NTNU

PROJECT TMA4500

Application of nek5000 to dispersion simulations

Author:
Magnus AARSKAUG RUD

Supervisor:
Anne Kværnø

Research Group Name
IMF

October 2015

NTNU

Abstract

Faculty Name
IMF

Master Thesis

Application of nek5000 to dispersion simulations

by Magnus Aarskaug Rud

In this project some basic properties of the least squares method is analysed. Both finite element and spectral methods are discussed, and the implementation is mainly performed using spectral basis functions. The problems analysed in this project are diffusion convection reaction PDEs, both linear and non-linear. The results are compared to standard Galerkin method where both error and condition number is considered. The least squares method is also applied to Galerkin methods to gain stability with great results.

Contents

Notation

CONVENTION we let subscript h denote the discretized variables

Chapter 1

Numerical theory

1.1 Galerkin formulation

Throughout this thesis all numerical methods will be based on the Galerkin formulation. Let us consider a general bounary value problem (BVP)

$$\begin{aligned}\mathcal{L}u =& f \quad \text{in } \Omega \\ \mathcal{B}u =& g \quad \text{on } \partial\Omega.\end{aligned} \tag{1.1}$$

The domain Ω is a closed subspace of $\mathbb{R}^d$, $\mathcal{L} : X(\Omega) \to Y(\Omega)$ and $\mathcal{B} : X(\partial\Omega) \to B(\partial\Omega)$ are two linear operators, $f \in Y(\Omega)$ and $g \in B(\partial\Omega)$ are known functions and $u \in X(\Omega)$ is the wanted solution. The space $X(\Omega)$ will be denoted as the search space. A weak formulation can now be obtained by multiplying the first equation in 1.1 by a test function $v \in X^t(\Omega)$ and integrating over the domain Ω. By choosing $X^t(\Omega) = X(\Omega)$ the Galerkin formulation is obtained. For more examples and information on this subject the first chapters in [1] are recommended.

By the Lax-Milgram theorem it is known that a BVP is well-posed if the Operator $\mathcal{L}$ is both bounded and coersive.

Solving the BVP numerically involves choosing discrete subsets of X, X^t, Y, B. These will be denoted X_h, X_h^t, Y_h, B_h. The discrete subspaces can be chosen in a number of ways and the defining basis functions vary from one numerical method to another. In this thesis the spectral and finite element basis will be shortly stated and the spectral element basis will be viewed in more detail.

- a common BVP, weak formulation
- search and solution space, test and basis functions
- error estimation

1.2 Finite element method

Finite element method is one of the most widely used numerical methods applied on problems within construction, flow simulation and many other areas. It offers a precise mathematical foundation and due to the connectivity properties of the elements it guaranties a sparse system. The decomposition of the geometrical domain into a finite amount of elements chosen according to the problem wanted to solve, makes it possible to create general algorithms applicable to all kinds of geometries. For the full mathematical foundation of FEM it will be referred to [1], but some of the key propertie will be stated here in order to provide a thourough understanding of the spectral element method.

FEM provides an alorithm for solving any well-posed BVP 1.1 and the mathematical formulation is obtained by first finding the Galerkin formulation and choosing a discrete subset $X_h^p(\Omega) \subset X(\Omega)$ spanned by the finite element basis functions ϕ_i^p. p denotes the polynomial degree of the basis-functions, in 1D and for $p = 1$ the basis functions are defined as

$$\phi_i(x) = \begin{cases} \frac{x-x_i}{x_i-x_{i-1}} & \text{if } x_{i-1} \leq x \leq x_i, \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & \text{if } x_i \leq x \leq x_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $\text{supp}(\phi_i) = [x_{i-1}, x_{i+1}]$ and as a consequence of this $(\phi_i, \phi_j)_\Omega = 0$ if $|i-j| > 1$. These qualities is what gives rise to the resulting sparse linear system. By increasing the polynomial order the number of gridpoints used to define the polynomial will need to increase as well. This implies either reducing the distance between the gridpoints or increasing the support of each basis function. Both aproaches will reduce the sparsity of the final matrix. Another key aspect of FEM is the treatment of the domain Ω, on which a triangulization $\{\mathcal{T}_h\}$ is defined such that the original domain is divided into elements. By defining a reference element ($[-1, 1]$ in 1D) and a general mapping function, all the local contributions can be calculated by a generalized quadrature rule before being added to the global system of equations. This is a process tailored for parallelization, and can be generalized for a wide range of problems.

FEM is called a projection method since the solution $u_h \in X^h$ is a projection of the actual solution u of the BVP onto the discrete space X^h. Provided that the initial BVP is well-posed there exists to constants $M, \alpha > 0$ known as the bounded and coercivity constant such that the error of the solution can be reduced to a pure interpolation error. The result is known as Cea's lemma,

$$||u - u_h||_X \leq \frac{M}{\alpha}||u - I_h u||_X. \tag{1.2}$$

Where I_h is the projection operator.

Before this section ends it is important to understand the two ways to improve the error and the effects these two ways have on the algorithm. Assume the solution of the BVP to be infinitely smooth and let h denote the general size of the elements and p the order of the polynomial basis that defines X^h. Roughly speaking the error is given as $e = Ch^p$ with C being some constant. This is not a formal truth but rather a guideline as to how the error behaves, factors such as geometric complexity, condition-number,non-linear operators and the regularity of the solution will all provide slightly more complicated error estimates. However for a simpler BVP such as $u, f, g \in C^\infty(\Omega), \Omega = [-1, 1]^d, \mathcal{L} = -\Delta, \mathcal{B} = 1$ the error estimate is valid. performing a h-refinement will lead to an algebraic convergence, while the sparsity of the system is conserved and the total algorithm does not change in any other way than increasing the number of elements. Keeping h constant and increasing p will provide spectral convergence, but the sparsity will be reduced and all integrals solved will require quadrature rules of higher order. A more formal statement and numerical validation can be found in [2] chapter 2.6.

1.3 Spectral methods

Spectral methods (SM) share a some of the mathematical ideas as FEM, but are not as widely used in real life problems. There are many ways to apply SM, and in this thesis only the Galerkin version with numerical integration (known as G-NI) will be considered and will be referred to only as SM. For a full introduction to SM and its applications to BVP see [**?**]. SM can be reduced to a interpolation problem such as FEM, and are very interesting from a theoretical point of view due to its spectral convergence rate which allows you to obtain solutions of extremely high accuracy. The most important draw-back of SM are the difficulties with applications to complex geometries. Allthough the system of equations surging from a BVP can be constructed in an elegant way it is rarely sparse and often result in expensive calculations.

For a BVP in one dimension SM defines a set of basis functions $\{\psi_i\}_N$ which spans the whole domain Ω. The discrete space $X_h(\Omega)$ spanned by the basis functions involves all polynomials up to degree N. A function u is projected onto X_h by the relation

$$u_h(x) = \sum_{i=0}^{N} a_i \psi_i(x). \tag{1.3}$$

Where the coefficients a_i are called the expansion coefficients. There are many possible choices for the basis and the belonging coefficients, in this thesis and the algorithms used the functions ψ_i will be the Lagrange polynomials based on the Gauss-Lobatto-Legendre (GLL) nodes. The GLL-nodes are given as the solutions of the equation

$$(1-\xi^2)L_N'(\xi) = 0. \tag{1.4}$$

L_N being the Legendre polynomial of degree N, defined from the Sturm-Louville problem

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}L_n(x)\right] + n(n+1)L_n(x) = 0. \tag{1.5}$$

With equations 1.5 and 1.4 the basis functions ψ_j can be stated as

$$\psi_j = \prod_{i\neq j}^{N} \frac{x - x_i}{x_j - x_i}. \tag{1.6}$$

This should be taken a bit more thouroughly, Quadrature!

$\{x_i\}$ beeing the solutions to 1.4. Note that $\psi_j(x_i) = 0$ when $i \neq j$, the expansion coefficients in 1.3 are then chosen as $a_i = u_i := u(x_i)$ to minimize the projection error in $L^2(\Omega)$. expanding a basis such that the coefficients are simply the evaluation of the function in that particular point, is known as a nodal SM. Creating a basis for 2 and 3 dimensions is done simply by taking the cross product of the basis functions in each direction

$$\Psi_{ijk}(x,y,z) = \psi_i(x)\psi_j(y)\psi_k(z). \tag{1.7}$$

In order to clarify some of the concepts the SM approach will be applied on the Helmholtz equation

$$-\Delta u + \lambda u = f \quad \text{in } \Omega, \tag{1.8}$$

$$u = 0 \quad \text{on } \partial\Omega. \tag{1.9}$$

Ω will for this example be defined as the unit square $[-1,1]^2$. Let us start by defining the space $V = H_0^1(\Omega)$ and assuming $f \in L^2(\Omega)$. The weak formulation after applying the divergence theorem is the given as

Find $u \in V$ st.

$$\int_\Omega \nabla u \cdot \nabla v \partial\Omega + \lambda \int_\Omega uv\partial\Omega = \int_\Omega fv\partial\Omega \qquad \forall v \in V \tag{1.10}$$

In order to solve this using SM the discrete space $V_h \subset V$ is defined as span$\{\psi_i\}$ following the preceding definitions the discrete weak formulation is stated as Find $u_h \in V_h$ st.

$$\sum_i \left(u_i \int_\Omega \nabla\psi_i \cdot \nabla\psi_j \partial\Omega + u_i\lambda \int_\Omega \psi_i\psi_j\partial\Omega \right) = \int_\Omega f\psi_j\partial\Omega \qquad \forall\psi_j \in V_h. \tag{1.11}$$

The following step of this particular SM method is evaluating the integrals by using the GLL-quadrature rule, the resulting system of equations is then given as

$$\sum_i \left(u_i \sum_k \rho_k \nabla\psi_i(\mathbf{x}_k) \cdot \nabla\psi_j(\mathbf{x}_k) + u_i\lambda \sum_k \rho_k\psi_i(\mathbf{x}_k)\psi_j(\mathbf{x}_k) \right) \tag{1.12}$$

$$= \sum_k \rho_k f\psi_j(\mathbf{x}_k) \qquad \forall\psi_j(\mathbf{x}_k) \in V_h. \tag{1.13}$$

ρ_k is the quadrature weight for the kth node, and $\mathbf{x}_k$ is the vector containing the coordinates to the kth node. Note that all the indices $i, j, k = 1, \cdots, N_x N_y$. This can be written in a compact matrix form as

$$(A + \lambda M) u_h = \tilde{f}. \tag{1.14}$$

Where the elements in the matrices and vectors are given as

$$\begin{aligned}
A_{ij} &= \sum_k \rho_k \nabla\psi_i(\mathbf{x}_k) \cdot \nabla\psi_j(\mathbf{x}_k), \\
M_{ij} &= \sum_k \rho_k \psi_i(\mathbf{x}_k)\psi_j(\mathbf{x}_k) = \rho_i \delta_{ij}, \\
(u_h)_i &= u(\mathbf{x}_i), \\
\tilde{f}_j &= \sum_k \rho_k f(\mathbf{x}_k)\psi_j(\mathbf{x}_k) = \rho_j f(\mathbf{x}_j).
\end{aligned} \tag{1.15}$$

From these equations it is clear that the mass matrix M is diagonal and the rhs vector $\tilde{f}$ is easily calculated, while the stiffness matrix A is symmetric but full.

some figures or references to illustrate this

- quadrature rule
- the choice of nodes and basis functions
- sparsity for different operators
- the role of the jacobian
- cross product formulations

[?]

1.4 Spectral element method

In the early 1980's the the idea to combine FEM and SM came along in order to obtain the robustness and resulting sparse system of FEM combined with the spectral convergence rate provided by SM. The result was the Spectral element method. An example of Read articles... Several formulations was investigated and the development of super computers has played an important role in deciding the method applied today. The basic idea is to divide the domain of the BVP wanted to solve into elements as in FEM and then use spectral basis functions of higher degree with support only within one element.

In the previous subsection the power of spectral methods was illustrated on the unit square in two dimensions. But the limitations when it comes to more complex geometry rapidly affects the spectral convergence rate. Let $\hat{\Omega}$ be the reference element $[-1,1]^d$, the standard procedure when working on a deformed geometry Ω with SM is to first create a map $\mathcal{F}:\hat{\Omega}\to\Omega$. The jacobian is then given as the transposed tensor derivative of $\mathcal{F}$

$$\mathbf{J} = (\nabla\otimes\mathcal{F})^T = \begin{bmatrix} \frac{\partial\mathcal{F}_1}{\partial x} & \frac{\partial\mathcal{F}_1}{\partial y} \\ \frac{\partial\mathcal{F}_2}{\partial x} & \frac{\partial\mathcal{F}_2}{\partial y} \end{bmatrix}, \tag{1.16}$$

$$J = \det(\mathbf{J}). \tag{1.17}$$

This allows us to transform both derivatives and integrals to the reference domain, let $\boldsymbol{\xi} = [\xi,\eta]^T$ denote the axis in the reference domain corresponding to $\mathbf{x} = [x,y]^T$ in the deformed domain. The transformation is performed according to the following identities

$$\begin{aligned} d\mathbf{x} &= \mathbf{J}d\boldsymbol{\xi} \\ \int_\Omega f(\mathbf{x})d\mathbf{x} &= \int_{\hat{\Omega}} \hat{f}Jd\boldsymbol{\xi} \\ \nabla u &= \mathbf{J}^{-T}\hat{\nabla}\hat{u}. \end{aligned} \tag{1.18}$$

Here $\hat{u},\hat{f}$ are obtain by simply substituting $\mathbf{x}$ with $\mathcal{F}(\boldsymbol{\xi})$ and $\hat{\nabla}$ is the partial differential operator wrt. $\boldsymbol{\xi}$. The important thing to notice here is that whenever an integral is solved and a derivative is introduced the Jacobian appears in the equation. When applying the GLL-quadrature to solve the integrals equality is guaranteed iff the function integrated is of polynomial degree $2n-1$ or less, and the error gets bigger with increasing polynomial degree.

By dividing the domain up into smaller elements Ω_e the initial deformation of the full domain Ω can be reduced to have a very small effect on the each of the elements.

- READ LITERATURE ON THIS !!

Chapter 2

Fluid dynamics and problem specific mathematics

2.1 Navier-Stokes equation

The physics regarding fluids in motion are described mathematically by the Navier-Stokes equation. The equations can be derived in many ways, and it is referred to [4] for a complete explication. The general idea is to conserve momentum and mass in a control domain and by applying Reynolds transport theorem and the result is the following two equations reffered to as the N-S equations.

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla \cdot \tau \tag{2.1}$$

$$\nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.2}$$

These equations have been studied for centuries and different physical situations lead to different simplifications and sets of equations. Examples are the Euler equations, Stokes problem, darcy flow and the incompressible navier-stokes equations. Without doing the derivation of the equations a short discussion of each term in equation N-S will be given here in order to understand there physical origin and their mathematical attributes.

The term $\partial\mathbf{u}/\partial t$ is the time-derivative of the flow, for a steady state flow this term will be equal zero. The discretization of this term is often based on some implicit scheme.

The convective term $(\nabla \cdot \mathbf{u})\mathbf{u}$ describes the transport due to the flow itself on each of its components. This term is non-linear and does therefore require the equations to be solved by some iteration procedure such as Newton iterations.

The diffusive term $\nu\Delta\mathbf{u}$ describes the natural diffusion of the fluid. The effect of diffusion is determined by the viscosity of the fluid. Mathematically this term makes the equations numerically stable and it is therefore generally easier to solve the N-S equations for high-viscosity fluids.

The pressure gradient ∇p is the only term involving the pressure, ...DISCUST THE MEANING OF PRESSURE...in modern solution techniques it is common to solve eq. 2.5 in a decoupled manner.

The reynolds stress tensor $\tau = \cdots$ is the turbulent term of the N-S equation. In laminar flow this term can be neglected.

The second equation in 2.5 imposes a divergence free constraint on the solution $\mathbf{u}$ if the fluid is assumed to be incombressible. The density ρ will be a constant and the simplification follows trivially.

The coefficient ν is the inverse Reynolds number $1/Re$. The reynolds number can be understood in many different ways, but perhaps the most import role is that it describes the relation between the biggest length scales of the flow and the viscous length scales. Turbulent flows are characterized by high reynolds number. The huge span in length scales requires an extremely fine mesh if the equations 2.5 are to be solved exactly. Because a fine mesh implies a high computational cost a direct numerical solution (DNS) is not feasible for problems of a certain complexity.

- The time-derivative
- The convective term
- The diffusive term
- The pressure gradient

- The reynolds stress tensor
- The reynolds number
- The mass equation

2.2 Resolving the turbulent term

Depending on the wanted accuracy of your solution and the number of computational units at hand the turbulent term can be solved in different ways. A brief overview of the different approaches to the turbulent term are discussed in the following subsections

- Laminar flow
- RANS
- LES
- K-epsilon and K-omega
- DNS

2.3 Solution methods of incombressible N-S

A non-linear set of equations requires a non-trivial solution method, and when the domain of the problem can be anyting from a simple channel to a moving turbine there are many considerations that needs to be made. Allthough the equations have been known for over 200 years no one has been able to prove or disprove the well-posedness of the problem. FINAL ELEMENT FORMULATION WELL-POSED?? Some of the most common algorithms will be discussed in the following subsections

- coupled versus decoupled
- The Uzawa algorithm
- Pressure correction method
- fractional step method

2.3.1 Decoupling the pressure

A common way to approach equation 2.5 is to solve the pressure and velocity field seperatly. The mathematical reasoning for this approach is obtained by taking the divergence on both sides,

$$\nabla \cdot \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \cdot \nabla)\mathbf{u} = -\Delta p + \nu \Delta \tau \tag{2.3}$$

$$\frac{\partial (\nabla \cdot \mathbf{u})}{\partial t} + (\nabla \cdot \mathbf{u}) \cdot \nabla \mathbf{u} = -\Delta p + \nu \Delta \tau \tag{2.4}$$

$$\Delta p = \nu \Delta \tau \tag{2.5}$$

$$\tag{2.6}$$

Chapter 3

Application of nek5000 and the creation of grid

3.1 Turbulent flow solvers

There are many numerical solvers for turbulent flows available on the market. From large commercial softwares such as Fluent which runs as a black-box solver, to full open-source codes such as nek5000 and openFOAM. The solvers can vary both in the fundamental numerical method (FV,FD,FEM,SEM) the time-step method (Fractional Step, Poisson pressure, Uzawa) and the type of simulation (RANS,LES,DNS).

3.2 nek5000

Nek5000 is a turbulent flow solver developed mainly by Paul Fischer and has through the past 20 years had several contributours. It is an open-source code applicable to many different types of flow and it has been put a lot of effort into making the code as parallelizable as possible. With SEM as the numerical method applied it is possible to obtain extremely accurate results.

nek5000 has its own mesh-generator for generating simpler geometries and it is also implemented the possibility to integrate CUBIT mesh files. A common starting point for simulating turbulent flow is a geometry given by a CAD-file, without any functions to describe the boundaries. The possibility to make a mesh through a visual gui is provided through for instance ICEM. This mesh can then later be transformed to the inputfile required by nek5000.

So far nek5000 has supported three automatic routine for generating curved edges; circles in 2-D geometries, spherical shell elements and a general 2nd degree interpolation. Further manipulation of the element edges is left to the user to define manually for each particular problem. One of the objectives of this thesis is to make Nek5000 more user-friendly and create automatic routines to handle complex geometry.

In order to get a quick overview of the framework in Nek a short list of the main functions are stated in the list below

1. *Initialize*

   Reading mesh and parameters from user, generating GLL-points, and defining the values needed to perform the simulations. PRESSURE SOLV INIT ???

2. *solve*

   Solving for each timestep, depending on the problem solvers for flow,temperature and other scalars will be performed. The user is free to choose different solvers which will be discussed later.

3. *post-processing*

   Processing data and writing statistics and other user-specified parameters to file.

Nek provides a basic tool for generation of mesh. For more complex geometries this tool cannot compare with more visualized-based softwares such as ICEM from ANSYS which exports mesh to several numerical solvers such as Fluent and nastran. It is therefore very useful to have an automatic way of converting a mesh created in ICEM to the format required by nek5000. In order for nek to run optimally the elements should be as homogenous and as similar to the reference element as possible. It is therefore of great interest to be able to propagate curved geometries into the neighbour-elements in order to have a smooth as possible transition from a boundary with high curvature.

3.2.1 Incompressible N-S solvers in Nek

A Convection-Diffusion problem can be stated as

$$M\frac{du}{dt} = Au - Cu + Mf \tag{3.1}$$

Where M and A is the mass, and stiffness matrix, f is the loading function and C is the matrix corresponding to the convective term. The non-linearity is represented in the convective term since C is dependent of u. The time-derivative is discretized by a Backward difference (BDFk) scheme using solutions from the k previous steps to extrapolate the current value. In order to gain stability an implicit scheme is chosen and the resulting eguation is given as OIFS ??? ———- CHECK MAKEF IN NAVIER1.F

———————

$$\sum_{j=0}^{k} \frac{b_j}{\Delta t} M u^{n-j+1} = Au^{n+1} - Cu^{n+1} + Mf^{n+1} \tag{3.2}$$

By extrapolating the convective term from the k previously calculated steps the equation simplifies to

$$\frac{b_0}{\Delta t} M u^{n+1} \sum_{j=1}^{k} \frac{b_j}{\Delta t} M u^{n-j+1} = Au^{n+1} - \sum_{j=1}^{k} a_j C u^{n-j+1} + Mf^{n+1} \tag{3.3}$$

and finally by moving all the explicit terms to the rhs the equation left to solve is given as

$$(\frac{b_0}{\Delta t} M - A) u^{n+1} = -\sum_{j=1}^{k} (\frac{b_j}{\Delta t} M - a_j C) u^{n-j+1} + Mf^{n+1} \tag{3.4}$$

Notice from Section 1 that this is equivalent to the matrix formulation of the Helmholtz equation.

3.3 Gas dispersion in a simplified urban area

The problem investigated in this work is gas dispersion of neutral gas in a velocity field through four cubic blocks. Similar simulations have been done in CDP and Fluent which are compared to data from a wind-tunnel experiment performed by ALAN.

- Description of problem and domain
- mesh and fluent/CDP mesh

3.4 Creation of the mesh-convertion script

The routine xyzarc:

The gordon hall algorithm was already implemented as a function in Nek, with the gll-points,plynomial degree and some initial coordinates to the element. The algorithm creates a distribution of the internal gll-points in each element. If the element consists of linear edges the only necessary input are the vertices, but by specifying the points on edges and faces the algorithm creates a logical distribution of the internal GLL-points to a deformed element.

The curved edge is specified in the .rea file and the routine genxyz() processes the input of each edge. By specifying the radius and the circle center genxyz calls the routine xyzarc() which performs the following algorithm;

a, b will be the two endnodes of the edge c will be the midnode, s will be the arc length, θ will be the full angle of the circle sector, cc is the center coordinates. g will be the vector containing the GLL-points in $[-1, 1]$. r will be the radius.

```
l = a-b                        # vector between the corner nodes
c = (a+b)/2                    # midpoint location
h = c-cc                       # height of the framed triangle
$\theta$ = arctan(abs(l)/2abs(h))    # half the angle of the circle sector
s = r*$\theta$                       # arclength
g' = g*$\theta$                      # angles to the gll-points on the circle-sector
#---------- Finding the intersecting points ----------#
#---- x on the line l, and extend x-cc to the arc ----#
for k in range(lx1):          # For the number of nodes in one direction
   $\alpha$ = h*tan(g'[k])          # Offset from the midpoint on l
   x = c-$\alpha$*l/abs(l)          # Actual coordinate on l
   m = x-cc                    # hypothenus of the imposed triangle
   edge(k) = cc+r*m/abs(m)     # final coordinate on the arc
```

These lines creates the wanted egde curved as a circle sector corresponding to the radius and circle center given. The remaining operation is to call the gordon hall algorithm and create the internal GLL-points defined by the edges provided.

- reading radius, and center
- finding the angular offset to match the gll-points
- finding the points on the linear edge
- projecting them on the surface

- initial script
- changes and modifications
- performance testing
- pitfalls

Chapter 4

Results

4.1 Drag and lift on a cylinder

The effect of the implemented algorithm in Nek5000 explained in Chapter 3 is illustrated by solving a laminar flow test problem. The solution is compared with previously benchmark computations performed by a number of contributors [5]. The test problem considered is steady flow with Re=20 in a rectangular channel past a cylinder. The drag and lift coefficients on the cylinder are calculated and compared to a pair of reference values. The results can be found in table 4.1. As the results clearly show the treatment of the geometry is crucial, both coefficients are computed with significantly better accuracy. The polynomial degree used in the calculations with Nek5000 was chosen as $p = 11$ in all directions. The explicit number of cells is therefore $2070 \cdot 11^3 = 2755170$, approximately 88% of the number of cells used for the benchmark simulations. Compared with the results from the other softwares applied in [5] Nek5000 performs just

# of Cells	Software	c_D	c_L	%**Err** c_D	%**Err** c_L
2070	Nek5000 (mid)	6.18349	0.008939	0.030	4.19
2070	Nek5000 (arc)	6.18498	0.009413	0.006	0.13
3145728	CFX	6.18287	0.009387	0.04	0.15
3145728	OF	6.18931	0.00973	0.06	3.5
3145728	FEATFLOW	6.18465	0.009397	0.01	0.05

TABLE 4.1: Results for the drag and lift coefficients with refences values $c_D = 6.18533$ and $c_L = 0.009401$.

as well or better. It should be mentioned that the division of the grid is done differently for Nek5000 so the comparison is not as direct as it may seem from the table. add more info on the mesh??

Bibliography

[1] Alfio Quarteroni. *Numerical Models for Differential Problems, 2. edition*. Springer, 2014.

[2] George Karniadakis and Spencer Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

[3] Canuto C. Canuto, M.Y Hussaini, and A. Quarteroni... [et al.]. *Spectral methods : evolution to complex geometries and applications to fluid dynamics*. Scientific computation. Springer, Berlin, 2007. ISBN 978-3-540-30727-3. URL http://opac.inria.fr/record=b1127560.

[4] Frank M. White. *Viscous fluid flow*. McGraw-Hill series in mechanical engineering. McGraw-Hill, New York, 1991. ISBN 0-07-069712-4. URL http://opac.inria.fr/record=b1096847.

[5] S. Turek and M. Schäfer. Recent benchmark computations of laminar flow around a cylinder, 1996.