements, the preconditioner is constructed on a nondeformed grid. Numerical tests show that the number of iterations is still significantly reduced in the case of deformed geometries. All preconditioners are local, elemental matrices and have great potential on parallel computers with shared or distributed memory.

For problems at high Reynolds numbers (i.e. $Re^{-1}\Delta t \ll 1$), a framework of two-stage preconditioners has been set up. This concept reduces the number of expensive S-evaluations. A considerable gain in cpu time is observed. The power of this technique is that the number of iterations decreases when the physical complexity of the problem increases (i.e. higher Reynolds numbers, implying ill-conditioned pressure operators).

The numerical tests point out that P4 (or P7) is the best preconditioner for problems at large values of $Re^{-1}\Delta t$, whereas P7 should be used for problems at high Reynold numbers. Both P4 and P7 are based on a direct application of the block preconditioner to the original operator S, resp. E (and not to S_N, resp. E_N). Moreover, P4 and P7 are based on Dirichlet-Neumann, and not homogeneous Dirichlet, boundary conditions. Using P7, a relatively large problem consisting of 77823 degrees of freedom has been solved in about three minutes per time step ($Re = 100, \Delta t = 0.01$, Convex 3820 computer, obtained performance about 20-25 Mflops). This opens the way to large-scale simulations of complex three-dimensional unsteady flows on modern (parallel)

computers.

Aknowledgement. The above text presents research results of the Belgian Incentive Program "Information Technology - Computer Science of the Future", initiated by the SPPS (Services du Premier Ministre. Programmation de la Politique Scientifique). The authors wish to thank V. Warichet for providing his code.

6.3 Preconditioning of different pressure operators

We recall from the previous section that the preconditioning of the Uzawa pressure operator S_u relies on the idea of Cahouet and Chabard [12] who stated that

$$S_u^{-1} \approx Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}E^{-1}. \tag{6.48}$$

This leads to the observation that P, defined by

$$P^{-1} = Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}E^{-1}, \tag{6.49}$$

is a good preconditioner for all values of $Re^{-1}\Delta t$. The problem of inverting S_u is transferred to E. We have seen that $block(E)^{-1}$ can be constructed by FDM and neglecting interelemental coupling on nondeformed grids. This observation, together with Equations (6.48) and (6.49) has led to a set of preconditioners for S_u. We give three of them, where we have modified the notation to fit better in the frame of this section and to distinguish the different operators. First, we replace E^{-1} in Equation (6.49) by $block(E)^{-1}$ and precondition S_u directly by P_u:

$$P_u^{-1} = Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}block(E)^{-1} \longrightarrow S_u \qquad \left[P_u^{-1} \rightarrow S_u\right]. \tag{6.50}$$

In the following we will use the symbolic notation that is given between the brackets; the arrow points from the preconditioner to the operator. Since $\frac{\Delta t}{\beta_s}E = S_{pc}$, the following preconditioner P_{pc} for the pressure correction operator S_{pc} is immediate

$$P_{pc}^{-1} = block(E)^{-1} \longrightarrow \frac{\Delta t}{\beta_s}E = S_{pc} \qquad \left[P_{pc}^{-1} \rightarrow_E S_{pc}\right]. \tag{6.51}$$

Rønquist [63] introduced the idea to solve the pressure in a coarse and a fine subspace. This leads to an equivalent formulation for P,

$$P^{-1} = Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}E_{0,N}^{-1}, \tag{6.52}$$

where $E_{0,N}^{-1}$ $(= E^{-1})$ means that the system $Ep = g$ is solved by

$$E_N p_N = (I - EJE_0^{-1}J^T)g \qquad E_0 p_0 = J^T(p - Ep_N), \tag{6.53}$$

with

$$E_0 = J^T EJ, \qquad E_N = (I - EJE_0^{-1}J^T)E, \qquad p = p_N + Jp_0. \tag{6.54}$$

A two-stage preconditioning technique has been introduced in order to avoid expensive S_u evaluations:

$$block(E)^{-1} \longrightarrow_E Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}E^{-1} \longrightarrow S_u \qquad \left[P_{pc}^{-1} \rightarrow_E P^{-1} \rightarrow S_u\right]. \tag{6.55}$$

In the same way we define the two-stage preconditioner based on the idea of Rønquist

$$block(E)^{-1} \longrightarrow_{E_N} Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}E_{0,N}^{-1} \longrightarrow S_u \qquad \left[P_{pc}^{-1} \rightarrow_{E_N} P^{-1} \rightarrow S_u\right]. \tag{6.56}$$

It is then a logical step to precondition S_{pc} in two subspaces:

$$P_{pc}^{-1} = block(E)^{-1} \longrightarrow_{E_N} \frac{\Delta t}{\beta_s}E_{0,N} = S_{pc} \qquad \left[P_{pc}^{-1} \rightarrow_{E_N} S_{pc}\right]. \tag{6.57}$$

Equation (6.7) shows us that $(Re^{-1}\Delta t \ll 1)$ S_{bp} and S_u are almost identical, suggesting the following preconditioner for S_{bp}:

$$P_u^{-1} = Re^{-1}\tilde{B}^{-1} + \frac{\beta_s}{\Delta t}block(E)^{-1} \longrightarrow S_{bp} \qquad \left[P_u^{-1} \rightarrow S_{bp}\right]. \tag{6.58}$$

The BP3 pressure operator can also be solved in two subspaces, but this technique has not been implemented. We close this overview with some remarks about the boundary conditions. As has been said in the previous section, it is advantageous to apply mixed boundary conditions to the elemental $block(E)$ matrix (i.e. to B^{-1}). More precisely, in the case of Dirichlet boundary conditions, the corresponding entry in B^{-1} is set to zero. At boundaries corresponding to Neumann conditions or an interelemental interface, B^{-1} is not changed. This complies with the weak imposition

	No prec.	$(P_u)^{-1} \rightarrow S_u$	$P_{pc}^{-1} \rightarrow_E$ $P^{-1} \rightarrow S_u$	$P_{pc}^{-1} \rightarrow_{E_N}$ $P^{-1} \rightarrow S_u$
$\alpha = 0$	400 ; 184.8	88 ; 40.4	(5)341 ; 5.8	(5)211 ; 6.6
$\alpha = 0.1$	411 ; 199.4	118 ; 57.6	(7)576 ; 9.3	(7)478 ; 12.8
$\alpha = 0.5$	450 ; 320.9	295 ; 212.0	(9)2194 ; 27.3	(9)1362 ; 32.0

Table VII: **Uzawa method. Number of iterations and cpu time for one time step ($\Delta t = 0.01$, $Re = 50$, $K = 16$, $N = 8$) as a function of the deformation of the geometry (α) and preconditioner. For the two-stage preconditioner, the number of "stage one" iterations (i.e. the number of multiplications by P^{-1}) is given between parentheses. The cpu time is normalized with respect to the preconditioned PC method ($\alpha = 0$), see Table PC. Tolerance 10^{-9} for pressure and 10^{-11} for the velocity.**

of $\partial/\partial n = 0$. The $block(E)$ operators that are used for the preconditioning of E_N will be submitted to homogeneous boundary conditions for every boundary and interface [63]. Consequently, the matrix $block(E)$ has a zero eigenvalue on each spectral element and the FDM can no longer be used. Two solutions have been proposed. First, the zero eigenvalue can be replaced by 10^{-2}. In order to investigate the influence on the number of iterations to solve E, we compared FDM to an iterative method to solve the $block(E)$ preconditioner. Of course, an iterative method is very unattractive because of its excessive cpu requirements, but it gives the exact "inverse" of $block(E)$ (at a precision of 10^{-12}), which is not the case for the FDM, because of the modified eigenvalue. It turned out that the two different ways of inverting the preconditioner hardly influences the number of iterations. Sometimes, FDM led even to slightly faster convergence than the "exact" iterative method. Second, Neumann boundary conditions can be applied at interfaces. In this sequel, we have adapted the first solution.

	No prec.	$P_{pc}^{-1} \rightarrow_E S_{pc}$	$P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$
$\alpha = 0$	424 ; 3.9	87 ; 1.0	58 ; 1.4
$\alpha = 0.1$	437 ; 4.0	121 ; 1.3	115 ; 2.4
$\alpha = 0.5$	527 ; 4.9	343 ; 3.5	278 ; 5.5

Table VIII: **PC method. Number of iterations and cpu time for one time step ($\Delta t = 0.01$, $Re = 50$, $K = 16$, $N = 8$) as a function of the deformation of the geometry (α) and preconditioner. The cpu time is normalized with respect to the preconditioned PC method ($\alpha = 0$). Tolerance 10^{-9} for pressure and 10^{-11} for the velocity.**

Next, we test the preconditioners on a geometry $[0, 12] \times [-1, 1] \times [-1, 1]$ consisting of $4 \times 2 \times 2$ spectral elements ($N = 8$). The three interfaces in x-direction are deformed by the function $\alpha \sin(\pi y) \sin(\pi z)$. At the inflow, we impose a Poiseuille profile and at the outflow we have $\partial u_1/\partial n = 0$. In Table VII, we compare the number of iterations and cpu time for one time step $\Delta t = 0.01$, $Re = 50$, $s = 1$ of the Uzawa method, for different values of α and for different preconditioners. Table VIII and Table IX represent the same test, using the PC and BP method respectively. The cpu time for Tables VII-IX, measured on a Silicon Graphics R3000 work station, is normalized with

	No prec.	$P_u^{-1} \rightarrow S_{bp}$
$\alpha = 0$	395 ; 10.3	88 ; 2.5
$\alpha = 0.1$	410 ; 10.7	117 ; 3.3
$\alpha = 0.5$	474 ; 12.4	304 ; 8.3

Table IX: **BP method. Number of iterations and cpu time for one time step** ($\Delta t = 0.01$, $Re = 50$, $K = 16$, $N = 8$) **as a function of the deformation of the geometry** (α) **and preconditioner. The cpu time is normalized with respect to the preconditioned PC method** ($\alpha = 0$), **see Table PC. Tolerance** 10^{-9} **for pressure and** 10^{-11} **for the velocity.**

respect to the preconditioned PC method on a non-deformed geometry ($\alpha = 0$). As expected, the PC method is the fastest. The interest of the two-stage preconditioning technique is very well illustrated by Table VII. We also see that the number of iterations increases with α, especially for the preconditioned operators. The difference in cpu time accounts for a high, moderate and low cost to evaluate S_u, S_{bp}, and S_{pc}, respectively. In Section 6.1, we observed that for $Re \rightarrow \infty$ the three operators S_u, S_{bp}, and S_{pc} are equivalent. Under the same assumption, we have also that $P_{pc} = P_u$, discarding the constant $\beta_s/\Delta t$, which does not influence the convergence rate. We do, indeed, find that $P_u^{-1} \rightarrow S_u$, $P_u^{-1} \rightarrow S_{bp}$, and $P_{pc}^{-1} \rightarrow S_{pc}$ require about the same number of iterations to converge.

The reason for solving the pressure in two subspaces is that the condition number of the preconditioned E_N matrix is independent of the number of spectral elements K (see [63]). The tests in the previous section did not exhibit enough elements to verify this. In Table X, we give the number of iterations and cpu time for $16 \leq K \leq 128$ and compare two preconditioners, $P_{pc}^{-1} \rightarrow_E S_{pc}$ and $P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$. First of all, we observe that the number of iterations for both methods grows linearly with N. Second, the number of iterations for $P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$ is always smaller than for $P_{pc}^{-1} \rightarrow_E S_{pc}$. Since the evaluation of the pressure operator E_N is twice as expensive as the evaluation of E, the latter preconditioner requires less cpu time, despite the larger number of iterations. From this we conclude that, for this example, the solution in two subspaces does not pay of. Third, the number of iterations grows for both preconditioning methods with K. This increase is however much smaller for $P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$. It is the author's opinion that the number of iterations is not independent of K because of the changing aspect ratio $l_x : l_y : l_z$. This influences the rate of convergence, explaining also the *drop* in the number of iteration when going from $4 \times 4 \times 2$ elements to $4 \times 4 \times 4$. Finally, by extrapolating the results to more than 128 elements, we expect that $P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$ will eventually win the race.

dimensions	$P_{pc}^{-1} \rightarrow_E S_{pc}$	$P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$
$K = 4 \times 2 \times 2$, $N = 7$	556 (393")	457 (515")
$K = 4 \times 2 \times 2$, $N = 10$	807 (1052)	662 (1357")
$K = 4 \times 2 \times 2$, $N = 13$	1006 (2360")	761 (2980")
$K = 4 \times 4 \times 2$, $N = 7$	760 (931")	668 (1363")
$K = 4 \times 4 \times 2$, $N = 10$	1079 (2730")	912 (3810")
$K = 4 \times 4 \times 2$, $N = 13$	1412 (6870")	1149 (9340")
$K = 4 \times 4 \times 4$, $N = 7$	776 (2140")	554 (2890")
$K = 4 \times 4 \times 4$, $N = 10$	1093 (5610")	743 (6990")
$K = 4 \times 4 \times 4$, $N = 13$	1409 (13270")	992 (15430")
$K = 4 \times 4 \times 8$, $N = 7$	1282 (7923")	740 (8300")
$K = 4 \times 4 \times 8$, $N = 10$	1810 (19630")	1060 (21800")
$K = 4 \times 4 \times 8$, $N = 13$	2346 (46100")	- (-")

Table X: **Pressure correction method. Number of iterations and accumulated cpu time on K processors of the Cray T3D for ten time step ($\Delta t = 0.01$, $Re = 0.033^{-1}$, $s = 1$). Tolerances for velocity and pressure are resp. 10^{-10} and 10^{-8}. Subcycling method for the computation of the nonlinear terms with $M = 3$. In order to get better parallel results, a modified version of the PCGM has been used. This method failed to produce results for $P_{pc}^{-1} \rightarrow_{E_N} S_{pc}$ ($K = 128$, $N = 13$). See the discussion in Section 7.3.1**

Chapter 7

Implementation and parallelization

In order to be able to actually perform numerical simulations, the theory that has been developed and tested in the previous chapters has to be implemented on a computer. The importance of careful programming is widely recognized. Everybody who has actually implemented a computational fluid dynamics (CFD) problem knows that a close inspection of only a few cpu-intensive routines (or even do-loops) can lead to impressive speed-ups. Most of the optimization rules are general and valid on every computer. A classical example is that loops over variables with multiple indices have to be avoided. Some optimization heuristics depend on the architecture of the machine, like, for example, cache manipulations. The differences between computers can be that big, that the discretization and/or solution methods are chosen in accordance with the target architecture. (Pseudo-) spectral Chebyshev methods, for instance, are known to be efficient on vector computers. We can also imagine a situation in which a direct method is preferred on one architecture and an iterative method on another. It is important to consider these questions at an early stage. The interpretation of results is also computer dependent. In general, it is not fair to compare the results supplied by different methods without mentioning the computer and without assuming that the code is optimized: Scientists become scientific programmers and the person who finds a way to accelerate a commercial or scientific CFD computer code contributes as much as the one who conceived the algorithms.

The interaction between the discretization/solution method and target machine is especially important in the context of massively parallel computers based on distributed memory. Such machines consist of a number of independent processors. Each processing unit possesses its own cpu and memory and is connected to a number of neighboring chips. A program that runs on a massively parallel processor (MPP) will typically let each processor perform a certain number of computations on its local variables, followed by a synchronization (communication) step. The synchronization phase is mere overhead and should be as short as possible. Depending on the amount of operations that can be done without communication, an application is called coarse, medium or fine grained. It is clear that a high ratio of computation to communication leads to parallel efficiency. The discretization should be chosen such that a high parallel performance can be obtained. Not only granularity, but also an equal distribution of the work load among the processors (also called load-balancing), a minimum of communication,

and scalability are important issues.

A very common way to exploit parallelism for periodic, three-dimensional CFD problems (for example in axisymmetric geometries) is to use a Fourier expansion in the periodic direction. In this way, the problem is reduced to a number of two-dimensional subproblems that can be solved in parallel. Another strategy is based on the decomposition of the domain in a number of subdomains, each of which is mapped on one processor. The solution method can either be direct (often in combination with a Schur complement technique), see e.g. Zone et al. [77], or iterative. The spectral element solver as discussed in the previous chapters embodies both domain decomposition and efficient iterative solvers. Implementation on a parallel computer is, therefore, rather straightforward (see e.g. Fischer and Patera [26]). The granularity can be tuned by N. Each matrix multiplication takes $O(N^4)$ operations followed by a communication step, so good parallel efficiency comes almost naturally with large values of N. However, the question (or challenge) is to obtain good results for reasonable values of N which lie within the same range as those on scalar machines. The interested reader is referred to the review article of Fischer and Patera [27] for more details about parallel CFD.

The outline of this chapter is as follows. We will start by describing globally the spectral element code, discarding parallelism. Then we discuss the hard- and software of modern MPP's, focused on the Cray T3D. Finally the implementation on the T3D is presented and some results are given to illustrate the obtained parallel efficiency.

7.1 Sequential code

We start this section by giving an overview of the possibilities and limitations of the developed spectral element code. We will first discuss the non-parallelized version. This code consists of 9000 Fortran lines and has been built from scratch. The mesh is defined by a parametric representation of the faces of the spectral elements and by the interfaces (faces, edges and corners): A file has to be supplied defining the two elements that share one face, the four elements that share one edge, and the eight elements that share one corner. Everything which is out of the ordinary, like edges that join three or five, instead of four elements or the situation in which two elements with a different orientation share an interface is possible, but has to be programmed by hand; A problem is defined by the boundary conditions (Dirichlet, Neumann or symmetry), initial conditions, right-hand-side vector and geometry (mesh). The polynomial degree N has to be the same on each element and in each spatial direction. Routines from the following libraries are called: Linpack, Eispack, Blas, and "Mitlib", a library developed at the Massachusetts Institute of Technology for the computation of the Gauss-Lobatto-Legendre and Gauss-Legendre grid points and weights, spectral derivatives, interpolants, etc. Furthermore, the iterative methods are programmed with the aid of SLAP (sparse linear algebra package), developed at the Lawrence Livermore National Laboratory. There is a difference between the parallel code and the one that runs on sequential machines. The latter is capable of

- Deformed spectral element discretization. See Chapter 2.

- Time integration of the linear part of the Navier-Stokes equations by backward differentiation schemes up to order three. See Chapter 3.
- Time discretization of the nonlinear term by extrapolation. See Chapter 3.
- Implicit-explicit splitting by the operator-integration-factor method, where the nonlinear term is computed by the fourth-order classical Runge-Kutta method. The nonlinear term can be evaluated in either its linearized or non-linearized form. See Chapter 3.
- The fast Helmholtz solver based on the Schur complement method has only been implemented for a limited number of geometries. However, the incomplete Schur method is general. According to Chapter 4.
- Decoupling of the velocities from the pressure by the Uzawa method, pressure-correction method and the third-order projection method acting on either the pressure or the pressure correction. See Chapter 5.
- Preconditioning of the aforementioned pressure operators by all the FDM-based preconditioners, with or without solves in coarse and fine pressure subspaces. The boundary conditions that are used to construct the block-diagonal preconditioners (at the interfaces) are either homogeneous Dirichlet or Neumann. According to Chapter 6.

The parallel solver contains all these features, except the fast Helmholtz solvers.

As has been stated in the previous chapters, the operation count for the basic operations, like the evaluation of all the operators and FDM-based preconditioners, scales like $O(KN^4)$. This is almost exclusively due to loops with the following form:

```
      DO 20 IEL = 1,NEL
      DO 20 I = 1,N
      DO 20 J = 1,N
      DO 20 K = 1,N
        SUM = 0.d0
        DO 10 M = 1,N
          SUM = SUM + AZ(K,M) * B(I,J,M,IEL)
10      CONTINUE
        C(I,J,K,IEL) = SUM
20    CONTINUE
```

The main line of loop 10 can also be

```
SUM = SUM + AY(J,M) * B(I,M,K,IEL)
```

or

```
SUM = SUM + AX(I,M) * B(M,J,K,IEL)
```

Here, N denotes the polynomial degree and NEL the number of spectral elements. In most cases, the arrays AX, AY, and AZ depend on IEL as well. Do-loop 10 can be seen as an $N \times N$ matrix-matrix multiplication and can be performed by BLAS. The advantage of such a subroutine call is that BLAS is optimized on most machines. Sometimes, BLAS is even written in assembler and care has been taken to reduce the number of cache-misses. The relatively small size of the matrices has a positive and a negative consequence: Each matrix-matrix multiplication can be done in-cache, but vectorization will not improve the speed of the multiplication. Some $O(KN^3)$ operations, like a daxpy and a scalar product can also be performed by calls to the BLAS library. The spectral element code is based on iterative solvers. Hence, the memory requirements are relatively low.

7.2 Parallel architectures and programming models

There are two reasons that explain the popularity of MPP's. The first one is technology. There are limits on the hardware level, like transistor switching speed and signal processing speed, that make it difficult to keep on improving the individual chips. There is also a need for ever larger memory, which is easier to obtain by many "small" processors than by a single large one. Moreover, the computational capacity of these MPP's is expected to increase faster than their serial counterparts. This growth is not only due to the ability of computer manufacturers to put more and more processors together, but also to the increasing power of the individual chips. The second reason is an economical one. It is well known that the highest Mflops rate per dollar is obtained by massively parallel computers. For these reasons, the MPP's are strongly represented at the world most important computer sites.

We have mainly used the Cray T3D for parallel processing. This machine is the first MPP developed by Cray and will be succeeded by the T3E and T3X (scheduled for 1996 and 1997, resp.), which will eventually be able to run at a sustained speed of one Teraflops (according to Cray). It is not appropriate in this context to give a full overview of the hardware of the Cray T3D. We will restrict ourselves to saying that the current EPFL configuration is equipped with 256 Dec Alpha RISC processors (DEC chip 21064) containing 64 Mbytes of memory with a peak performance of 150 Mflops. However, most single-node applications do not even come close to this figure for a number of reasons: For example, all memory operations stall upon a cache miss. This underlines the importance of the interpretation of most basic operations as matrix-matrix products, which can, hopefully, entirely be computed in-cache. Another, less serious problem is the extremely low speed the square roots are computed with.

All the 256 PE's together constitute a theoretical peak performance of 38.4 Gflops. Two processing elements (PE's) reside on one node, which is placed in a three-dimensional torus topology. In this way, each node is connected to six neighbors. In contrast with the previous generation of parallel supercomputers, a message sent from node A to B is considered to be independent of whether A and B are physically neighbors or not. That is, according to computer vendors' advertisement, the number of "hops"

```
      REAL U(N,2), V(N,2), SUM(2), MYSUM
CDIR$ SHARED U(:,:BLOCK), V(:,:BLOCK), SUM(:BLOCK)
CDIR$ DOSHARED (IEL) ON SUM(IEL)
      DO 10 IEL = 1,2
        SUM(IEL) = SDOT(N, U(1,IEL), 1, V(1,IEL), 1)
10    CONTINUE
      MYSUM = 0
      DO 20 IEL = 1,2
        MYSUM = MYSUM + SUM(IEL)
20    CONTINUE
```

Figure 7.1: **Pseudo-code that illustrates the computation on two PE's of the scalar product of U and V by the work-sharing model. The ":BLOCK" declaration enforces that the first columns of arrays U and V is stored on one PE and the second column on the other one. Note that the address space is global. The arrays are supposed to be initialized. Do-loop 10 is a parallel loop: Each processor calls the BLAS subroutine SDOT to compute the local contribution of the scalar product. The last do-loop represents communication: The local sum residing on the other PE has to be loaded to compute MYSUM, which is identical on both processors, once the last line has been executed.**

does not play a role.

The software that manages the inter-processor communication is crucial for the appreciation of modern MPP's. Keywords are ease of programming, speed, and compatibility. In addition, the availability of parallel debugging tools and profilers are highly desirable. Cray uses the concept of programming model to indicate different ways to manage communication. We will briefly introduce the three models supported by Cray.

The first programming model, which is of little interest for our applications, is the data-parallel model. Although a number of definitions for "data parallel" are used by different companies, we refer here to the presence of array syntax, virtualization (the user perceives a logical granularity , i.e. the array dimensions appropriate to the code) and a set of intrinsic functions that are used for communication, global operations, etc. (these intrinsics are typically those of Fortran 90). The parallelism is said to be fully implicit; the communication and data distribution is taken care of by the compiler and is invisible for the programmer.

The so-called work-sharing model is a natural style for a multiprocessor since it contains a global addressing space. The data distributions are managed by compiler directives which specify the way in which arrays are stored. These directives are also used to distribute the work, for example parallel do-loops, over the different processors. The resulting computer code is identical to a serial code, except for the directives. Communication is implicit, but the synchronization and data decomposition is specified by the user. In Figure 7.1, the computation of the scalar product by the work-sharing programming model on two PE's is illustrated.

```
      REAL U(N), V(N), OTHERSUM, MYSUM
      INTEGER MYPROC
CCC   *** PVM INITIALIZATION ***
CCC   *** MYPROC EITHER 0 OR 1 ***
      MYSUM = SDOT(N, U, 1, V, 1)
      IF (MYPROC=0) THEN
        CALL PVMFRECV(OTHERSUM,1)
        MYSUM = MYSUM + OTHERSUM
        CALL PVMFSEND(MYSUM,1)
      ELSE IF (MYPROC=1) THEN
        CALL PVMFSEND(MYSUM,0)
        CALL PVMFRECV(MYSUM,0)
      ENDIF
```

Figure 7.2: **Pseudo-code that illustrates the computation on two PE's of the scalar product of U and V by PVM. The address space is local. The above program is started on both PE's and the only way to distinguish them is by the variable MYPROC, which obtains the value 0 on one PE, and the value 1 on the other in the PVM initialization phase. The arrays are supposed to be initialized. Each processor calls the BLAS subroutine SDOT to compute the local contribution of the scalar product. The last lines represents simplified calls to the PVM communication library. PE0 receives the local value of PE1, adds it to its own value and sends the result back.**

The third programming model is PVM (Parallel Virtual Machine). PVM is a well-established message-passing library that allows the programmer to send messages from one processor to another (originally, PVM was also conceived for heterogeneous networks). There is no longer a global address space, but each processor has its own separate one. Communication, data distribution and synchronization are explicit. The advantages of PVM are robustness and portability, since it is supported by a large number of nowadays parallel computer vendors (Convex, Intel, etc.). Unfortunately, however, there exist almost as many different PVM versions as computer vendors. In particular, the Cray-PVM instructions for sends and receives differ slightly from standard PVM. Moreover, "fast" sends and receives have not been implemented. As another disadvantage, we mention that PVM is a C-library and that, consequently, communication is managed by potentially slow routines. Figure 7.2 represents the computation of the scalar product on two PE's using the standard PVM programming model. There exist so-called shared memory operations ("shmem-get" and "shmem-put") on the T3D, which can be used in the frame of a PVM program. These instructions allow a processor to get (or put) information from (or on) another PE by means of direct addressing (i.e. by specifying the name of the array and the (logical) processor it resides on). In this way, sends and receives do no longer have to match and very fast communication is obtained. In the next section, we will report the speed of these "shared" communication routines.

The term "programming model" intrinsically implies a programmer's point of view. However, ease of programming is often difficult to combine with fast communication. If no information is specified on the location of certain arrays (cf. implicit data dis-

tribution), there is a potentially high ratio between communication and computation. The work-sharing model and PVM are very much alike from the conceptual point of view. Although the former does not require explicit communication, the programmer should have a very clear view of where to place data and when to communicate it, as for PVM. The concept of shared variables is present, but its use is immediately penalized by expensive communication, and should be kept as low as possible. Our choice for the standard PVM model, which had to be made at an early stage, was motivated by two issues. First, PVM was available on the T3D before work-sharing. The first version of the compiler for the latter model was expected to suffer from some start-up problems and to be rather slow. Second, PVM is supported on many other MPP's leading to a portable code. At the moment of writing this thesis, these two arguments have become less convincing: It will be shown that the work-sharing model is at least competitive with PVM. Furthermore, once the communication routines based on standard PVM have been written, they can easily be replaced by their shared-memory counterparts, or even by NX routines, in the case of the Intel Paragon. The gain in efficiency that can be obtained by calling the architecture-specific routines is very impressive, as will be illustrated in the next section, and portability is easily recovered.

7.3 Parallel spectral element solver

In this section, we will study the parallelization of the iterative spectral element solver using the PVM programming model. We will use the convention that there is a 1-1 mapping between the spectral elements and the processors. For reasons of convenience, it is also assumed that $K(= P$, the number of processors) is a power of two.

7.3.1 Parallel conjugate gradient method

From several runs of the serial code, we deduce that more than 90% of the cpu time is consumed by the PCGM's to solve the Helmholtz and pressure equations. The subcycling method comes in second place. Let us consider the PCGM to solve the system Qu =rhs, where Q can be thought of as the Helmholtz operator or as one of the pressure operators. The algorithm is depicted in Figure 7.3. Communication is explicitly required for the computation of the scalar products and in most practical cases also for the evaluation of Qp_i, which requires the update of the interfaces (direct stiffness). The preconditioner P is assumed to be block diagonal and can be evaluated without communication, like the vector updates. From Figure 7.3 it is clear that two scalar products are required in each PCGM iteration. They can not be computed at the same time, so we will speak of two synchronization points. Although most cpu time is spent in the evaluation of Qp_i and in the solution of $Pz_i = r_i$, it is, according to Amdahl's law, not correct to conclude that little attention has to be paid to the remaining parts. Therefore, an algorithm that reduces the number of synchronization points has a practical interest. Meurant [53] has proposed a modified version of the PCGM which contains only one synchronization point: Three, instead of two scalar products have to be computed, but this can be done at the same time. The modified version is based on the observation that (r_{i+1}, z_{i+1}) can be computed before z_{i+1}. The

solve $Pz_i = r_i$
$\boldsymbol{\beta_i = (r_i, z_i)(r_{i-1}, z_{i-1})^{-1}}$
$p_i = z_i + \beta_i p_{i-1}$
$\boldsymbol{q_i = Qp_i}$
$\boldsymbol{\alpha_i = (r_i, z_i)(q_i, p_i)^{-1}}$
$u_{i+1} = u_i + \alpha_i p_i$
if u_{i+1} accurate enough quit
$r_{i+1} = r_i - \alpha_i q_i$

Figure 7.3: **Standard version of the preconditioned conjugate gradient method to solve Qu =rhs. The algorithm is initialized by r_0 =rhs$-Qu_0$, with u_0 and p_{-1} arbitrary. The preconditioner P is assumed to be block diagonal. Steps that require communication are represented in bold.**

$\boldsymbol{q_i = Qp_i}$
solve $Pv_i = q_i$
$\boldsymbol{\alpha_i = (r_i, z_i)(q_i, p_i)^{-1}}$
$\boldsymbol{\beta_{i+1} = (r_i, z_i)(v_i, q_i)(q_i, p_i)^{-2} - 1}$
$u_{i+1} = u_i + \alpha_i p_i$
if u_{i+1} accurate enough quit
$r_{i+1} = r_i - \alpha_i q_i$
$z_{i+1} = z_i - \alpha_i v_i$
$p_{i+1} = z_{i+1} + \beta_{i+1} p_i$

Figure 7.4: **Modified version of the preconditioned conjugate gradient method to solve Qu =rhs. The algorithm is initialized by r_0 =rhs$-Qu_0$, $Pz_0 = r_0$, $p_0 = z_0$, with u_0 arbitrary. The preconditioner P is assumed to be block diagonal. Steps that require communication are represented in bold.**

computation of β_i in the second line of the original PCGM (Figure 7.3) is then replaced by β_{i+1}, which can take place at the same time as α_i: From $r_{i+1} = r_i - \alpha_i q_i$ it follows that

$$(r_{i+1}, z_{i+1}) = (r_i, z_{i+1}) - (\alpha_i q_i, z_{i+1}). \tag{7.1}$$

With the aid of the orthogonality relation $(r_i, z_j) = 0$ for $i \neq j$ and $z_{i+1} = z_i - \alpha_i P^{-1} q_i$, we derive that

$$(r_{i+1}, z_{i+1}) = \alpha_i^2 (P^{-1} q_i, q_i) - (\alpha_i q_i, z_i). \tag{7.2}$$

From Equation (7.2), $-\alpha_i q_i = r_{i+1} - r_i$ and the orthogonality relation it follows that

$$(r_{i+1}, z_{i+1}) = \alpha_i^2 (P^{-1} q_i, q_i) - (r_i, z_i). \tag{7.3}$$

It is clear that three scalar products have to be computed. The algorithm is then rewritten, as displayed in Figure 7.4. Meurant reports that his version of the PCGM converges in about the same number of iterations as the standard algorithm and that unstable behaviour was not encountered. We can confirm the first claim, but not the second one. In some situations in which the standard PCGM performs correctly, the modified version fails to converge. We have already met one example in Chapter 6, Table X. Another situation in which this modified method is often found to be unstable is in the context of the solution of the Uzawa pressure operator. This is not only the case when a modified PCGM is used for the inner *and* outer iterations, but also when the outer *or* inner iterative solver is replaced by the modified PCGM. However, in most practical situations the modified PCGM is reliable and less expensive in terms of cpu time. Therefore, we have used the Meurant algorithm for all the tests that will be presented in this chapter and in the next. In the following two sections we will focus on the two operations that require communication; the computation of the scalar product and direct stiffness, respectively.

7.3.2 Parallel computation of the scalar product

The computation of the scalar product by PVM has already been addressed in Figure 7.2. Usually, there are more than two PE's involved and several techniques can be used which differ in the way they communicate the local sums over the processors. Their efficiency varies with the number of PE's. We propose four algorithms. The most straightforward method to construct the global sum out of the local contributions is that all the processors send their result to one PE which performs the addition and broadcasts the result back. (A broadcast is a send to all the other processors.) This process is schematically represented for eight PE's in Figure 7.5. All the PE's will be blocking until one of them has received all the messages, which come in sequentially. Especially for a large number of spectral elements, the waiting time might become unacceptably long. This saturation problem can be solved by communicating the local sums along a binary-tree configuration, as has been displayed in Figure 7.6. In this way, $^2 \log P$ communication steps are necessary, but no processor is waiting for more than one message at the time. It can be interesting to replace the broadcast by an inverse binary tree. This approach is rather straightforward and is not represented in a figure. The efficiency of this strategy allows us to investigate the speed of a PVM

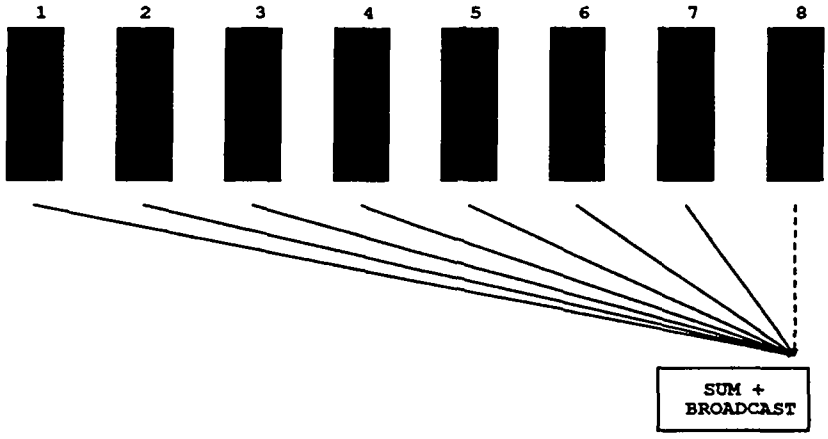

Figure 7.5: Sendall-broadcast algorithm to compute scalar product. The eight processors ($K = P = 8$) are represented by black bars. Messages are indicated by straight arrows. The sendall-broadcast technique consists of communicating all the local sums to one processor that performs the addition. The final value is broadcast back.

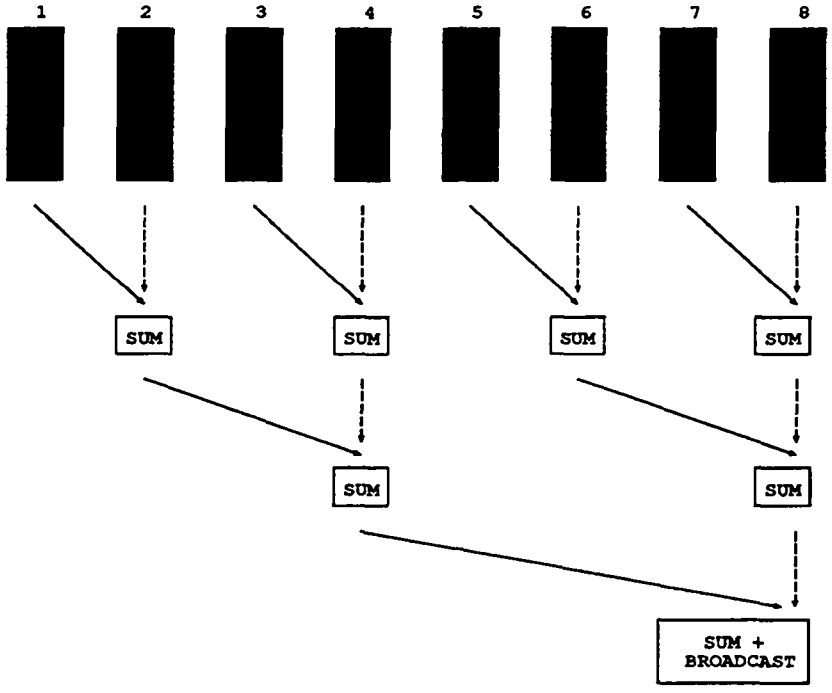

Figure 7.6: Tree-broadcast algorithm to compute scalar product. The eight processors ($K = P = 8$) are represented by black bars. Messages are indicated by straight arrows. The tree-broadcast technique consists of communicating all the local sums to one processor in $^2 \log P$ steps. The final value is broadcast back.

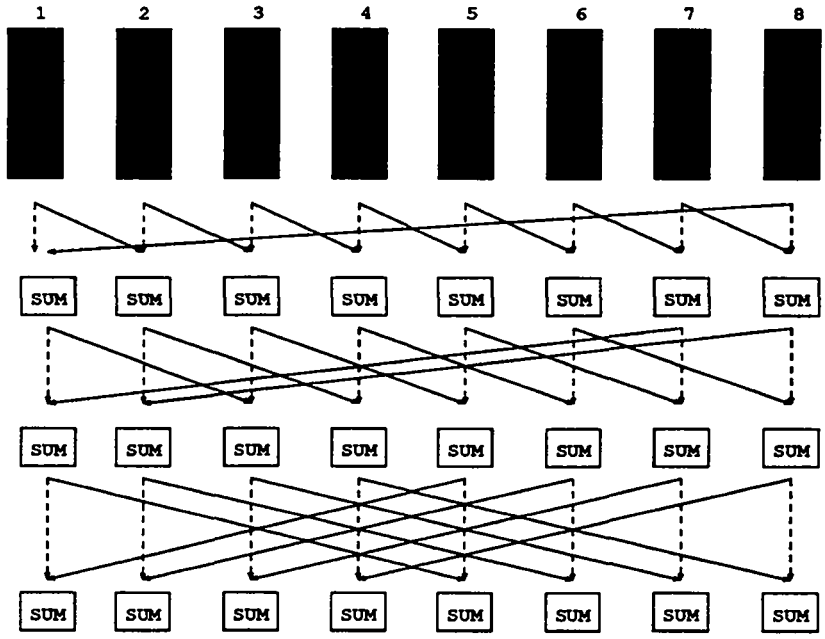

Figure 7.7: **Cascade algorithm to compute scalar product. The eight processors ($K = P = 8$) are represented by black bars. Messages are indicated by straight arrows. The cascade technique is very similar to the tree-broadcast method, except that at the end of $^2 \log P$ communication steps, all the processors possess the correct value.**

broadcast instruction. Finally, we present the cascade algorithm (see e.g. [39]) in Figure 7.7. This is a clever modification of the binary tree algorithm (Figure 7.6), such that the global result is known on every processor after $^2 \log P$ communication steps. By comparing the algorithms represented in Figure 7.6 and Figure 7.7, we see that the latter leads to an explosive increase of the number of messages. However, none of the processors is waiting for more than one message at the time and the final broadcast is not necessary. We will refer to the four methods as respectively sendall-broadcast, tree-broadcast, tree-tree, and cascade.

We have investigated the four algorithms on the T3D for a different number of processors, using the PVM programming model. The results can be found in Table I. It should be noted that the cpu times are measured with the tool Apprentice, and include waiting time. The sendall-broadcast method performs well on a small number of processors, but is inefficient for $P > 32$, due to the fact that many messages have to be received sequentially (although they may arrive in any order). The cpu time for the three communication algorithms based on a binary tree increases by an almost constant value each time the number of PE's is doubled. It is remarkable that the cascade method is the most efficient one, despite the high number of messages that are sent. Apparently, the T3D does not get easily congested.

Two alternative ways to compute the scalar product have been investigated. They are both based on (virtual) shared-memory operations and are exclusive on the T3D.

$K = P$	Sendall-broadcast	Tree-tree	Tree-broadcast	Cascade
2	0.36	0.36	0.36	0.25
4	0.51	0.64	0.53	0.42
8	0.71	0.92	0.70	0.59
16	1.13	1.19	0.86	0.76
32	2.10	1.46	1.02	0.93
64	4.31	1.75	1.19	1.10
128	11.9	2.08	1.39	1.29
256	37.2	2.39	1.57	1.47

Table I: Time in seconds per processor to compute 1000 scalar products ($N = 9$) on the Cray T3D. Standard PVM programming model.

$K = P$	Sendall-bcst. (PVM)	cascade (PVM)	work-sharing(1)	work-sharing(2)
2	0.076	0.078	0.076	0.11
4	0.083	0.087	0.078	0.12
8	0.097	0.097	0.084	0.12
16	0.13	0.11	0.096	0.13
32	0.18	0.12	0.12	0.13
64	0.30	0.13	0.17	0.14
128	0.53	0.14	0.27	0.15
256	1.00	0.15	0.48	0.16

Table II: Time in seconds per processor to compute 1000 scalar products ($N = 9$) on the Cray T3D. The sendall-broadcast and cascade algorithms have been implemented with so-called shared-memory routines that belong to the Cray PVM programming model. The results in the third and fourth columns have been obtained by the work-sharing model. The results in the third column have been obtained by summing the local sums using a do-loop (as displayed in Figure 7.1). The results in the last column are based on the summation of the local sums by the "native" sum instruction which is part of the work-sharing model.

The first option is the "shmem-get" instruction which allows the direct access of the cache of other processors. Addressing is based on two parameters, the name of the variable, and the processor it resides on. For this reason, the shmem-get instruction is said to be based on shared memory. The data transfer is very fast. As an example, we changed the "standard" PVM sends and receives by their "shared" counterparts and computed the scalar product by the sendall-broadcast and cascade algorithms. A comparison between Table I and Table II reveals that, indeed, the shmem-get instruction allows for a fast data exchange. The previously observed problem of saturation for the sendall-broadcast algorithm is not solved. Another alternative consists in the use of the work-sharing model. The results for the computation of the scalar product (based on Figure 7.1) can be found in the third column of Table II. The fourth column displays the results when a special sum instruction, which belongs to the work-sharing syntax, is used. In all cases, it is clear that the use of the special (architecture-specific) communication instructions is worthwhile, although compatibility is sacrificed. The issue of programming models is again discussed in Section 7.4.

7.3.3 Direct stiffness on parallel computers

The concept of direct stiffness is the implementation of Equation (2.49) and consists of the summation of the contributions of two or more grid points that represent the same physical node, followed by redistribution: On the physical grid, interface grid points are unique (see Figure 7.8). In practice, however, two, or more elements (processors) that share an interface possess their own, local copy of the same grid point. After certain operations, the respective values may be different, and synchronization is necessary. Here, we will distinguish two strategies to manage the communication which is involved in this synchronization process.

The first method is represented for a two-dimensional geometry on the left of Figure 7.8: First, the edges are summed and then the corners. In three dimensions, these two phases are preceded by summation of the faces. When direct stiffness is implemented in this way, it is clear that a high number of messages is involved. The processor that performs an update of a corner node (3D), for example, receives seven messages, sums the eight corner values and redistributes the total sum. It is important that receives are non-blocking when more than one message can come in. The implication of this is that the receiving node identifies the sender of the incoming message and waits for the next one, instead of waiting for a message from a specific sender. Although Figure 7.8 suggests that there are two distinct communication phases, it should be noted that the (faces,) edges and corners can be updated independently from each other. Furthermore, this implementation is flexible as regards "special" nodes, i.e. interfaces that join an irregular number of faces, edges and/or corners.

The second method has been proposed by Fischer and Patera [26]. Their approach consists in three consecutive element face exchanges (3D), as can be seen on the right in Figure 7.8. In contrast with the above described method, complete faces (and faces only) are exchanged. After three directional exchanges, corresponding to three spatial directions, edges and corners have automatically received their correct values. It is important to notice that each exchange has to be entirely completed before the next one

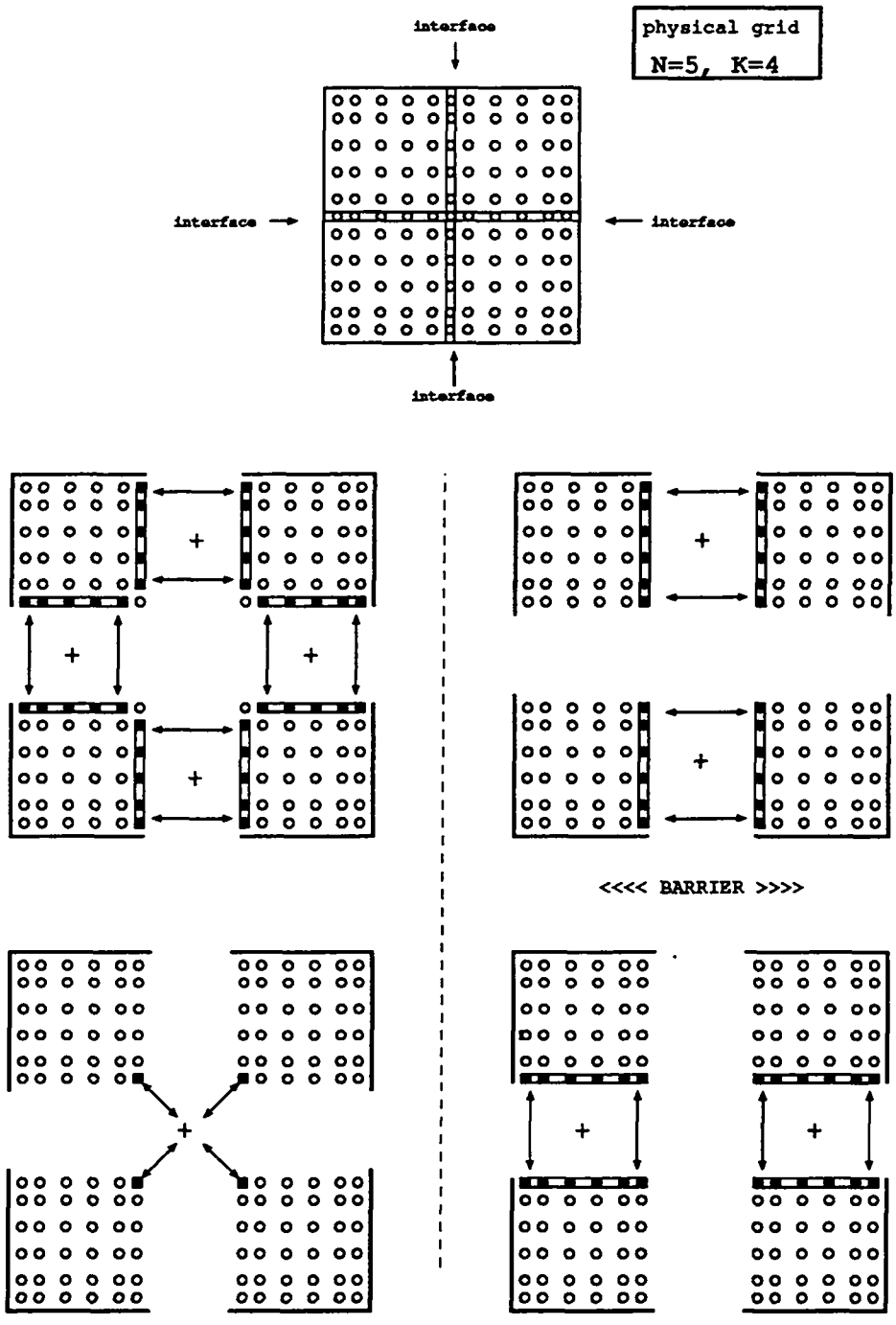

Figure 7.8: Two methods to perform direct stiffness on a four-element grid of degree $N = 5$ (two spatial dimensions). On the physical grid (depicted at the top), each node is unique. In practice, however, each processor has its own copy of the grid points at interfaces, which have to be summed and redistributed. On the left hand, edges and corners are updated by separate mechanisms. Many messages are necessary. On the right hand two directional exchanges are performed. The number of messages is reduced, but synchronization between the two phases is needed.

can start. If this rule is not respected, wrong values might occur at edges and corners. Software barriers are required to enforce correct synchronization. This concept is very simple, and the number of messages is much less than for the previously discussed method. In [26] it is explained how special nodes can be handled: First, their local contributions are accumulated, then the three directional exchanges are performed, and, finally, the correct values are redistributed to the special nodes.

The two methods that have been proposed to perform direct stiffness have not been tested individually, but in the frame of a larger test problem, which will be presented in the next subsection.

7.3.4 Parallel efficiency on the Cray T3D

In this subsection and in the next, we will be concerned with the determination of the parallel efficiency of the spectral element code. Unfortunately, the parallel code has evolved independently of the sequential version, which makes the standard definition of parallel efficiency (the quotient of the cpu time for the fastest serial code and P times the cpu time of the parallel code) impossible. Therefore, we adopt the alternative definition of parallel efficiency $\mathcal{E}_{N,K}$ to solve a K-element problem of degree N on K processors as

$$\mathcal{E}_{N,K} = \left(1 - \frac{\text{cpu time for communication}}{\text{total cpu time}}\right) 100\%. \tag{7.4}$$

Again, the cpu time for communication and the total cpu time include the time that processors are waiting for messages to arrive. Although the processors receive exactly the same amount of work, problems concerning load-balancing are immediately detected by this definition. The test problem that we have selected is the simulation of a Navier-Stokes flow in a domain $\Omega = [-1,1]^3$, $t \in [0,0.1]$, $\Delta t = 0.01$, $Re = 0.033^{-1}$, with an analytical solution given by Equations (6.46) and (6.47). A BDF1 is chosen as the temporal method, combined with a subcycling method based on the fourth-order Runge-Kutta scheme, $M = 3$. The pressure-correction method is applied to decouple the velocities from the pressure. The Helmholtz matrix is preconditioned by its diagonal and the pressure operator as $P_{pc}^{-1} \rightarrow_E S_{pc}$. The relative tolerance for the velocity is 10^{-10} and 10^{-9} for the pressure. The modified version of the PCGM has been used. In this subsection, we will only consider timings that have been obtained on the T3D. Start-up time has been discarded.

It is our first goal to obtain some reference timings for different values of N and K. To this end, we have used the fastest method to compute the scalar product; the cascade algorithm. Direct stiffness is performed by separate communication of faces, edges and corners. Standard PVM instructions have been used. The results are depicted in Figure 7.9. We see that the efficiency increases with N. This is not surprising since the number of operations on each spectral element (processor) scales like N^4. For $N = 10$ and $N = 13$, the percentage of cpu time for communication grows slightly with K. This is mainly due to an increasing ratio between the number of interfaces (which equals

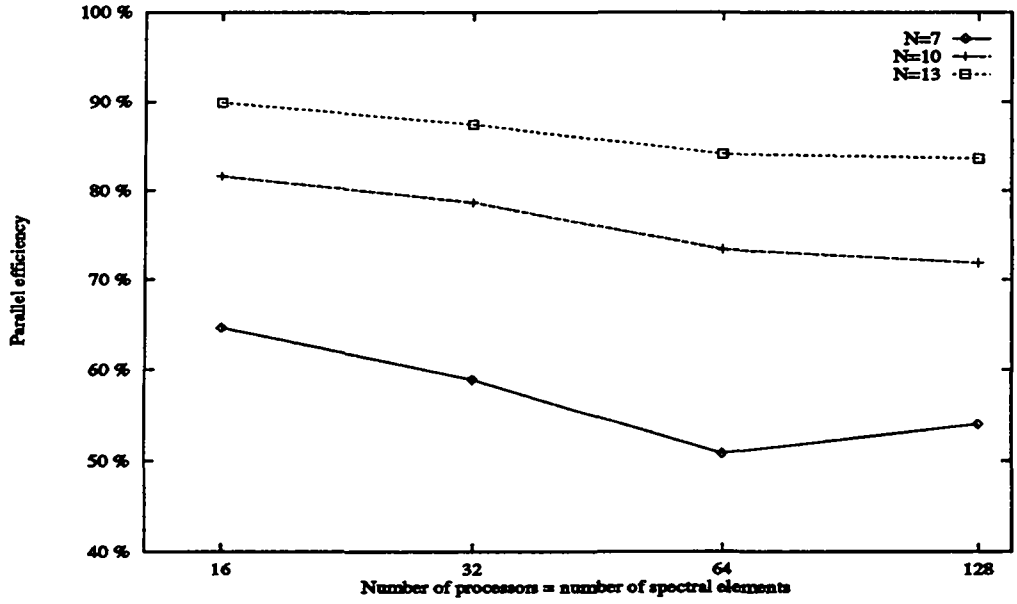

Figure 7.9: **T3D. Parallel efficiency as a function of N and K. Standard PVM instructions. Direct stiffness is performed separately on faces, edges and corners; Cascade algorithm for scalar products.**

the number of messages during the direct stiffness phase) and the number of spectral elements, and only partially to a degrading parallel performance. We investigated in more detail the results for $N = 10$, $K = 128$: Only 5.6 percent of the total cpu time has been used for communicating the scalar products around the processors; 22.5 percent was taken by the direct stiffness phase, which can be subdivided in 7.5 percent for the corners, 9.1 percent for the edges and 5.9 percent for the faces.

We will now show how the efficiency $\mathcal{E}_{N,K}$ can be improved. The first step is to use the tri-directional face exchange to compute the direct stiffness. In the absence of special nodes, we obtained much better results, as is shown in Figure 7.10. For a fixed N, the only decrease in efficiency is due to the computation of the scalar product: the computation of the direct stiffness is such that the number of messages per face during one directional exchange is exactly one, followed by synchronization. This implies that the direct stiffness procedure is scalable with respect to the number of processors.

Finally, we replace the standard PVM instructions by their "shared-memory" PVM-counterparts. As can be seen from Figure 7.11, parallel scalability has been achieved, and communication time has become almost negligible. We conclude that the spectral element solver runs very efficiently on the Cray T3D.

7.3.5 Parallel efficiency on the Paragon

The same program has also been tested on the Intel Paragon, at the ETH Zürich, Switzerland. We took the same test case that led to the results of Figure 7.9 and compared the performances of the two MPP's. First, we remarked that PVM on the Paragon produces disastrous results. At the time of the tests (fall of '94), PVM was implemented as an intermediate layer between the Fortran program and the Intel com-

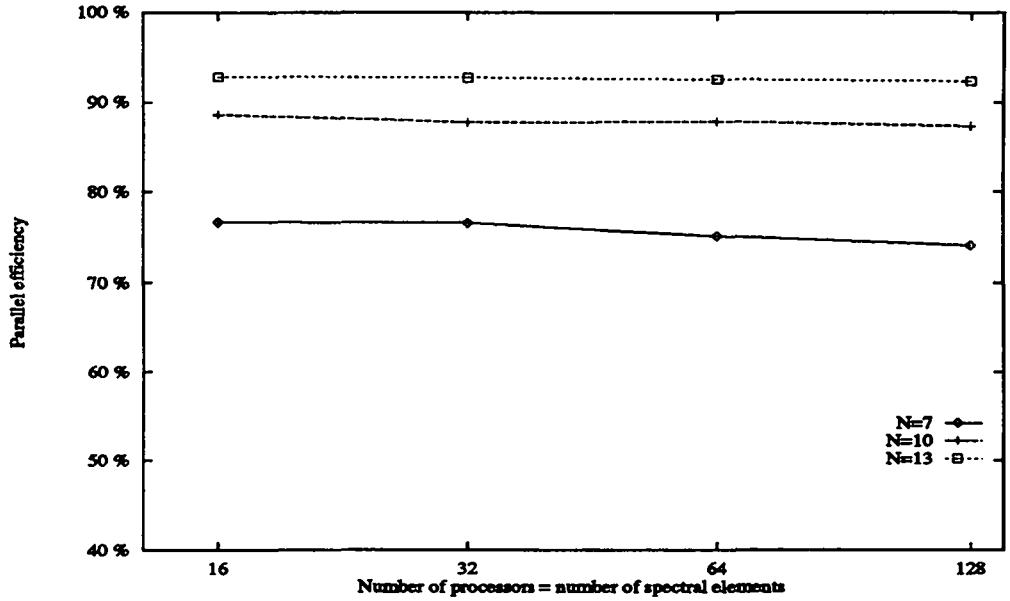

Figure 7.10: **T3D. Parallel efficiency as a function of N and K. Standard PVM instructions. Direct stiffness is performed by a tri-directional face exchange; cascade algorithm for scalar products.**

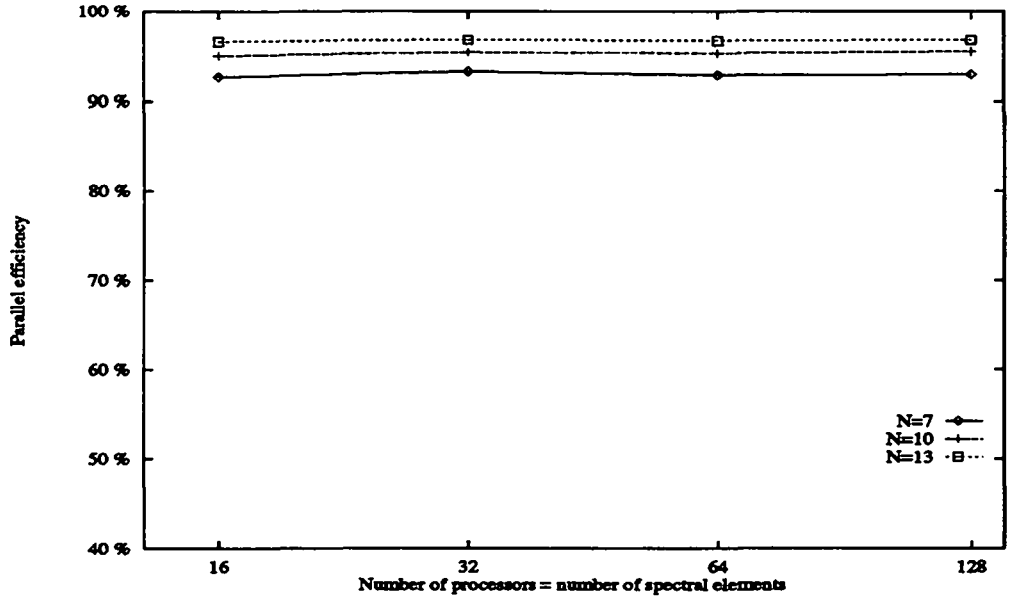

Figure 7.11: **T3D. Parallel efficiency as a function of N and K. Shared-memory PVM instructions. Direct stiffness is performed by a tri-directional face exchange; cascade algorithm for scalar products.**

munication routines. The function calls to and fro the different layers slow down the performance of the program and lead to unacceptable cpu times. Therefore, we have resorted to NX, the Intel library of communication routines. The cpu time and efficiency are represented in Table III; the T3D results are the same as in Figure 7.9. We conclude

	Cray T3D		Intel Paragon	
	cpu time	$\mathcal{E}_{N,K}$	cpu time	$\mathcal{E}_{N,K}$
N=7, K=16	25	64.6%	53	78.7%
N=10, K=16	66	81.6%	169	90.2%
N=13, K=16	148	89.9%	390	94.0%
N=7, K=32	29	58.9%	69	74.6%
N=10, K=32	85	78.7%	234	88.0%
N=13, K=32	215	87.4%	590	92.4%
N=7, K=64	33	50.8%	74	70.2%
N=10, K=64	88	73.4%	233	85.3%
N=13, K=64	207	84.1%	563	90.6%

Table III: **Cpu time in seconds (per processor), for a problem that ran on $K = P$ processors and parallel efficiency. Standard PVM instructions for the Cray T3D, NX for the Intel Paragon. Direct stiffness is performed separately on faces, edges and corners; Cascade algorithm for scalar products.**

that the single-processor performance for the T3D is considerably higher than for the Paragon. The communication seems to be faster on the Paragon, but is in fact slightly slower when we take the relatively low single-node speed into account. From further tests, we deduce that the tri-directional face exchange method for the computation of direct stiffness does not improve the parallel performance on the Paragon.

7.4 Discussion

We close this chapter by three remarks.

Parallel processing is a rapidly evolving domain, implying that decisions and choices can become obsolete within the time span of a few months. This is more or less the case for our decision to start the spectral element simulations (see next chapter) with a code based on standard PVM, and not on shared-memory PVM-instructions. It was difficult to foresee that architecture-specific communication would be much more efficient, as has been shown in the previous sections. Furthermore, the loss of compatibility due to the use of "shared" memory operations turned out to be not crucial: We learned by experience that the FORTRAN routines which manage the communication are easily rewritten. However, we should realize that it is not dramatic to have a code running at a parallel efficiency of, let's say, 80%, whereas 95% could also be obtained. Anyway, the potential to develop a code which is almost optimal as regards parallelism is present, and should certainly be exploited in the future. We remark also that the results of the previous section could be different when "special" nodes are present.

As can be seen from Definition (7.4), a high parallel efficiency can either be obtained by little, fast communication, or by slow single-processor performance. Therefore, we have to assure ourselves that the good parallel results are not due to the latter factor. The Apprentice performance analyzer allows us to measure exactly the speed at which the spectral element solver runs. We found that for $K = P = 128$, a performance of about 1.3 Gflops ($N = 7$), 2.7 Gflops ($N = 10$), 3.9 Gflops ($N = 13$) up to 6 Gflops for the unrealistic value of $N = 20$ can be achieved. On the first hand, these results seem a bit disappointing compared to the theoretical peak performance of 19.2 Gflops. On the other hand, it is a well-known problem to obtain a reasonable single-node performance on the current generation of MPP's. This is mainly due to the fact that the processor stalls on a cache miss. We are confident that the use of BLAS for all cpu-intensive operations pushes the code to a performance which is quite optimal.

When a pressure preconditioner based on coarse and fine subspace solutions (see Chapter 6) is used, a strategy has to be developed to perform the coarse-grid solves, i.e. $E_0 x = \text{rhs.}$, cf. Equation (6.54). Fischer and Rønquist [28] propose to construct the inverse, which is a relatively small $(K \times K)$ matrix, in a preprocessing stage. The computation of $E_0^{-1}\text{rhs.}$ proceeds then as follows: First the right-hand-side vector, which is stored in a distributed way over $K = P$ processors, is gathered on each processor by ${}^2\log P$ communication steps. Then, on each processor, the Kth line of E_0^{-1} is multiplied with the right-hand-side vector. The gathering procedure represents an additional communication overhead which leads to a degrading parallel efficiency. However, the difference with the results as displayed in Figure 7.9, Section 7.3.4 is always less than 3%.

Chapter 8

Spectral element simulations

8.1 Study of the overall time accuracy

In the previous chapters the developed algorithms have been tested separately. This is a sound strategy, because the interpretation of the results is not hindered by side effects. It is, however, also appropriate to test all the components together, especially when the overall order of accuracy of the time-integration scheme is concerned. Many factors influence this global order: the time schemes for the linear and nonlinear terms, the splitting method and the decoupling method. In this section we want to find out how these factors interact, and if we can point out which component yields the dominant error.

It is difficult to find a good three-dimensional benchmark problem with an analytical solution. Therefore, we resort to the test problem with a solution as described in Equations (6.46) and (6.47). Here, both the velocity and the pressure have been multiplied by e^{-t}. Furthermore, we took $\Omega = (-1,1)^3$, $N = 9$, $K = 16$, $T = 0.75$, and a Reynolds number of $Re = 100$. The BDF is of order three. The error in the first component of the velocity is depicted in Figure 8.1 for the following time-schemes: pressure correction, the third-order projection method of Blair-Perot and Uzawa. Each of these schemes has been combined with an extrapolation scheme (BDF3/EX3) or a subcycling method (BDF3/RK4, $M = 3$).

As expected, the pressure-correction method is second-order, and the projection and Uzawa methods are third-order accurate in time. The results for the former two methods are independent of the scheme for the nonlinear terms. The implication of this is that the decoupling error is dominant to the splitting error. In absence of a decoupling error, the Uzawa method is more accurate when an extrapolation method is used. However, for $\Delta t < 0.01$, the spatial error becomes dominant. We can conclude that, for this problem, the decoupling error for the third-order projection scheme is one to two orders of magnitude, and dominates the splitting error.

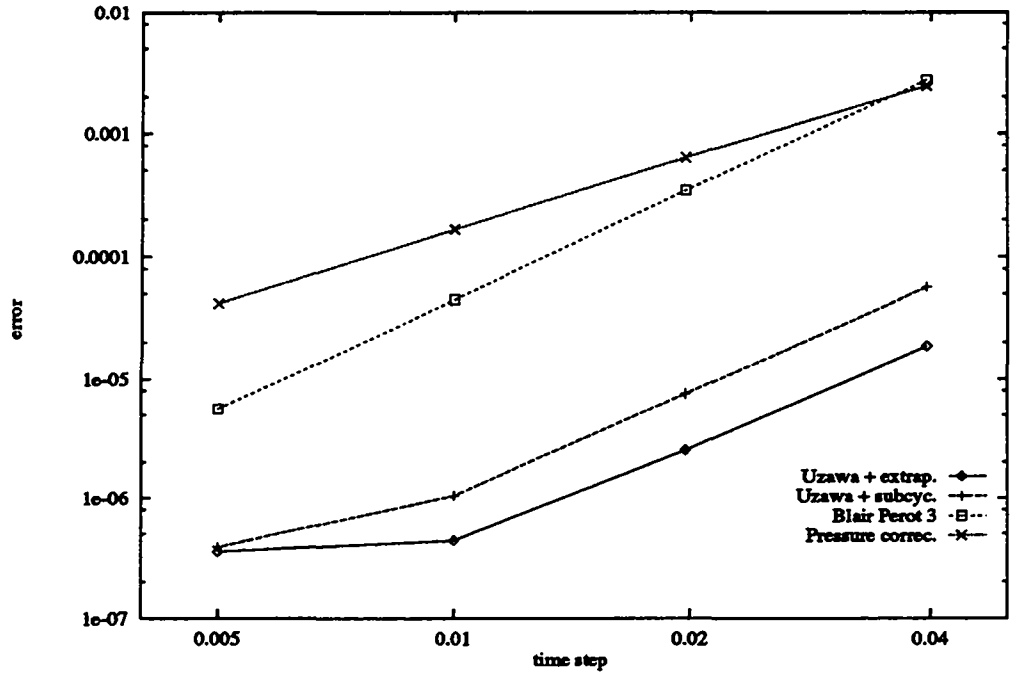

Figure 8.1: **Overall time accuracy of the Uzawa, the pressure correction, and the third-order projection method. The latter two methods show the same results with the subcycling and the extrapolation scheme.**

8.2 Three-dimensional flow over a backward-facing-step

In this section, we investigate the flow over a backward-facing step. An extensive comparison between experimental data and the results from our spectral element simulations will be made. First, the flow characteristics are described, as they have been observed by experiments and two-dimensional simulations. Then the three-dimensional simulations will be exposed for different Reynolds numbers.

8.2.1 Description of the backward-facing-step flow

The two-dimensional flow over a backward-facing step (BFS) is very well documented. One of the reasons is that the BFS flow at $Re = 800$ has been proposed as a benchmark problem (see e.g. Sani and Gresho [68]). The results of Gartling [30] and Kim and Moin [43] are often used as a reference solution. There is also an ongoing discussion on the steadiness of the flow at $Re = 800$, which started with the paper of Kaiktsis et al. [40]. The authors used a spectral element code and observed that the flow is unsteady with eddies oscillating around the alleged steady solution. Gresho et al. [35] disagree with this conclusion which is contradicted by four independent analyses that all show steady state. Especially the results obtained by the NEKTON spectral element solver are interesting because they suggest that marginally resolved flow in the streamwise direction can alter the temporal behaviour of the numerical solution. The authors claim also that high-order methods are more sensitive to this phenomenon than low-order

methods. The main argument for this theory is that high-order polynomials are forced to fit through spatial variations that are too rapid to be represented accurately. This results in nonphysical wiggles which are convected in the underresolved, streamwise direction. Kaiktsis et al. react in a second paper [41] in which they conclude that, indeed, the observed unsteadiness is unphysical and due to a lack of spatial accuracy in the streamwise direction. Furthermore, they try to explain this phenomenon in terms of convective instable behaviour, implying that beyond a critical Reynolds number ($Re > 700$), upstream generated disturbances are amplified downstream, before the flow reaches, eventually, its steady state. Convective instability in conjunction with insufficient spatial accuracy is held responsible for unphysical unsteadiness.

The reason to bring up the discussion about the two-dimensional BFS flow is that many lessons have been learned that apply also to the three-dimensional case. First, care has to be taken that the flow is sufficiently resolved. In Gresho et al. [35], it is suggested that the divergence computed in the "strong" way acts as a barometer of the quality of the solution. A safer instrument is, however, grid refinement. The fact that the flow is almost convectively unstable warns us that time integration over a long interval will be necessary before a steady solution is reached. Finally, we recall that the flow is weakly singular at the step corner, that is, the pressure is negatively infinite. Therefore, mesh refinement is needed close to the corner in order to keep the effects of the singularity local.

Another very important contribution has been made by Armaly et al. [1], who simulated experimentally the three-dimensional flow over a BFS for a large range of Reynolds numbers. In particular, they investigated the flow for three-dimensionality, steadiness and symmetry. Furthermore, they characterized the flow by the locations of a number of recirculation zones as a function of the Reynolds numbers. Their paper constitutes a good framework for comparison with numerical simulations. Let us briefly describe the experimental set-up, the projection of which resembles very much to the two-dimensional benchmark problem. The dimensions have been used for our numerical simulations, as depicted in Figure 8.2.

The experiments were carried out in an air-driven flow channel incorporating a two-dimensional BFS with an expansion ratio of 1:1.943. The downstream channel has an aspect ratio of 18:1. The inlet section of the experimental setting was such that the flow is two-dimensional and fully developed when arriving at the step. The flow sticks to the solid boundaries, leading to two boundary layers in the transversal (z) direction. In this chapter, we will non-dimensionalize all the lengths by the height of the downstream section. The Reynolds number is based on the average inlet velocity (two-third of $U_{\max}$, $U_{\max} = 1$), a characteristic length which is chosen as two times the inlet height, and the kinematic viscosity ν:

$$Re = \frac{0.68647}{\nu}. \tag{8.1}$$

Armaly et al. predict that the flow is symmetric for all Reynolds numbers $Re < 8000$. Furthermore, the flow is found to be two-dimensional for $Re < 400$ and $Re > 6600$.

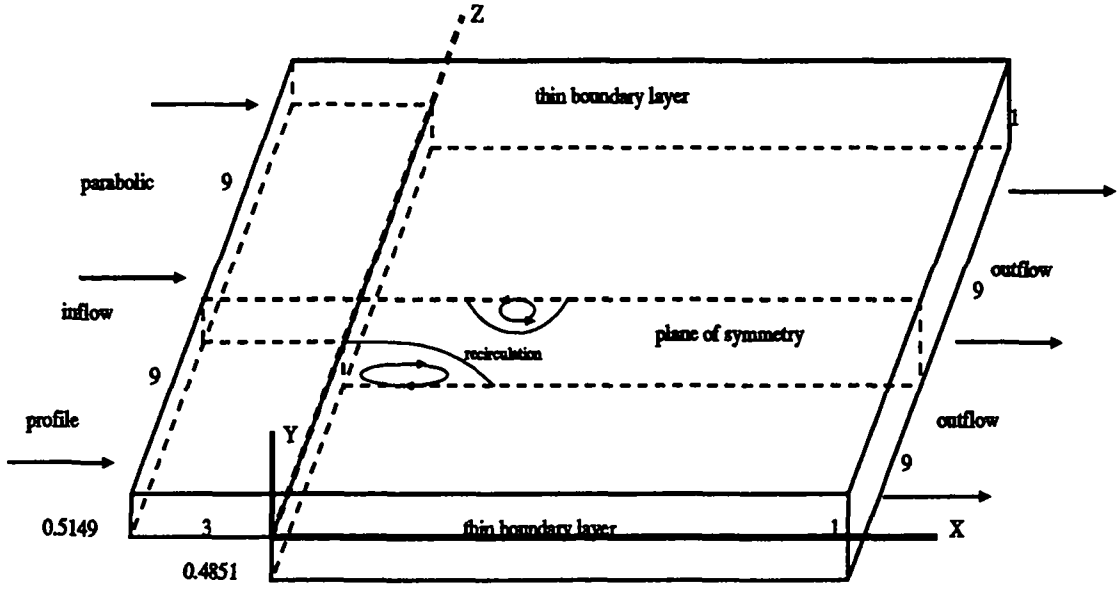

Figure 8.2: **The geometry of the backward-facing step with a 1:1.94 expansion ratio. The inflow profile is given as the tensor product of a parabola, which is zero at the walls and one at the center, and a Blasius boundary-layer profile characterized by $\delta_{.99} = 0.50$. The size of the geometry behind the step is 19, which is large enough to ensure fully developed outflow. For the range of Reynolds numbers that we consider, two recirculation zones are of interest. Their locations are indicated in the plane of symmetry.**

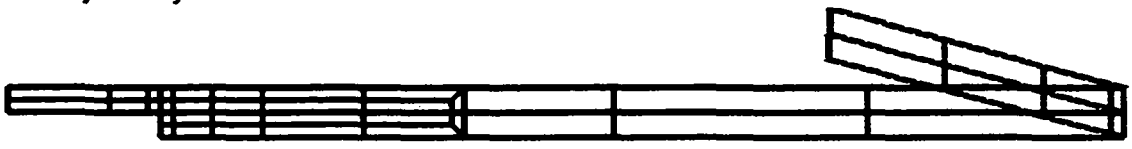

Figure 8.3: **Spectral element distribution. True aspect ratio.**

For $400 \leq Re \leq 6600$ three-dimensional effects have been observed. Each value of the Reynolds number is characterized by a certain length of the recirculation zone. In fact, there are three of them; The first one is located at the bottom half, directly downstream of the step. The second one was measured at the upper wall downstream of the expansion for $400 \leq Re \leq 6600$. A third recirculation zone occurs at the bottom wall, just downstream of the first one, for $1200 \leq Re \leq 2300$. The positions of these zones have been reported in detail.

There is a discrepancy between the results reported in Gresho and al. [35] and the experimental results for $Re > 400$, due to the three-dimensionality of the flow. Therefore, it seems interesting to apply our spectral element code to the three-dimensional BFS flow for $Re = 648$, which is very well documented in the paper of Armaly. In this text, we will go up to a Reynolds number of 343, leaving the problem at $Re = 648$ for the near future. To the author's knowledge, reliable numerical 3D simulations are rare. The spectral element discretization technique is very suited for this kind of fluid-flow problems. Mesh refinement at places where accurate results are essential (in the

boundary layers and behind the step) is obtained by taking many, small spectral elements. Larger elements can be used at the outflow and close to the plane of symmetry, see Figure 8.3.

In Armaly et al., streamwise velocity profiles are supplied along a number of cross-lines ("observation lines") in the spanwise direction. Their conclusion on two- or three-dimensionality is based on these streamwise velocity transverses. In this chapter, we will also supply these profiles. Armaly et al. call a flow three dimensional if the streamwise velocity component is constant (away from the boundary layer) in the spanwise direction. However, strictly speaking, the flow is always three-dimensional, since there is a dependence on x, on y (the parabolic profile), and on z (boundary layer). Another way of defining three dimensionality is by investigating the spanwise velocity component, either locally along the observation lines, or globally, by the parameter η, defined by

$$\eta = \frac{\| \underline{u}_3 \|_{L^2}}{\| \underline{u}_1 \|_{L^2} + \| \underline{u}_2 \|_{L^2} + \| \underline{u}_3 \|_{L^2}}. \tag{8.2}$$

The three-dimensional phenomena that we observe in our simulations are recognized by both definitions; the streamwise velocity component is not constant in the spanwise direction, and also the spanwise velocity component is not zero, leading to $\eta \neq 0$.

The BFS geometry that we used for our numerical simulations is displayed in Figure 8.2. The dimensions are essentially those used by Armaly et al., although the outflow section is slightly shorter because the range of Reynolds numbers that we will consider is smaller. According to the experimental data, we can safely assume that the flow is symmetric. Hence, the spectral element simulations have been performed in a half geometry. The two recirculation zones of interest for the range of Reynolds numbers that we consider are indicated in the plane of symmetry. The inflow boundary condition at $x = -3$ is the tensor product of a parabola and a Blasius profile:

$$\underline{u}(-3, y, z) = (4.0035323 \cdot b(-3, z) \cdot (y + 0.48514851)(0.51485148 - y), 0, 0)^T. \tag{8.3}$$

The Blasius profile $b(x, z)$ mimics a laminar boundary layer and is characterized by its thickness $\delta_{.99}(x)$, which represents the z-value at which the boundary layer attains 99% of its maximum (see e.g. Ryhming [66, pp. 216–226]). The boundary-layer thickness is known to grow like $O(x/Re)^{1/2}$. We set $\delta_{.99}(-3) = 0.50 \approx 9/\sqrt{343}$ (9 is the distance between the wall and the plane of symmetry).

In order to, eventually, obtain results for $Re = 648$, we have computed intermediate results at $Re = 172$ and $Re = 343$. These simulation are presented in the next sections.

8.2.2 Backward-facing-step flow at $Re = 172$

The initial solution for the backward-facing-step flow at $Re = 172$ is the solution of the Stokes problem. The spectral element mesh consists of 128 elements of degree 9 for the velocity and 7 for the pressure (see Figure 8.3). The elements are concentrated

around the step. There are four layers of elements in the z-direction close to the step and only two more downstream. The number of elements in the x, y-plane is 32, and four in the z-direction. We respect the rule-of-the-thumb that at least 10 spectral grid points are required to represent a boundary layer, i.e. the lengths of the elements in the z-direction are respectively 0.5, 1.5, 3, and 4. The large aspect ratio of the elements and the considerable difference in size in the x- and z-direction influence the condition numbers of the operators negatively. The simulation has been carried out by the pressure-correction method in conjunction with a subcycling scheme (BDF3/RK4, $M = 3$). The third-order projection method has been applied to the converged solution, but did not yield any modification of the results. The tolerances for the pressure and velocities have been chosen as $3_{10^{-9}}$ and $3_{10^{-6}}$ respectively; the time step is about 0.008. The simulation is proceeded until a steady state has been encountered, viz., until the streamwise velocity component at a number of predefined points in the geometry does no longer significantly change in time.

For reasons of lay-out, the figures representing the simulations at $Re = 172$ and $Re = 343$ are given in Section 8.2.4. In Figure 8.5, the streamwise velocity component is displayed in the plane of symmetry. The black zone represents negative velocities. The length of the recirculation zone corresponds to the experimental results of Armaly et al. and to the two-dimensional computations of Kim and Moin [43], see Figure 8.4. In order to get an idea of the flow pattern at the recirculation zone, the streamwise velocity component is represented in Figures 8.6 and 8.7, corresponding to the cross sections $x = 1.00$ (at the middle of the zone) and $x = 2.00$ (at the end of the zone), respectively. The spanwise velocity component is plotted at the former cross section ($x = 1.00$), according to Figure 8.8. An unexpected phenomenon is encountered in this figure embodied by a non-negligible spanwise velocity component which seems to have its origin in the intersection of the recirculation zone and the boundary layer. Furthermore, in Figure 8.7 we see that the form of the zone of negative velocities changes in the boundary layer, visualized by the small "dip" near the wall. From these observations, we conclude that, despite the small value of η ($\eta = 9.2_{10^{-5}}$), three-dimensional effects significantly influence the flow pattern. Their effect, however, is mainly limited to the boundary layer, explaining the good qualitative agreement with the experimental data as regards the length of the zone of recirculation. Three-dimensionality is more pronounced for the case $Re = 343$ and will be discussed again in the next subsection. Finally the streamwise velocity component is plotted at the six observation lines downstream of the separation zone (Figures 8.9 and 8.10). These figures show that the flow develops rapidly downstream of the recirculation zone, and confirm the small "overshoot" at the end of the boundary layer, which was also observed as a dip in the recirculation zone (Figures 8.6 and 8.7).

8.2.3 Backward-facing-step flow at $Re = 343$

The simulation of the backward-facing step flow at the Reynolds number $Re = 343$ has been performed with the same mesh and parameters as presented in the previous subsection. The transient is very long, as could be expected from the discussion in Section 8.2.1. The streamwise velocity in the plane of symmetry is represented in Figure 8.11. Again, a good qualitative agreement as regards the length of the recirculation

zone (see also Figure 8.4) with other computations and experiments is observed. The recirculation zone is shown for three cross sections in Figures 8.12, 8.13, and 8.14. Like for the plots at $Re = 172$, we see that the boundary-layer dynamics play an important role. Close to the step (Figure 8.12), the form of the recirculation zone and boundary layer are as expected, except for the small dip near the wall. Further downstream of the step, this dip grows and, eventually, negative velocities appear along a large part of the side wall. Although the flow near the plane of symmetry remains apparently unaffected, we see that the influence of the boundary layer stretches out much further than for $Re = 172$. The three-dimensionality is confirmed by Figure 8.15. From the streamwise velocity components along the observation lines (see Figures 8.16 and 8.17), we learn that the overshoot is still present far downstream of the recirculation zone. In particular, the observation line for $x = 3.22$, $y = 0.75$, shows the vertical recirculation zone at the side wall. Although the value of η is again rather small ($\eta = 1.8_{10^{-4}}$), Figure 8.18 indicates that three-dimensional effects are not limited to the boundary layer. Moreover, the spanwise velocity component also changes sign near the side wall.

From the above described simulations, we conclude that the global behaviour corresponds to the experimental data, but also that the modeling of boundary layer seems to induce three-dimensional effects for Reynolds numbers which do not agree with the observations of Armaly et al. In fact, many of the observed features were also found in the experimental setting, but at a higher Reynolds number. It should be noted that this premature transition to three-dimensionality is confirmed by computations of Sagaut [67], who used the PEGASE code developed at the aerodynamics division of ONERA, Chatillon, France.

In order to confident in our results, we first have to determine the quality of our simulations. To this end, we repeated the computation on a finer grid, using a polynomial degree of $N = 11$. The distribution of the spectral elements remained the same. The initial solution of this "fine-grid" simulation is formed by the interpolated results of the "coarse" grid. The steady-state solutions are compared in Figures 8.16 and 8.17, from which we conclude that the spatial resolution is sufficient. The influence of the modeling of the boundary layer has also been investigated. It should be noted that the equation for the Blasius profile corresponds to the modeling of a boundary layer over a flat plate. Therefore, we suggest that the same simulations are repeated with a longer inflow channel, such that the flow can settle before entering the step-region. Simultaneously, we take a thicker boundary layer to investigate the influence on three-dimensionality. We found out that these two modifications only induce small differences compared to the original case, and do not give a reason to change our conclusions. The abundant tests of spatial and temporal accuracy of the code, as presented in previous chapters, and the agreement of the lengths of the recirculation zone make us confident that the results are correct from a numerical viewpoint.

8.2.4 Figures

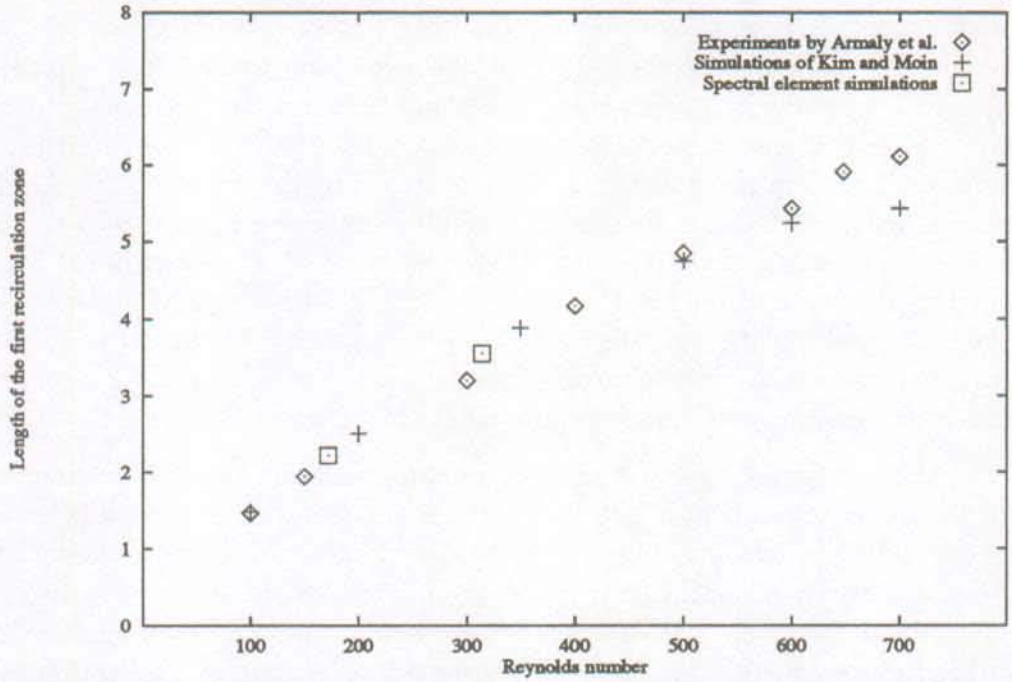

Figure 8.4: Comparison of the length of the first recirculation zone as a function of the Reynolds number.

Figure 8.5: $Re = 172$. Streamwise velocity component in the symmetry plane. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

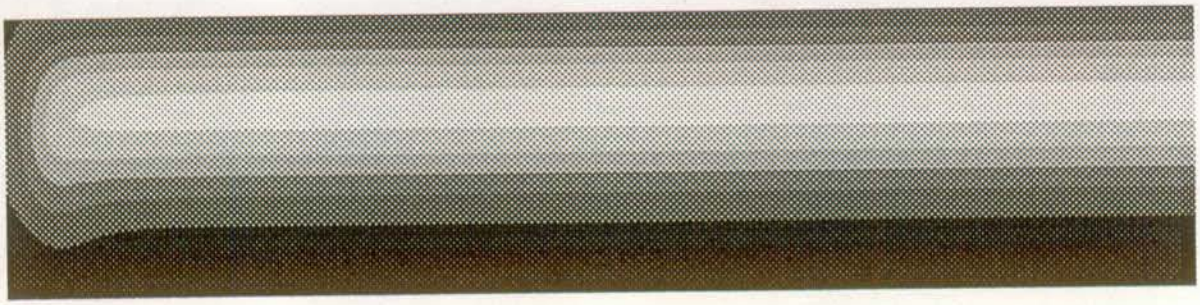

Figure 8.6: $Re = 172$. Streamwise velocity component at $x = 1.00$, half-way the recirculation zone. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

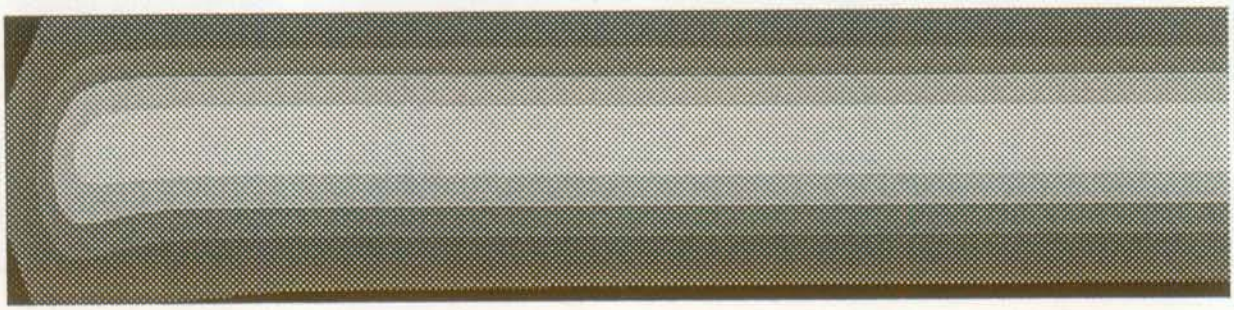

Figure 8.7: $Re = 172$. Streamwise velocity component at $x = 2.00$, at the end of the recirculation zone. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

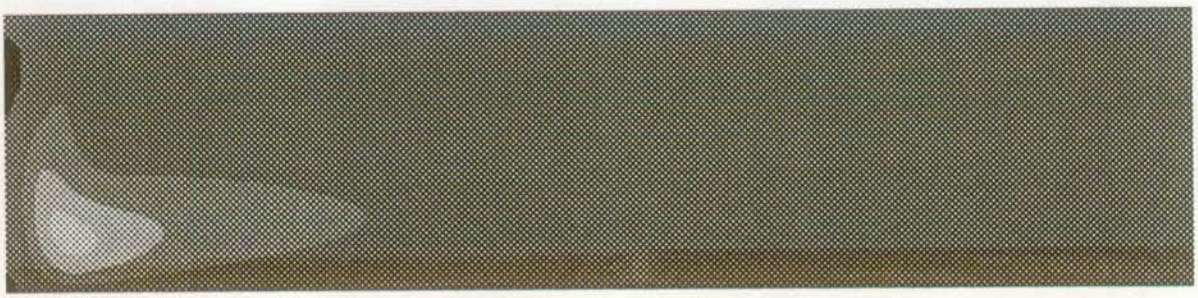

Figure 8.8: $Re = 172$. Spanwise velocity component at $x = 1.00$, half-way the recirculation zone. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.02 each.

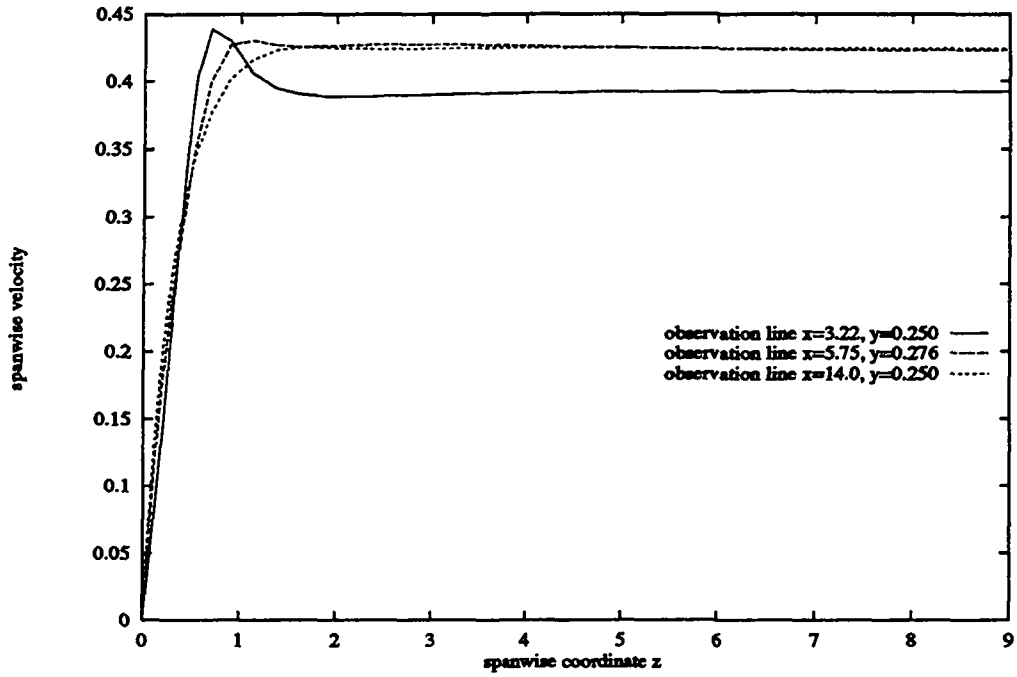

Figure 8.9: $Re = 172$. Streamwise velocity component along the three observation lines in the lower part of the geometry, downstream of the recirculation zone.

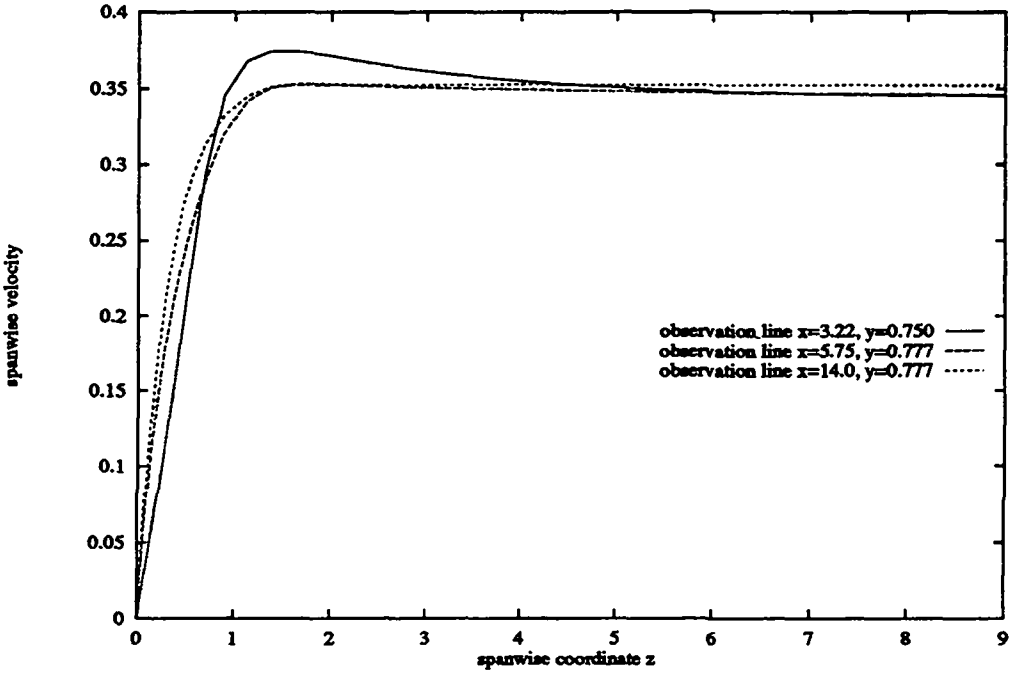

Figure 8.10: $Re = 172$. Streamwise velocity component along the three observation lines in the upper part of the geometry, downstream of the recirculation zone.

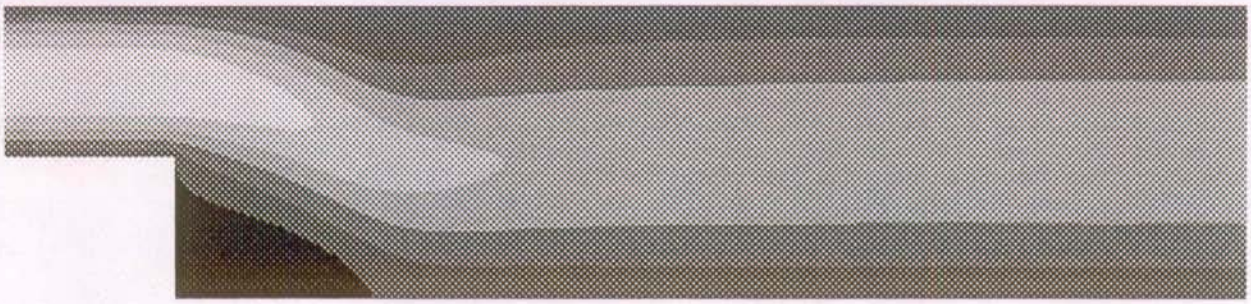

Figure 8.11: $Re = 343$. Streamwise velocity component in the symmetry plane. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

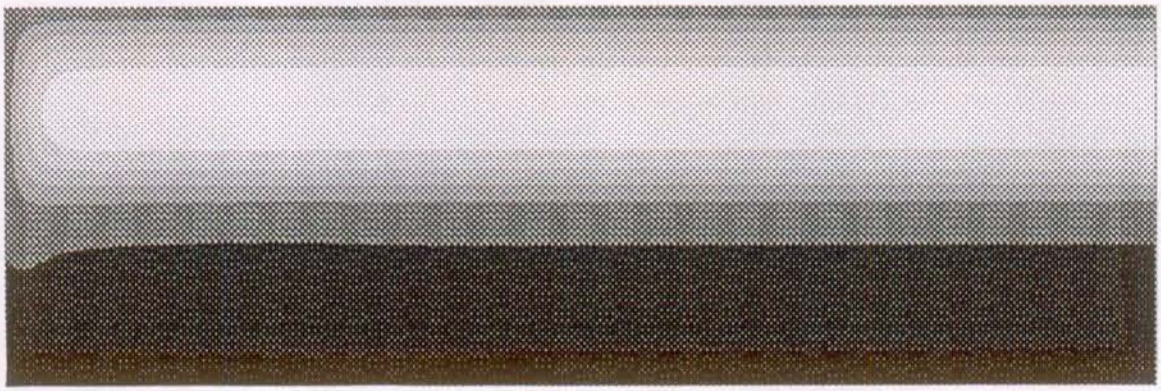

Figure 8.12: $Re = 343$. Streamwise velocity component at $x = 0.26$, just downstream of the step. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

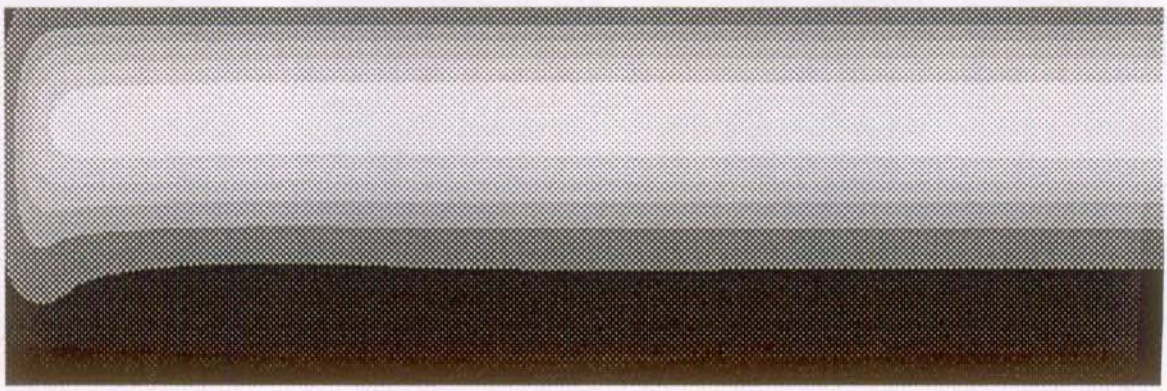

Figure 8.13: $Re = 343$. Streamwise velocity component at $x = 0.99$, at about one-third of the recirculation zone. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

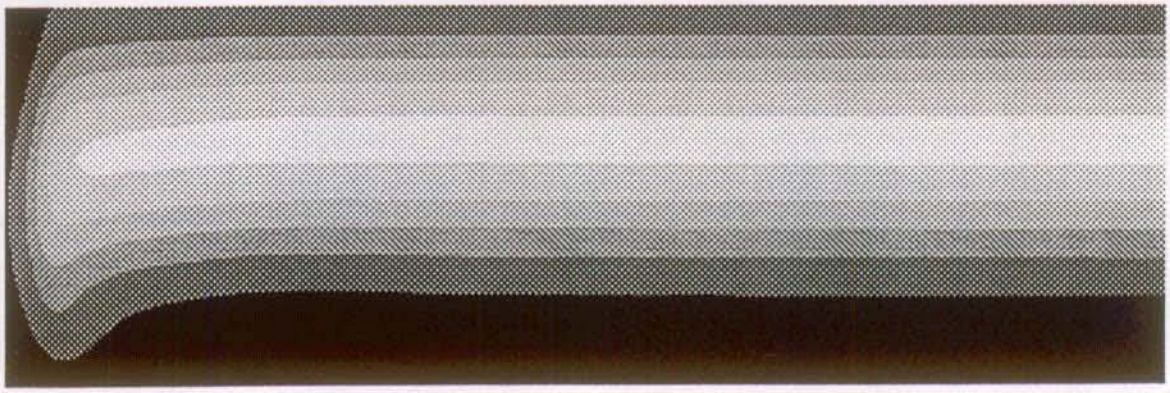

Figure 8.14: $Re = 343$. Streamwise velocity component at $x = 2.00$, at about two-thirds of the recirculation zone. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.2 each.

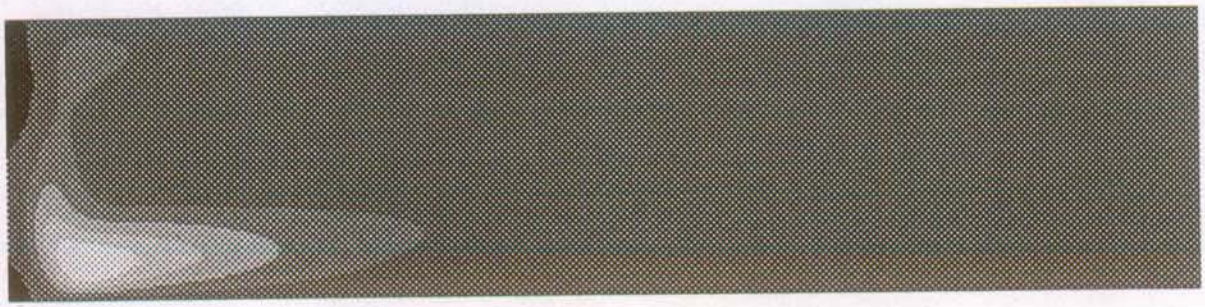

Figure 8.15: $Re = 343$. Spanwise velocity component at $x = 2.00$, at about two-thirds of the recirculation zone. The black zones represent negative values of the velocity; the consecutive shades of gray indicate a velocity increase of 0.02 each.

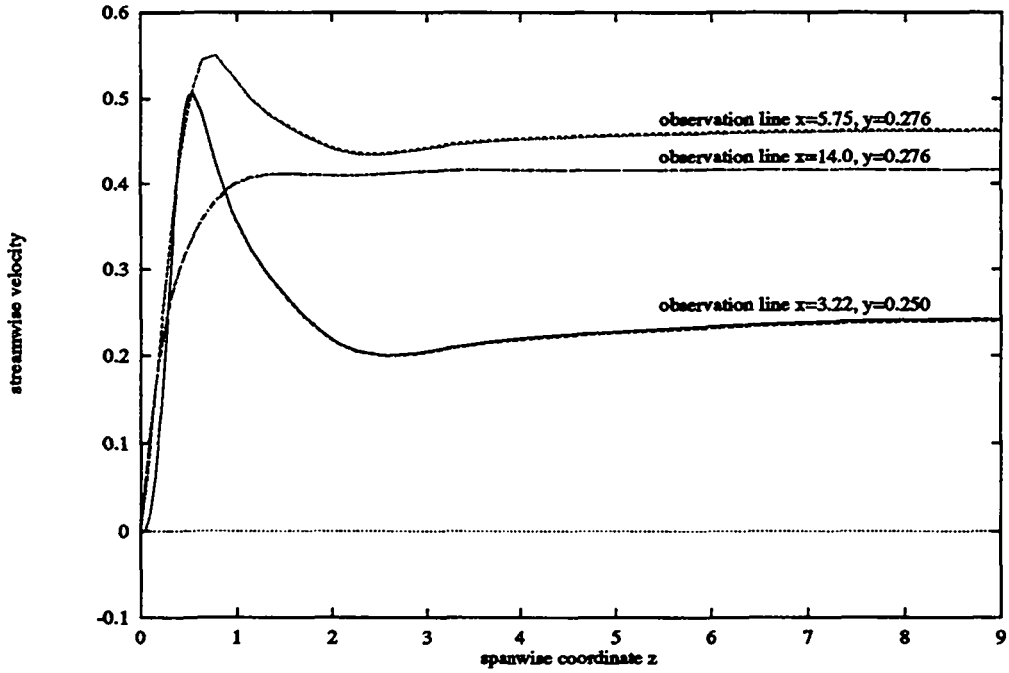

Figure 8.16: $Re = 343$. Streamwise velocity component along the three observation lines in the lower part of the geometry, downstream of the recirculation zone. The results for $N = 9$ and $N = 11$ are given, but are difficult to distinguish.

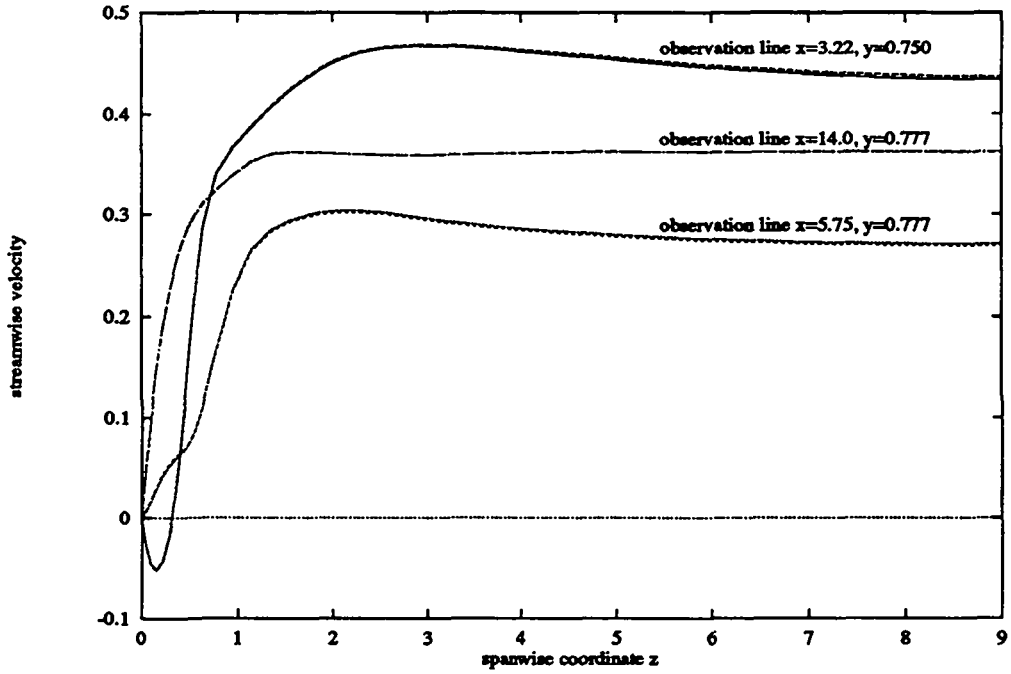

Figure 8.17: $Re = 343$. Streamwise velocity component along the three observation lines in the upper part of the geometry, downstream of the recirculation zone. The results for $N = 9$ and $N = 11$ are given, but are difficult to distinguish.

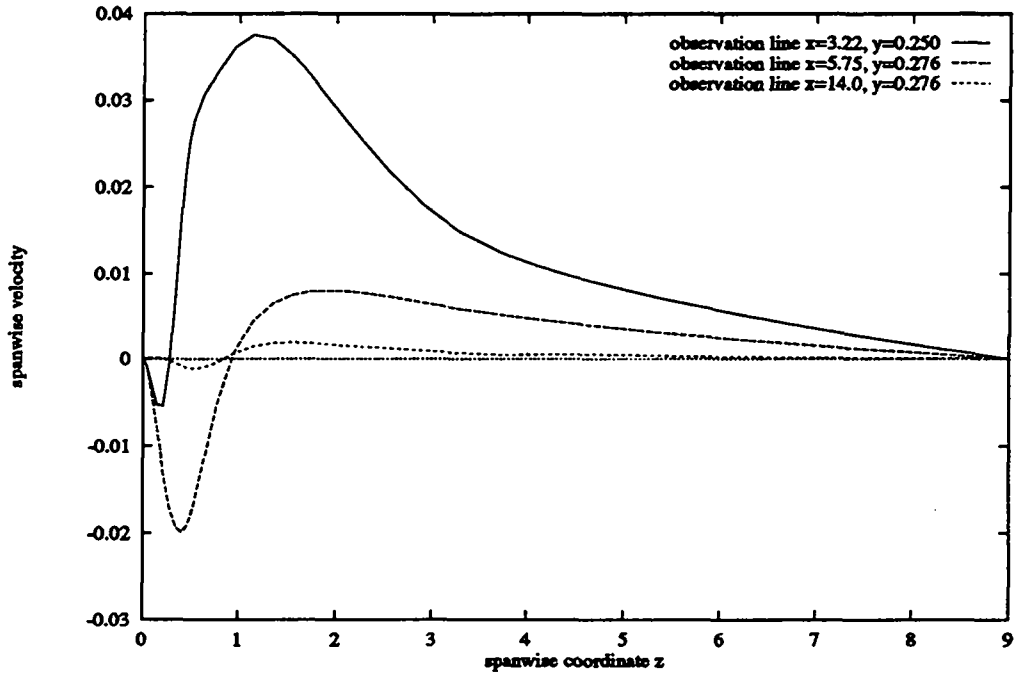

Figure 8.18: $Re = 343$. **Spanwise velocity component along the three observation lines in the lower part of the geometry, downstream of the recirculation zone.**

Chapter 9

Conclusions

In the previous chapters, the construction and validation of a spectral element solver have been described. The Legendre spectral element discretization technique has become a well-established tool for numerical simulations. Therefore, our attention has been focused on the development of high-order time-integration schemes with advantageous stability characteristics and on the construction of efficient iterative solution methods for the discrete systems. The implementation on parallel computers is a third point that has received much attention. Although the spectral element method belongs to the class of finite element methods, most of the difficulties that we encounter are typically "spectral", like, for instance, high condition numbers and long-range coupling. There are also substantial differences with the h-p finite element method which have an impact on the structure of the operators. Therefore, many of the presented algorithms apply only to spectral element discretizations, and some of them are even limited to the particular choice for the $\mathcal{P}_N - \mathcal{P}_{N-2}$ spaces. With this respect, we refer in particular to the Helmholtz solver and to the pressure preconditioners.

As a starting point, we have used the Uzawa method to decouple the velocities from the pressure. Unfortunately, this method has a major inconvenience, expressed by the solution of the pressure system. Two nested, iterative methods lead to very long computation times. A possible solution to this problem is to accelerate the inversion of the Helmholtz system, to develop efficient preconditioners for the Uzawa pressure operator, or both. The former proposition has been accomplished by the direct solution of the Helmholtz equations, based on rapid diagonalization methods. Their application is, however, limited to very simple geometries, but could be useful for the simulation of turbulent duct flows. The generality is augmented by introducing a Schur complement method which separates the Helmholtz system in K local problems and an interface problem. The most general solution is the incomplete Schur preconditioner, which can be used within the framework of deformed geometries. We have to remark that the parallelization of these fast Helmholtz solvers is not evident and deserves further study. Another way to accelerate the Uzawa method is by proper preconditioning. To this end, a two-stage preconditioning technique has been developed. The first stage effectively reduces the number of pressure iterations and, hence, the number of Helmholtz inversions. The second stage consists of local, elemental solves.

Another option is to replace the Uzawa method by a high-order projection method,

applied to the discrete equations. The resulting equations for the velocities and the pressure are no longer equivalent to the coupled Navier-Stokes system, but nested iterative solves are avoided. The time error that is introduced by the projection method is proportional to $(\Delta t/Re)^q$. We have followed the suggestion of Blair Perot [9] to construct projection methods for which $q = 3$. The error can even be diminished to $\Delta t(\Delta t/Re)^q$ by computing a correction to the pressure, instead of the pressure. The numerical results show that, indeed, projection methods are very accurate and cheap, especially for moderate to high Reynolds numbers.

The projection method gives rise to a pressure operator that can be preconditioned along the same lines as the second stage of the Uzawa pressure matrix. It remains an open question if the local solves should be combined with the construction of the coarse pressure skeleton, as has been proposed by Rønquist [63], or not. All the numerical test in this thesis indicate that this approach leads to a lower number of pressure iterations, but also to a higher cpu time. However, extrapolation of the results, combined with the knowledge that the condition number of the preconditioned operator is independent of K, predict that for $K \gg 1$ the coarse/fine solves à la Rønquist should be preferred.

The accuracy in time depends on the order of the time schemes for the implicit and explicit terms, the order of the linear/nonlinear splitting and the order of the decoupling method. In this thesis we have presented a solver with a global order of accuracy in time of three, implying that all of the aforementioned components are at least third-order. The ingredients for a fourth-order method are also given. The implication of an accurate time-integration scheme is that the time step is no longer governed by considerations of precision, but of stability. We have adopted the operator-integration-factor splitting method [48] for its large stability zones, which have been determined by an original analysis. The explicit terms are integrated by the standard Runge-Kutta method, which possesses almost optimal stability properties for a four-stage method. In fact, it can be shown that the optimal β_{imag} for an m-stage Runge-Kutta method is $\beta_{imag} = m - 1$. The use of stabilized Runge-Kutta methods with $m > 4$ will further enhance the stability, but it should be investigated whether this also improves the performance.

The spectral element solver has been implemented on the Cray T3D in Lausanne. Initially, the most logical choice to manage the communication was by standard PVM. Indeed, a good parallel efficiency has been obtained for values of N that are also used for the simulation on serial computers. Moreover, the efficiency decays only slightly for an increasing number of spectral elements and the code is portable to most other MPP's. Some additional tests have pointed out, however, that the use of shared memory operations, such as "shmem-get", leads to a much higher parallel performance: On a typical test problem we have even obtained an efficiency which is independent of the number of spectral elements. For future developments it is recommended to forget about portability and to focus on architecture-specific communication routines, which have proven to be superior on the Cray T3D and on the Intel Paragon.

Improvements are possible on many other levels. As far as the spectral element method itself is concerned, it would be interesting to investigate the recent developments for stabilized spectral discretizations, based on finite element bubble functions (see e.g. Canuto [13]). In a recent paper, Canuto and Van Kemenade [15] show that

the addition of local bubble functions not only cures the instabilities due to convection, but also removes spurious pressure modes in the $\mathcal{P}_N - \mathcal{P}_N$ formulation. This approach might imply that alternative preconditioners have to be found. The number of iterations for the pressure operator can be reduced by supplying good initial guesses. One way to do this has been proposed by Fischer [25]. His approach is based on projection and is especially effective in the case of flows with a significant dynamic activity. This projection technique exhibits, for example, a 50% reduction of cpu time for the simulation of an impulsively started flow past cylinder.

Scalability is an important issue when we want to simulate problems consisting of many spectral elements ($K = O(10^3)$). It is desirable that the computational complexity does not augment as long as each spectral element can be mapped on one processor. (The reader is referred to [62, Chapter 13] for a discussion on scalability in parallel processing.) The coarse/fine pressure subspace preconditioner is an example of a scalable algorithm, since the condition number of the preconditioned operator is independent of K. Furthermore, the results obtained by shared memory PVM indicate that the parallel efficiency remains constant for increasing values of K. The only part of the solver that does not respect the concept of scalability is the Helmholtz solver. The dependence of the condition number of the Schur preconditioner on K should be studied together with its parallelization. For an alternative approach we refer to Rønquist [64]. Throughout this thesis we have only considered algorithms with an operation count of $O(N^4)$. Preconditioning of the Helmholtz and pressure operators leads to a maximum increase of the number of iterations which is linear with N. Consequently, the theoretical relation between N and the cpu time is of order N^5, when we do not take the effect of N on the stability into account. In practice, however, we also see that the performance increases considerably with N.

We close this chapter with some practical considerations. The simulations as described in Chapter 8 could be performed much more effectively if we know how to distribute the spectral elements and how to choose N to obtain a sufficient spatial resolution. The only tool we dispose of before starting the computation is common sense. Error estimates could be computed by transforming the results to the Legendre space and by investigating the spectrum. This will give information whether a sufficient precision has been obtained, and the mesh could be adapted (refined) accordingly. Tuning of the solver is also a tricky issue: The optimal values for the time step, tolerances for iterative methods, etc. are unknown and can only be approximated by trial and error and by the "intuition" of the programmer. Rigorous methods to determine these parameters will certainly save a lot of cpu time.

Appendix A

Characteristic polynomials

BDF1/AB3

$$C(\zeta, \lambda_1, \lambda_2, \Delta t) = (1 - \Delta t\lambda_1)\zeta^3 + \frac{23\lambda_2\Delta t\zeta^2}{12} - \frac{4\lambda_2\Delta t\zeta}{3} + \frac{5\lambda_2\Delta t}{12} \tag{A.1}$$

BDF1/EX3

$$C(\zeta, \lambda_1, \lambda_2, \Delta t) = (1 - \Delta t\lambda_1)\zeta^3 + 3\lambda_2\Delta t\zeta^2 - 3\lambda_2\Delta t\zeta + \lambda_2\Delta t \tag{A.2}$$

BDF1/RK4, M=1

$$C(\zeta, \lambda_1, \lambda_2, \Delta t) = (1 - \Delta t\lambda_1)\zeta + \frac{24 + 24\lambda_2\Delta t + 12\lambda_2^2\Delta t^2 + 4\lambda_2^3\Delta t^3 + \lambda_2^4\Delta t^4}{24} \tag{A.3}$$

BDF1/RK4, M=2

$$C(\zeta, \lambda_1, \lambda_2, \Delta t) = (1 - \Delta t\lambda_1)\zeta + \left(\frac{384 + 192\lambda_2\Delta t + 48\lambda_2^2\Delta t^2 + 8\lambda_2^3\Delta t^3 + \lambda_2^4\Delta t^4}{384}\right)^2 \tag{A.4}$$

BDF1/RK4, M=3

$$C(\zeta,\lambda_1,\lambda_2,\Delta t) = (1-\Delta t\lambda_1)\zeta + \left(\frac{1944+648\lambda_2\Delta t+108\lambda_2^2\Delta t^2+8\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{1944}\right)^3 \tag{A.5}$$

BDF3/EX3

$$C(\zeta,\lambda_1,\lambda_2,\Delta t) = (11-6\Delta t\lambda_1)\zeta^3 - 18(1+\lambda_2\Delta t)\zeta^2 + 9(1+2\lambda_2\Delta t)\zeta - 2 - 6\lambda_2\Delta t \tag{A.6}$$

BDF3/RK4, M=1

$$\begin{aligned} C(\zeta,\lambda_1,\lambda_2,\Delta t) &= (11-6\Delta t\lambda_1)\zeta^3 \\ &- 18\left(\frac{24+24\lambda_2\Delta t+12\lambda_2^2\Delta t^2+4\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{24}\right)\zeta^2 \\ &+ 9\left(\frac{24+24\lambda_2\Delta t+12\lambda_2^2\Delta t^2+4\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{24}\right)^2\zeta \\ &- 2\left(\frac{24+24\lambda_2\Delta t+12\lambda_2^2\Delta t^2+4\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{24}\right)^3 \end{aligned} \tag{A.7}$$

BDF3/RK4, M=2

$$\begin{aligned} C(\zeta,\lambda_1,\lambda_2,\Delta t) &= (11-6\Delta t\lambda_1)\zeta^3 \\ &- 18\left(\frac{384+192\lambda_2\Delta t+48\lambda_2^2\Delta t^2+8\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{384}\right)^2\zeta^2 \\ &+ 9\left(\frac{384+192\lambda_2\Delta t+48\lambda_2^2\Delta t^2+8\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{384}\right)^4\zeta \\ &- 2\left(\frac{384+192\lambda_2\Delta t+48\lambda_2^2\Delta t^2+8\lambda_2^3\Delta t^3+\lambda_2^4\Delta t^4}{384}\right)^6 \end{aligned} \tag{A.8}$$

BDF3/RK4, M=3

$$
\begin{aligned}
C(\zeta, \lambda_1, \lambda_2, \Delta t) &= (11 - 6\Delta t\lambda_1)\zeta^3 \\
&- 18\left(\frac{1944 + 648\lambda_2\Delta t + 108\lambda_2^2\Delta t^2 + 8\lambda_2^3\Delta t^3 + \lambda_2^4\Delta t^4}{1944}\right)^3\zeta^2 \\
&+ 9\left(\frac{1944 + 648\lambda_2\Delta t + 108\lambda_2^2\Delta t^2 + 8\lambda_2^3\Delta t^3 + \lambda_2^4\Delta t^4}{1944}\right)^6\zeta \\
&- 2\left(\frac{1944 + 648\lambda_2\Delta t + 108\lambda_2^2\Delta t^2 + 8\lambda_2^3\Delta t^3 + \lambda_2^4\Delta t^4}{1944}\right)^9 \qquad \text{(A.9)}
\end{aligned}
$$

Bibliography

[1] Armaly, B.F., Durst, F., Pereira, J.C.F., and Schönung, B. (1983). Experimental and Theoretical Investigation of Backward-Facing Step Flow. *J. Fluid Mech.*, Vol. 127, pp 473-496.

[2] Arrow, K., Hurwicz, L., and Uzawa, H. (1968). *Studies in Nonlinear Programming.* Stanford University Press, Stanford.

[3] Ascher, U.M., Ruuth, S.J., and Wetton, B.T.R. (1995). Implicit-Explicit Methods for Time-Dependent PDE's. *SIAM J. Num. Anal.*, to appear in June 1995.

[4] Azaiez, M., and Coppoletta, G. (1992). Calcul de la Pression dans le Problème de Stokes par une Méthode Spectrale de "Quasi-Collocation" à Grille Unique sans Modes Parasites. *Annales Maghrébines de l'Ingénieur.*

[5] Babuška, I. (1973). The Finite Element Method with Lagrangian Multipliers, *Numer. Math.*. Vol 20, pp 179–192.

[6] Bernardi, C. and Maday, Y. (1992). *Approximations Spectrales de Problèmes aux Limites Elliptiques.* Springer-Verlag, Paris.

[7] Bernardi, C., and Maday, Y. (1988). A Collocation Method over Staggered Grids for the Stokes Problem. *Int. J. of Numer. Meth. Fluids*, Vol. 8, pp 537–557.

[8] Bjørstad, P.E., and Widlund, O.B.. (1986). Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures. *SIAM J. Numer. Anal.*, Vol. 23(6), pp 1097–1120.

[9] Blair Perot, J. (1993). An Analysis of the Fractional Step Method. *J. Comp. Phys.*, Vol 108, pp 51–58.

[10] Boukir, K. (1994). *Méthodes en Temps d'Ordre Elevé par Décomposition d'Opérateurs. Application aux Equations de Navier-Stokes.* PhD Thesis, Université de Pierre et Marie Curie, Paris 6.

[11] Brezzi, F. (1974). On the Existence, Uniqueness and Approximation of Saddle-point Problems Arising from Lagrangian Multipliers. *R.A.I.R.O. Anal. Numér.*, Vol. 8, pp 129–151.

[12] Cahouet, J., and Chabard, J.P. (1986). Some Fast 3D Finite Element Solvers for the Generalized Stokes Problem. *Internat. J. Numer. Meth. Fluids,* Vol. 8, pp 869-895.

[13] Canuto, C. (1994). Stabilization of Spectral Methods by Finite Element Bubble Functions. *Comput. Meth. Appl. Mech. Engrg.*, Vol 116, pp 39–51.

[14] Canuto, C., Hussaini, M.Y., Quarteroni, A., and Zang, T.A. (1988). *Spectral Methods in Fluid Dynamics.* Springer-Verlag, New York.

[15] Canuto, C., and Van Kemenade, V. (1995). Bubble-Stabilized Spectral Methods for the Incompressible Navier-Stokes Equations. Submitted to *Comp. Meth. Appl. Mech. Engrg.*

[16] Chan, T.F., and Goovaerts, D. (1989). Schur Complement Domain Decomposition Algorithms for Spectral Methods. *Applied Numer. Math., Special Issue on Spectral Multi-Domain Methods*, Vol. 6, pp 53–64.

[17] Chorin, A.J. (1968). Numerical Solution of the Navier-Stokes Equations. *Math. Comp.*, Vol. 22, pp 745–761.

[18] Couzy, W., Deville, M.O. (1993) *Solution numérique de haute précision des équations de transfert du type diffusif et advectif-diffusif en géométrie complexe par les techniques de décomposition de domaines sur les ordinateurs parallèles.* Activity report Catholic University of Louvain (in English).

[19] Couzy, W,. and Deville, M.O. (1995). A Fast Schur Complement Method for the Spectral Element Discretization of the Incompressible Navier-Stokes Equations. *J. of Comp. Phys.*, Vol. 116, pp 135–142.

[20] Couzy, W., and Deville, M.O. (1994). Spectral-Element Preconditioners for the Uzawa Pressure Operator Applied to Incompressible Flows. *J. Sci. Comp.*, Vol. 9, No. 2, pp 107–122.

[21] Couzy, W., and Deville, M.O. (1994). Iterative Solution Technique for Spectral-Element Pressure Operators at High Reynolds Numbers. *Proceedings of the Second European Computational Fluid Dynamics Conference*, (Ed. Wagner, S. et al.) Stuttgart, Germany, pp 613–618.

[22] Dahl, O., and Wille, S.Ø. (1992). An ILU Preconditioner with Coupled Node Fill-in for Iterative Solution of the Mixed Finite Element Formulation of the 2D and 3D Navier-Stokes Equations. *Int. J. Num. Meth. Fluids*, Vol. 15, pp 525-544.

[23] Demaret, P., and Deville, M.O. (1989). Chebyshev Pseudospectral Solution of the Stokes Equations Using Finite-Element Preconditioning. *J. Comp. Phys.*, Vol. 83, pp 463-484.

[24] Deville, M.O., and Mund, E.H. (1990). Finite Element Preconditioning of Pseudospectral Solutions of Elliptic Problems. *SIAM J. Sci. Stat. Comp.*, Vol. 12, pp 311–342.

[25] Fischer, P.F. (1993). Projection Techniques for the Iterative Solution of $A\underline{x} = \underline{b}$ with Successive Right-Hand Sides. *ICASE Report*, No. 93-90.

[26] Fischer, P.F., and Patera, A.T. (1991). Parallel Spectral Element Solution of the Stokes Problem. *J. Comp. Phys.*, Vol. 92, pp 380-421.

[27] Fischer, P.F., and Patera, A.T. (1994). Parallel Simulation of Viscous Incompressible Flows. *Annu. Rev. Fluid Mech.*, Vol. 26, pp 483–527.

[28] Fischer, P.F., and Rønquist, E.M. (1994). Spectral Element Methods for Large Scale Parallel Navier-Stokes Calculations. *Comp. Meth. Appl. Mech. Engr.*, Vol. 116, pp 414-443.

[29] Fischer, P.F., Rønquist, E.M., Dewey, D., and Patera, A.T. (1987). *Spectral Element Methods; Algorithms and Architectures.* First Int. Symp. on Domain Decomposition Methods for Partial Differential Equations, SIAM, p 397.

[30] Gartling, D.K. (1990). A Test Problem for Outflow Boundary Conditions - Flow over a Backward-Facing Step. *Int. J. Num. Meth. Fluids*, Vol. 11, pp 953-967.

[31] Gear, G.W. (1970). *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, New Jersey.

[32] Gervasio, P., Ovtchinnikov, E., and Quarteroni, A. (1994). The Spectral Projection Decomposition Method for Elliptic Equations. *Report 128/P*, Dipartimento di Matematica, Politecnico di Milano.

[33] Golub, G.H., and Van Loan, C.F. (1983). *Matrix Computations.* North Oxford Academic, Oxford.

[34] Gordon, W.J., and Hall, C.A. (1973). Transfinite Element Methods: Blending Functions Interpolated over Arbitrary Curved Element Domains. *Numer. Math.*, Vol. 21, pp 109-129.

[35] Gresho, P.M., Gartling, D.K., Torczynski, J.R., Cliffe, K.A., Winters, K.H., Garratt, T.J., Spence, A., and Goodrich, J.W. (1993). Is the Steady Viscous Incompressible Two-Dimensional Flow over a Backward-Facing Step at Re=800 Stable? *Int. J. Num. Meth. Fluids*, Vol. 17, pp 501–541.

[36] Haidvogel, D.B., and Zang, T. (1979). The Accurate Solution of Poisson's Equation by Expansion in Chebyshev Polynomials. *J. Comp. Phys.*, Vol. 30, pp 167-180.

[37] Haldenwang, P., Labrosse, G., Abboudi, S.A., and Deville, M.O. (1984). Chebyshev 3-D Spectral and 2-D Pseudospectral Solvers for the Helmholtz Equation. *J. of Comp. Phys.*, Vol. 55, pp 115–128.

[38] Ho, L-W., Maday, Y., Patera, A.T., and Rønquist E.M., (1989). A High-Order Lagrangian-Decoupling Method for the Incompressible Navier-Stokes Equations. *Proceedings of the ICOSAHOM '89 Conference, Eds. Canuto, C., and Quarteroni, A.*, Como, Italy, pp 65–90.

[39] Hockney, R.W., and Jesshope, C.R. (1981). *Parallel Computers. Architecture, Programming and Algorithms.* Adam Hilger Ltd., Bristol.

[40] Kaiktsis, L., Karniadakis, G.E.M., and Orszag, S. (1991). Onset of Three-Dimensionality, Equilibria, and Early Transition in Flow over a Backward-Facing Step. *J. Fluid Mech.*, Vol. 231, pp 501–528.

[41] Kaiktsis, L., Karniadakis, G.E.M., and Orszag, S. (1994). Supercritical States and Convective Instabilities in Two-Dimensional Flow over a Backward-Facing Step. Submitted to *J. Fluid Mech.*

[42] Karniadakis, G.E.M., Israeli, M., and Orszag, S.A. (1991). High-Order Splitting Methods for the Incompressible Navier-Stokes Equations. *J. Comp. Phys.*, Vol. 97, pp 414-443.

[43] Kim, J., and Moin, P. (1985). Applications of a Fractional Step Method to Incompressible Navier-Stokes Equations. *J. Comp. Phys.*, Vol. 59, pp 308–323.

[44] Lynch, R.E., Rice, J.R., and Thomas, D.H. (1964). Direct Solution of Partial Difference Equations by Tensor Product Methods. *Numerische Mathematik*, Vol. 6, pp 185-199.

[45] Maday, Y. (1994). *Private communication.*

[46] Maday, Y., Meiron, D., Patera, A.T. and Rønquist, E.M. (1993). Analysis of Iterative Methods for the Steady and Unsteady Stokes Problem: Applications to Spectral Element Discretizations. *SIAM J. Sci. Comp.*, Vol. 14, No. 2, pp 310-337.

[47] Maday, Y., and Patera, A.T. (1989). Spectral Element Methods for the Incompressible Navier-Stokes Equations. *State-of-the-Art Surveys on Computational Mechanics* (Ed. Noor, A.K. and Oden, J.T.), ASME, pp 71–143.

[48] Maday, Y., Patera, A.T. and Rønquist, E.M. (1990). An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow. *J. Sci. Comp.*, Vol. 5, No. 4, pp 263–292.

[49] Maday, Y., Patera, A.T. and Rønquist, E.M. (1992). The $\mathcal{P}_N - \mathcal{P}_{N-2}$ Method for the Approximation of the Stokes Problem. *submitted to Numer. Math.*

[50] Maday, Y., and Rønquist, E.M. (1989). Optimal Error Analysis of Spectral Methods with Emphasis on Non-Constant Coefficients and Deformed Geometries. *Proceedings of the ICOSAHOM '89 Conference, Eds. Canuto, C., and Quarteroni, A.*, Como, Italy, pp 91–115.

[51] Magère, E. (1994). *Résolution de l'Equation de la Chaleur en Géometrie Axisymétrique par la Méthode des Eléments Spectraux.* Technical report T-94-15, IMHEF, EPFL, Lausanne, Switzerland.

[52] Métivet, B. (1987). *Résolution Spectrale des Equations de Navier-Stokes par une Méthode de Sous-Domaines Courbes.* PhD Thesis, Université Pierre et Marie Curie, Paris 6, France.

[53] Meurant, G. (1987). Multitasking the Conjugate Gradient Method on the Cray X-MP/48. *Parallel Computing*, Vol. 5, pp 267–280.

[54] Morchoisne, Y. (1981). Pseudo-spectral Space-Time Calculations of Incompressible Viscous Flows. *AIAA paper No. 81-0109.*

[55] Orszag, S.A. (1980). Spectral Methods for Problems in Complex Geometries. *J. Comp. Phys.*, Vol 37, p70.

[56] Orszag, S.A., Israeli, M., and Deville, M.O. (1986). Boundary Conditions for Incompressible Flows. *J. Sci. Comp.*, Vol. 1, pp 75-111.

[57] Patera, A.T. (1984). A Spectral Element Method for Fluid Dynamics; Laminar Flow in a Channel Expansion. *J. Comp. Phys.*, Vol. 54, pp 468-488.

[58] Patera, A.T. (1986). Fast Direct Poisson Solvers for High-Order Finite Element Discretizations in Rectangulary Decomposable Domains. *J. Comp. Phys.*, Vol. 65, pp 474–480.

[59] Phillips, T.N., and Roberts, G.W. (1993). The Treatment of Spurious Pressure Modes in Spectral Incompressible Flow Calculations. *J. Comp. Phys.*, Vol. 105, pp 150–164.

[60] Pinelli, A., and Vacca, A. (1994). Chebyshev Collocation Method and Multi Domain Decomposition for the Incompressible Navier-Stokes Equations. *Int. J. Num. Meth. Fluids*, Vol. 18 (No 8), pp 781–799.

[61] Pironneau, O. (1982). On the Transport Diffusion Algorithm and its Applications to the Navier-Stokes Equations. *Numer. Math.*, Vol. 38, pp 309–332.

[62] *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing* (Ed. Bailey, D.H. et al.), San Francisco, USA, 1995.

[63] Rønquist, E.M. (1991). A Domain Decompostion Method for Elliptic Boundary Value Problems: Application to Unsteady Incompressible Fluid Flow. *Proceedings of the Fifth Conference on Domain Decomposition Methods for Partial Differential Equations* (Ed. Keyes, D.E. et al.), Norfolk, USA, pp 545–557.

[64] Rønquist, E.M. (1993). An Elliptic Solver Based on Conjugate Gradients, Domain Decomposition and Deflation. Submitted to *SIAM J. Sci. Comp.*

[65] Rønquist, E.M. (1991). *Spectral Element Methods for the Unsteady Navier-Stokes Equations.* Lecture Series 1991-01, Von Karman Institute for Fluid Dynamics, Belgium.

[66] Ryhming, I.L. (1991). *Dynamique des Fluids.* Second Edition, Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland.

[67] Sagaut, P. (1995). *Private communication.*

[68] Sani, R.L., and Gresho, P.M. (1994). Résumé and Remarks on the Open Boundary Condition Minisymposium. *Int. J. Num. Meth. Fluids*, Vol. 18, pp 983–1008.

[69] Schneidesch, C.R. (1992). *Numerical Simulation of Incompressible Flows in Complex Geometries by Preconditioned Chebyshev Collocation.* PhD Thesis, Université Catholique de Louvain.

[70] Schumann, U. (1993). *Introduction to the Modelling of Turbulence.* Lecture Series 1993-02, Von Karman Institute for Fluid Dynamics, Belgium.

[71] Streett, C., and Hussaini, M.Y. (1987) A Numerical Simulation of Finite-Length Taylor-Couette Flow. *AIAA paper 87-1444.*

[72] Szabó, B., and Babuška, I. (1991). *Finite Element Analysis.* John Wiley and Sons, New York.

[73] Téman, R. (1969). Sur l'Approximation de la Solution de Navier-Stokes par la Méthode des Pas Fractionnaires. I, II. *Arch. Rat. Mech. Anal.*, Vol. 32, pp 135–153, pp 377–385.

[74] Timmermans, L.J.P. (1994). *Analysis of Spectral Element Methods with Application to Incompressible Flow.* PhD Thesis. Eindhoven University of Technology.

[75] Van Kan, J. (1986). A Second-Order Accurate Pressure-Correction Scheme for Viscous Incompressible Flow. *SIAM J. Sci. Stat. Comp.*, Vol. 7, pp 870–891.

[76] Zhou, R.Q.N. (1993). Preconditioned Finite Element Algorithms for 3D Stokes Flows. *Int. J. Num. Meth. Fluids*, Vol. 17, pp 667–685.

[77] Zone, O., Vanderstraeten, D., Henriksen, P., and Keunings, R. (1994). A Parallel Direct Solver for Implicit Finite Element Problems Based on Automatic Domain Decomposition. *Massively Parallel Processing: Applications and Development* (Eds. Dekker, G. et al.), Elsevier, Amsterdam, pp 809–816.

Curriculum Vitae

Wouter COUZY

- **Place and date of birth:** Landsmeer, The Netherlands, July 19th, 1967
- **Nationality:** Dutch
- **Sex:** male

Education

- **1985-1990** University of Amsterdam, The Netherlands.
  Degree in mathematics. General courses in mathematics and specialized projects in numerical mathematics and computer science. Graduation project: "Stable, parallel time-integration schemes", performed during a six-months stage at the Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.
- **1991-1995** Ecole Fédérale Polytechnique de Lausanne, Switzerland.
  PhD candidate at the Department of Mechanical Engeneering.

Professional Experience

- **1989-1991** Delft University of Technology, The Netherlands.
  Researcher at the Faculty of Technical Mathematics and Informatics. Field of research: Discretization of the unsteady, incompressible Navier-Stokes equations by a co-located finite volume method.
- **1991-1995** Catholic University of Louvain, Belgium.
  Assistant researcher at the Department of Applied Mechanics. Field of research: Spectral element discretizations of the unsteady, incompressible Navier-Stokes equations in three dimensions, with emphasis on stable and accurate time-integration schemes, efficient iterative solvers, and parallelization.

Publications

- Couzy, W. (1989). Performance Evaluation of Rosser's Method. *C.W.I. Technical Report NM-N9001.*

- Sommeijer, B.P., Couzy, W., and Van der Houwen, P.J. (1992). A-Stable Parallel Block Methods for Ordinary and Integro-Differential Equations. *Appl. Numer. Math.*, Vol. 9, pp 267–281.

- Van der Houwen, P.J., Sommeijer, B.P., and Couzy, W. (1992). Embedded Diagonally Implicit Runge-Kutta Algorithms on Parallel Computers. *Math. Comp.*, Vol. 58, No. 197, pp 135–159.

- Wilders, P., and Couzy, W. (1992). The Discretization of the 2D Incompressible Navier-Stokes Equations on a Co-Located Grid. *Proceedings of the First European Computational Fluid Dynamics Conference*, (Ed. Hirsch, Ch. et al.), Brussels, Belgium, pp 503-509.

- Couzy, W., and Deville, M.O. (1993). A Parallel Spectral Element Method for the 3D Navier-Stokes Equations. *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, (Ed. Sincovec, R.F. et al.), Norfolk (Va), USA, pp 40–43.

- Couzy, W., and Deville, M.O. (1994). Iterative Solution Technique for Spectral-Element Pressure Operators at High Reynolds Numbers. *Proceedings of the Second European Computational Fluid Dynamics Conference*, (Ed. Wagner, S. et al.) Stuttgart, Germany, pp 613–618.

- Couzy, W., and Deville, M.O. (1994). Spectral-Element Preconditioners for the Uzawa Pressure Operator Applied to Incompressible Flows. *J. Sci. Comp.*, Vol. 9, No. 2, pp 107–122.

- Couzy, W,. and Deville, M.O. (1995). A Fast Schur Complement Method for the Spectral Element Discretization of the Incompressible Navier-Stokes Equations. *J. Comp. Phys.*, Vol. 116, pp 135–142.

- Pinelli, A., Couzy, W., Deville, M.O., and Benocci, C. (1995 ?). An Efficient Iterative Solution Method for the Chebyshev Collocation of Advection-Dominated Transport Problems. *SIAM J. Sci. Comp.*, Accepted for publication.

- Couzy, W. (1995). Iterative Solutions of the 3D Transient Navier-Stokes Equations on Parallel Computers. To appear in *Proceedings of the Third International Conference on High-Order and Spectral Methods*, Houston (TX), USA.