Embedded Boundary Method for Navier-Stokes Equations

OLE KRISTIAN OLSEN

Master’s Degree Project
Stockholm, Sweden 2005

TRITA-NA-E05048
Embedded Boundary Method for Navier-Stokes Equations

OLE KRISTIAN OLSEN

TRITA-NA-E05048

Master’s Thesis in Numerical Analysis (20 credits) at the Scientific Computing International Master Program, Royal Institute of Technology year 2005
Supervisor at Nada was Björn Sjögreen
Examiner was Jesper Oppelstrup
“Do not worry too much about your difficulties in mathematics, I can assure you that mine are still greater” - Albert Einstein

Abstract

The unsteady Navier-Stokes equations describing compressible flow in 2D are solved by the embedded boundary method. Boundary conditions at the embedded boundaries are imposed by special interpolation methods, which maintain high order accuracy.

Numerical experiments with the method are presented for two test problems. The first is a supersonic flow around an ellipse, while the other is a supersonic flow around three disks.
I would like to thank Björn Sjögreen, my supervisor, who gave me the opportunity to do this thesis work here at Nada. I really appreciate the help and guidance provided by Marco Kupiainen who carefully read the thesis, correcting errors, pointing out weak points, and suggesting additional topics. A special thanks to Katarina Gustavsson for always being there with good references and books on my topic.
Contents

1 Introduction .............................................. 1
  1.1 Background and aim .................................. 1
  1.2 Outline of the Thesis ............................... 3

2 Equations .................................................. 4
  2.1 Compressible Navier-Stokes equations ............... 4
    2.1.1 Mathematical properties ...................... 5
    2.1.2 Boundary condition ............................ 6
  2.2 Approximation of the Navier-Stokes equations ...... 6
    2.2.1 Discretization of viscous terms ................ 7
    2.2.2 Approximation near the outer boundary .......... 8
    2.2.3 Time stepping discretization ................... 8

3 Embedded boundary technique ............................ 9
  3.1 Introduction ......................................... 9
  3.2 The no-slip condition ............................... 10
  3.3 Extrapolation boundary condition .................... 11
  3.4 Approximation of the viscous terms at the embedded boundary ... 12

4 Results ..................................................... 16
  4.1 Flow past a cylinder .................................. 16
    4.1.1 Boundary layer ................................ 18
    4.1.2 Skin friction .................................. 19
  4.2 Flow past disks ...................................... 20
  4.3 Parallel performance .................................. 23

5 Conclusion .................................................. 25

References ................................................... 26
Chapter 1

Introduction

1.1 Background and aim

Discretization of partial differential equations with the embedded boundary method date all the way back to the first order technique by Weller and Shortley [7] in 1939. In more recent time, several embedded boundary methods have been presented for various types of partial differential equations. For example, Ditkowski, Dridi and Hesthaven [1] used a staggered grid for solving Maxwell’s equations on a Cartesian grid. Kreiss and Petersson [3] have used a second order accurate embedded boundary method for the wave equation with Dirichlet data. For the compressible gas dynamics described by the Euler equations, Sjögreen and Petersson [4] developed a Cartesian embedded boundary method for hyperbolic conservation laws.

We will with this paper extend the method, which has successfully solved the wave equation, Maxwell equations and the Euler equations, and develop a numerical method for the Navier-Stokes equations, with a description of what is special about the method. Solving the full system of Navier-Stokes equations is one of the ultimate goals of a numerical flow simulation, as it describes most of the properties of a continuous flow and is therefore a more realistic model than the Euler equations, given here by (1.1)

\[ u_t + f(u)_x + g(u)_y = 0. \] (1.1)

The Navier-Stokes equations include effects of fluid viscosity and heat conduction. The stress terms involve second-order derivatives that would make the system parabolic rather than hyperbolic, and cause the solution to be smooth. The discontinuous shock waves in the Euler equations are replaced by thin regions over which the solution is rapidly varying.

We consider the Navier-Stokes equations in a two-dimensional domain \( \Omega \) with boundary \( \Gamma \), here written in abbreviated form

\[ u_t + f(u)_x + g(u)_y = f_v(u, u_x, u_y)_x + g_v(u, u_x, u_y)_y, \ (x, y) \in \Omega, \ t > 0, \] (1.2)
Chapter 1. Introduction

![Computational domain](image)

Figure 1.1. Computational domain

Together with initial data and boundary conditions on the outer boundary:

\[ u(x,y,0) = u_0(x,y), \quad (x,y) \in \Omega, \quad (1.3) \]
\[ u(0,y,t) = f_0(y,t), \quad u(l_x,y,t) = f_1(y,t), \quad 0 \leq y \leq l_y, \quad t > 0, \quad (1.4) \]
\[ u(x,0,t) = f_2(x,t), \quad u(x,l_y,t) = f_3(x,t), \quad 0 \leq x \leq l_x, \quad t > 0. \quad (1.5) \]

The problem above will be solved numerically with a Cartesian embedded boundary approach. In the embedded boundary method, objects of arbitrary shape are cut out, hence the computational domain \( \Omega \) will be the difference of an outer rectangle with boundary \( \Gamma_o \) and one or more internal objects with boundary \( \Gamma_i \), see Figure 1.1. \( \Omega \) is covered by a regular Cartesian grid with points \( x_{i,j} = (ih,jh) \), where \( i,j \) are the indices in \( x \)- and \( y \)-direction and \( h \) is the step size. On the boundary of the internal objects \( \Gamma_i \), we impose

\[ u(x,y,t) = f_4(x,y,t), \quad (x,y) \in \Gamma_i, \quad t > 0. \quad (1.6) \]

Many methods of this type suffer from poor accuracy at the internal boundary. In our approach, boundary conditions are imposed by special interpolation methods, which maintain high order accuracy.

Typical for the Navier-Stokes equations is the presence of boundary layers, caused by the viscosity and the no-slip condition. The flow is directed along the boundary. In standard boundary fitted grid methods the grid is aligned with the boundary, making it very suitable for such flow. With the embedded boundary method, the grid is not aligned with the flow, and the accuracy obtained with the embedded boundary method needs to be investigated.
Chapter 1. Introduction

1.2 Outline of the Thesis

The remainder of this thesis is organized as follows. The governing equations of the problem, mathematical properties and numerical approximation are discussed in Chapter 2. In Chapter 3 we look at how the embedded boundaries are treated. The results from the numerical experiments are provided in Chapter 4. And finally we have a conclusion in Chapter 5.
Chapter 2

Equations

2.1 Compressible Navier-Stokes equations

The system of Navier-Stokes equations, supplemented by empirical laws for the dependence of viscosity and thermal conductivity with other flow variables and by a constitutive law defining the nature of the fluid, are commonly accepted to describe fluid and gas flow phenomena.

We consider the unsteady Navier-Stokes equations for compressible fluid flow in two space dimensions. The equations are, c.f. [6]

\[
\begin{pmatrix}
\rho \\
\rho u \\
\rho v \\
e
\end{pmatrix}_t + \begin{pmatrix}
\rho u \\
\rho u^2 + p \\
\rho uv \\
u(e+p)
\end{pmatrix}_x + \begin{pmatrix}
\rho v \\
\rho v^2 + p \\
\rho uv \\
v(e+p)
\end{pmatrix}_y = \left(\frac{\alpha(T)}{Re} F\right)_x + \left(\frac{\alpha(T)}{Re} G\right)_y,
\]

(2.1)

where

\[
F = \begin{pmatrix}
0 \\
\frac{4}{3} u_x - \frac{2}{3} v_y \\
\frac{1}{3} v u_x - \frac{2}{3} v u_y + v u_y + v v_x + \frac{\gamma}{\gamma-1} \left(\frac{E}{\rho}\right)_x
\end{pmatrix}
\]

(2.2)

and

\[
G = \begin{pmatrix}
0 \\
u_y + v_x \\
\frac{4}{3} v y - \frac{2}{3} u_x \\
\frac{1}{3} v v_y - \frac{2}{3} v u_x + u u_y + u v_x + \frac{\gamma}{\gamma-1} \left(\frac{E}{\rho}\right)_y
\end{pmatrix}
\]

(2.3)

In (2.1) \( \rho \) is the density, \( u \) and \( v \) are the velocities in the \( x \)- and \( y \)-direction respectively, and \( e \) is the total energy, and in (2.1)-(2.3) the following parameters occur: \( Re \).
Chapter 2. Equations

is the Reynolds number, $Pr$ is the Prandtl number and $\gamma$ is the isentropic exponent, the ratio of the gas specific heats, $\gamma = 1.4$ for air. The preceding set of equations are written in non-dimensional form. Distances have been non-dimensionalized by a reference length $L$; velocities by a reference velocity $U_{ref} = c_\infty$, where $c_\infty = \sqrt{\frac{\gamma p_\infty}{\rho_\infty}}$ is the freestream speed of sound. Density and pressure are scaled by freestream values given by

$$\rho^* = \frac{\rho}{\rho_\infty}, \quad p^* = \frac{p}{\rho_\infty U_{ref}^2} \quad (2.4)$$

where $*$ signifies a non-dimensional variable. The formulas for $Re$ and $Pr$ are given in equation (2.5) and (2.6), respectively.

$$Re = \frac{\rho_\infty u_\infty L}{\mu_\infty}, \quad (2.5)$$

where $\rho_\infty$ is the density in freestream, $u_\infty$ the velocity in freestream, $\mu_\infty$ the viscosity in freestream, and $L$ is the characteristic length.

$$Pr = \frac{\mu_\infty C_p}{\lambda_\infty}, \quad (2.6)$$

where $C_p$ is the specific heat and $\lambda_\infty$ is the thermal conductivity parameter.

The function $\alpha(T)$ describes how the viscosity depends on temperature, where the temperature is given by

$$T = b \frac{p}{\rho}, \quad (2.7)$$

where $b$ is an additional parameter determined from the conditions at the freestream, i.e. $b = T_\infty \rho_\infty / p_\infty$. In our numerical experiments we start simple with $\alpha = 1$, but $\alpha(T)$ can in further development be constructed from Sutherland’s law. We use the ideal gas law

$$p = (\gamma - 1)(e - \frac{1}{2} \rho (u^2 + v^2)) \quad (2.8)$$

to calculate the pressure.

2.1.1 Mathematical properties

The Navier-Stokes equations differ from the Euler equations, solved in [4], in several ways. The presence of viscosity and heat conduction transform the conservation laws of momentum and energy into second-order partial differential equations, which make them parabolic. The continuity equation, on the other hand is hyperbolic since it is a first-order differential equation. The full system of the Navier-Stokes equations is of incomplete parabolic type.
2.1.2 Boundary condition

Physical experience has to be used in order to determine the boundary condition to impose along solid walls. According to [2], all known experiments indicate that relative velocity between the fluid and the solid wall is zero, within the framework of continuum mechanics. This is called the no-slip condition. At a solid wall boundary, \( u = v = 0 \) are therefore taken as the boundary condition for the velocity. We also need a boundary condition for the temperature, and it is given as adiabatic

\[
\frac{\partial T}{\partial n} = 0
\]

on the wall, where \( \frac{\partial T}{\partial n} \) denotes the wall normal derivative.

2.2 Approximation of the Navier-Stokes equations

We can write equation (2.1) in the following form

\[
\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x + \mathbf{g}(\mathbf{u})_y = f_v(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_y)_x + g_v(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_y)_y,
\]

(2.9)

where \( \mathbf{u} = (\rho, \rho u, \rho v, e) \), and \( f_v \) and \( g_v \) are the viscous terms. As we wish to solve (2.9) numerically, we first need to discretize the computational domain \( \Omega \). This is carried out on an uniform Cartesian grid with grid points \( (x_i, y_j), x_i = (i-1)\Delta x, i = 1, 2, ..., M, y_j = (j-1)\Delta y, j = 1, 2, ..., N. \Delta x = l_x/(M-1) \) and \( \Delta y = l_y/(N-1) \), and denotes the grid spacing in the x- and y-direction, respectively.

The numerical approximation of the Navier-Stokes equations for high-Reynolds number flows relies largely on the methods developed for inviscid flows. Most of the schemes applied to the Euler equations can be used for the Navier-Stokes equations by discretizing the viscous and heat conduction terms, using central differences.

The approximation of the convective terms is given by, see [4],

\[
f(\mathbf{u}(x_i, y_j, t_n))_x + g(\mathbf{u}(x_i, y_j, t_n))_y = \frac{a_{i+1/2,j} - a_{i-1/2,j}}{\Delta x} + \frac{b_{i+1/2,j} - b_{i-1/2,j}}{\Delta y},
\]

(2.10)

where \( a_{i+1/2,j} \) and \( b_{i+1/2,j} \) are discrete flux functions approximated with a five point second order accurate MUSCL (monotonic upstream-centered scheme for conservation laws) scheme, give by

\[
a_{i+1/2,j} = h(\mathbf{u}_{i+1/2,j}^R, \mathbf{u}_{i+1/2,j}^L).
\]

(2.11)

Here \( h(\mathbf{u}, \mathbf{v}) \) is the numerical flux function of the first order upwind method, and \( \mathbf{u}_{i+1/2,j}^R \) and \( \mathbf{u}_{i+1/2,j}^L \) are the right and left limits of the piecewise linear reconstruction of the solution.

We can now write the semi-discrete approximation of the system of equations (2.9) as

\[
\frac{d\mathbf{u}_{i,j}(t)}{dt} = -L(\mathbf{u}_{i,j}(t)) + W(\mathbf{u}_{i,j}(t)),
\]

(2.12)

where \( L(\mathbf{u}_{i,j}(t)) \) is the approximation of the convective terms (2.10), and \( W(\mathbf{u}_{i,j}(t)) \) is the approximation of the viscous terms.
2.2.1 Discretization of viscous terms

The viscous terms are discretized with the Finite Difference Method. This method is based on the properties of Taylor expansions and of the definition of derivatives. On a Cartesian grid, as we have, it is fairly straightforward to apply. Discretization near the embedded boundary will be more carefully described in Section 3, as it needs special consideration. The $u_{xx}, v_{xx}, u_{yy}$ and $v_{yy}$ in the momentum equations are approximated directly with the second order compact central difference formulas for the inner points

\[
(u_{xx})_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} - \frac{\Delta x^2}{12} \frac{\partial^4 u}{\partial x^4} \tag{2.13}
\]

with the forward difference formulas (second order accuracy)

\[
(u_{xx})_{i,j} = \frac{2u_{i,j} - 5u_{i+1,j} + 4u_{i+2,j} - u_{i+3,j}}{\Delta x^2} + \frac{11}{12} \Delta x^2 \frac{\partial^4 u}{\partial x^4} \tag{2.14}
\]

and backward difference formulas (second order accuracy)

\[
(u_{xx})_{i,j} = \frac{2u_{i,j} - 5u_{i-1,j} + 4u_{i-2,j} - u_{i-3,j}}{\Delta x^2} - \frac{11}{12} \Delta x^2 \frac{\partial^4 u}{\partial x^4} \tag{2.15}
\]

for outer boundary points. Similar equations, (2.13) - (2.15), are also used to calculate the second-order derivatives of the temperature, $T_{xx}$ and $T_{yy}$, in the energy equation, $v_{xx}, v_{yy}$ and $u_{yy}$.

The mixed derivative, e.g. $(vu_y)_x$, is approximated in the following manner. First we compute $u_y$ with the second order central difference scheme

\[
(u_y)_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} - \frac{\Delta y^2}{6} u_{yyy}, \tag{2.16}
\]
Then we multiply \((u_y)_{i,j}\) with \(v\) and finally we take the central difference approximation of this results with respect to \(x\),

\[
((vu_y)_x)_{i,j} = \frac{(vu_y)_{i,j+1} - (vu_y)_{i,j-1}}{2\Delta x}.
\] (2.17)

From Figure 2.1 we see that this approximation becomes a nine point stencil. All the different variations of the mixed derivatives \(((uv)_x, (uv)_y, \text{etc})\) are approximated in similar matter as described above, except on the outer boundaries. Here we use the second order forward- and backward formulas

\[
(u_x)_{i,j} = \frac{-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}}{2\Delta x} + \frac{\Delta x^2}{3}u_{xxx}
\] (2.18)

\[
(u_x)_{i,j} = \frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2\Delta x} + \frac{\Delta x^2}{3}u_{xxx},
\] (2.19)

with similar formulas for the \(y\) spatial direction.

### 2.2.2 Approximation near the outer boundary

The left boundary on \(\Gamma_o\) is an inflow boundary. The other boundaries are outflow boundaries. At the inflow, values are given for all the dependent variables in the Navier-Stokes equations. At outflow boundaries we extrapolate values from the interior of the domain to the boundary, as described in [4].

### 2.2.3 Time stepping discretization

The semi-discrete system (2.12) is integrated in time with a two stage, second order accurate, Runge-Kutta method. We use the same approach as in [4], adapted to the added viscous terms in the Navier-Stokes equations. For the time-step method chosen the CFL condition for the Euler equations is defined as

\[
(|u| + |v| + c)\frac{\Delta t}{\Delta x} < 0.5.
\] (2.20)

When the Reynolds number gets “small” and we get more contribution for the viscous term, the equations transforms to parabolic type. The CFL condition now places more severe constraints on the time stepping. For the Navier-Stokes equations we have the relation between \(\Delta t\) and \(\Delta x\) as

\[
\lambda \frac{\Delta t}{\Delta x^2} < 0.5,
\] (2.21)

where

\[
\lambda = \frac{2}{Re}\Max\left(\frac{4}{3} \frac{\gamma}{Pr} \frac{\alpha(T)}{\rho}\right)
\] (2.22)

and is taken from the eigenvalues of the Jacobi matrix of the Navier-Stokes equations.
Chapter 3

Embedded boundary technique

3.1 Introduction

In the embedded boundary method the internal boundary $\Gamma_i$ is allowed to intersect the grid in an arbitrary fashion, as long as it is resolved on the grid, see Figure 1.1. In this section we will describe the technique on how to impose Dirichlet and extrapolated boundary conditions on this embedded boundary.

Let $u_{i,j}(t)$ denote the numerical solution at grid point $(i,j)$. In order to update all the grid points inside the domain $\Omega$, with the internal approximation scheme, we use ghost points just outside the domain $\Omega$, see Figure 3.1 for a close up. We must, therefore, at each time step define new values of $u_{i,j}(t)$ at the ghost points. This is accomplished with the following procedure. Consider the ghost point from Figure 3.1 with index $(i,j)$. $U_I, U_{II}$ and $U_{III}$ are the intersections between the outward normal going through $x_{i,j}$ and the grid lines $y = y_{i,j+1}$, $y = y_{i,j+2}$ and $y = y_{i,j+3}$, respectively. Values for these points are obtained from linear interpolation along the grid lines $y = y_{i,j+1}$, $y = y_{i,j+2}$ and $y = y_{i,j+3}$ between values from the nearest neighboring grid points to left and right. The ghost point at $(i,j)$ in Figure 3.1 gives us the following interpolation formulas.

\begin{align*}
    u_I &= w_1 u_{i,j+1} + (1 - w_1) u_{i+1,j+1}, \\
    u_{II} &= w_2 u_{i,j+2} + (1 - w_2) u_{i+1,j+2}, \\
    u_{III} &= w_3 u_{i+1,j+3} + (1 - w_3) u_{i+2,j+3},
\end{align*}

where $0 \leq w_k \leq 1$, $k = 1, 2, 3$, depend on where the normal intersects the horizontal grid lines. The computation of the weights, $w_k$, will not take up much of the overall execution time since they are computed once and stored in the program. The horizontal interpolations in Figure 3.1 holds when the angle $\theta$ between the normal and the $x$-axis is $\pi/4 \leq \theta \leq \pi/2$. With this angle on $\theta$ and a positive $y$-component of the normal, it will always intersect the grid line $y = y_{j+1}$ between $x_i$ and $x_{i+1}$.

The intersection with grid line $y = y_{j+2}$ can have two different cases, between $x_i$ and $x_{i+1}$ or between $x_{i+1}$ and $x_{i+2}$. For the grid line $y = y_{j+3}$, we can have three
different cases. When $\theta$ is between 0 and $\pi/4$, the horizontal interpolations are replaced by corresponding interpolations in the vertical direction. For other directions of the normal, the interpolations are simply obtained by reflections and/or as mirror images of Figure 3.1.

3.2 The no-slip condition

For the Navier-Stokes equations we want the velocity at solid walls to be zero. Consider imposing the Dirichlet condition $u(x^\Gamma, y^\Gamma) = d(x^\Gamma, y^\Gamma)$ on $\Gamma_i$. In our case $d(x^\Gamma, y^\Gamma)$, the given boundary value, is zero. First we calculate $u_I, u_{II}$ and $u_{III}$ for velocity in the x direction and $v_I, v_{II}$ and $v_{III}$ for velocity in y direction, with the method described in the section above. Next we define values $u_{b1}, u_{b2}, v_{b1}$ and $v_{b2}$ at points $b + \Delta$ and $b + 2\Delta$ from the ghost point, respectively, by interpolation. From Figure 3.1 we see that $b$ is the distance between the ghost point and the boundary $\Gamma_i$, and $\Delta$ the distance between the ghost point and the grid line $y = y_{j+1}$. We now have the formulas for the x-component velocity

$$u_{b1} = (b/\Delta)u_{II} + (1 - b/\Delta)u_I, \quad u_{b2} = (b/\Delta)u_{III} + (1 - b/\Delta)u_{II}$$

(3.4)

and for the y-component velocity

$$v_{b1} = (b/\Delta)v_{II} + (1 - b/\Delta)v_I, \quad v_{b2} = (b/\Delta)v_{III} + (1 - b/\Delta)v_{II}.$$  

(3.5)
Chapter 3. Embedded boundary technique

We now use this values to define a limited boundary slope, the slope at the point \( b + \Delta \) away from from the ghost point, for both velocity components,

\[
s_{\Gamma} = S_{mm}(u_{b1} - d(x_{\Gamma}^1, y_{\Gamma}^1), u_{b2} - u_{b1}).
\]

(3.6)

\( S_{mm}(x, y) \) is the min-mod limiter, defined by

\[
S_{mm}(x, y) = \begin{cases} x & \text{if } |a| < |b| \text{ and } ab > 0, \\ y & \text{if } |b| < |a| \text{ and } ab > 0, \\ 0 & \text{if } ab \leq 0. \end{cases}
\]

(3.7)

The min-mod limiter returns the smallest of \( x \) and \( y \) in modulus if they have the same sign, else it returns zero. If it returns zero, the two slopes have different sign, and we then have a local maximum or minimum.

Finally, using the limited boundary slope, we can define the value of the ghost point by extrapolating the boundary value with

\[
u_{i,j} = d(x_{i,j}^\Gamma, y_{i,j}^\Gamma) - \frac{b}{\Delta} s_{\Gamma}.
\]

(3.8)

### 3.3 Extrapolation boundary condition

For an “adiabatic” wall, the heat flux is zero, so the wall-normal derivative of the temperature vanishes, \( \frac{\partial T}{\partial n} = 0 \). From equation (2.7), the temperature is given as the pressure divided by the density. For these two variables there are no physical boundary values to impose. We use extrapolation to update the ghost points for these values. The extrapolation can be done without using any information about the boundary location since there is no condition to impose. The formula to extrapolate the pressure along the normal is

\[
p_{i,j} = p_I - S(p_{III} - p_{II}, p_{II} - p_I).
\]

(3.9)

We first use linear interpolation to find \( p_I, p_{II} \) and \( p_{III} \) with the method described in section 3.1, then extrapolate the pressure \( p \) with equation (3.9). We can now determine the value for the density, that will give \( \frac{\partial T}{\partial n} = 0 \) on the embedded boundary, from

\[
\frac{\partial T}{\partial n} = 0 \Rightarrow \frac{\partial (\rho \frac{\partial T}{\partial n})}{\partial n} = 0 \Rightarrow \frac{p_{i,j}}{\rho_{i,j}} = \frac{p_I}{\rho_I} \Rightarrow \rho_{i,j} = \frac{p_{i,j} \rho_I}{p_I}.
\]

(3.10)

Finally, we use the values of the pressure and the density to calculate the energy in the ghost points with

\[
e_{i,j} = \frac{p_{i,j}}{\gamma - 1} + \frac{1}{2} \rho_{i,j}(u_{i,j}^2 + v_{i,j}^2).
\]

(3.11)

Another possibility is to extrapolate the density and find the pressure from \( \frac{\partial T}{\partial n} = 0 \).

We chose to extrapolate the pressure because for a non-adiabatic wall there is a boundary layer in density, but there is no boundary layer in the pressure.
Chapter 3. Embedded boundary technique

3.4 Approximation of the viscous terms at the embedded boundary

So far in this chapter, we have considered how to find the values of the velocities, density, pressure and energy in the ghost points. Now we will see how these ghost points can be used to approximate the viscous terms around the embedded boundaries.

![Figure 3.2.](image)

We know that we always have one ghost point inside the embedded boundary that can be used in our approximation with the finite difference method. Since the approximation of the second derivative \( u_{xx}, v_{yy}, \text{ etc.} \) uses a three point stencil, we can use the method described in 2.2.1 to compute the values for this derivatives close to the embedded boundary, see Figure 3.2. If we now instead imagine we have the situation in Figure 3.3, and we want to approximate \((vu_y)_x\). We can, clearly,

![Figure 3.3.](image)

not use the nine point stencil from Figure 2.1 because we don’t have any values for \( u \) and \( v \) at the point \( i + 1, j + 1 \). To solve this problem, we first approximate the first derivative at all the ghost points and then use the central difference operator

\[
 u_{xx} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}
\]
to find the mixed derivatives. The ghost points are approximated to second order accuracy by the one sided difference formulas in equations (2.18) and (2.19) for x- and y spatial directions.

To be sure that we only use valid grid points, we need some kind of test. To accomplish this we choose \( u = \sin x \cos y \) and \( v = \cos y \), and approximate the derivatives of this function with the help of our program. Now that we have both the true- and approximated values of the derivatives, we can easily find out how accurate the approximation is. We can for example look at the errors of \( u_{xx} \) and \( u_{xy} \), calculated as described earlier in this section on the domain in figure 3.4. The analytical function of the derivatives is \( u_{xx} = -\sin x \cos y \) and \( u_{xy} = -\cos x \sin y \). Figure 3.5 shows the error of \( u_{xx} \) on a 305\( \times \)305 grid. Since we calculated \( u_{xx} \) directly with the three point stencil over the whole domain, we do not get any extra error around the cylinder. We see that the error at the outer boundary dominates. It is due to the fact that the forward- and backward difference formulas have a higher truncation error than the difference scheme for the interior points. When the number of grid points are increased the error decreases, see Figure 3.6. The error is proportional to \( h^2 \), both on the outer boundary and in the interior points. Thus the approximation of the second order derivatives are of second order accuracy. We see in Figure 3.7 that the error in \( u_{xy} \) is highest around the cylinder. From the situation in Figure 2.1 we know that truncation error, for \( u_y \), on each side of point \( i,j \) are different. When we now use the approximation of \( u_y \) to find the mixed derivatives in \( i,j \), the accuracy is no longer second order. By determining the max error on different grids (i.e. 305\( \times \)305 and 605\( \times \)605), we found the accuracy to be just between first and second order.
Figure 3.5. Error of $u_{xx}$ on a 305×305 grid

Figure 3.6. Error of $u_{xx}$ on a 605×605 grid
Chapter 3. Embedded boundary technique

Figure 3.7. Error of $u_{xy}$ on a $305 \times 305$ grid
Chapter 4

Results

We have written a computer code, implementing the methods described in the preceding chapter, to simulate a compressible flow. In this section we will look at the numerical results obtained by solving equation (2.1) with this computer code.

4.1 Flow past a cylinder

In our first problem, we compute a supersonic two dimensional compressible flow past a cylinder of radius 0.5. The compressible flow around a cylinder is a complex case, as it contains shocks, separation and compressibility effects in interaction with boundary layers. In the free stream, the Mach number is 3, \( \gamma = 1.4 \) (for air) and \( Re = 100 \). The computational domain is \([-2,2] \times [-2,2]\) with 305\times305 grid points. The results are reported at time \( t = 10 \). A big challenge to get a converging solution, was to find good initial data to the problem. Badly chosen initial data can cause the solution to diverge. What we did was to set the value of the density behind the cylinder lower than what we have in the freestream, 1.4, in front of the cylinder. Figure 4.1 shows the density contours for the embedded boundary method. We see that a shock arises in front of the cylinder and that the flow separates symmetrically to both sides. In front of the cylinder we also get the highest density. Figure 4.2, shows the velocity around the cylinder.

To determine the accuracy we ran on three different resolution (100\times100, 200\times200 and 400\times400) and compared the solution with the L2-norm.

\[
\frac{\|u_{200} - u_{100}\|_{L^2}}{\|u_{400} - u_{200}\|_{L^2}} = 3.213
\]  

(4.1)

From equation (4.1) we see that order of accuracy is between one and two. We can not expect to get second order, because as we remember the mixed derivatives don’t hold second order accuracy either.
Chapter 4. Results

Figure 4.1. Density contour for supersonic flow past a cylinder

Figure 4.2. Quiver plot of the velocity around the cylinder
Chapter 4. Results

4.1.1 Boundary layer

When gas flows over a surface, as shown in Figure 4.3, the viscous effects are confined to a thin layer just above the surface, known as the boundary layer.

![Diagram of boundary layer](image)

Figure 4.3. Boundary layer

Figure 4.4 illustrates the boundary layer from our numerical results. We see that lower Reynolds numbers give thicker boundary layers, which is consistent with the theory, see e.g. [5]. We also see that the separation of the boundary layer from the solid surface occurs later for lower Reynolds numbers.

![Contour plots showing the boundary layer](image)

Figure 4.4. Contour plots showing the boundary layer
4.1.2 Skin friction

The boundary layer exerts a drag force on the surface it flows over. We can think of drag as aerodynamic friction. One of the sources of drag is the skin friction between the molecules of the gas and the solid surface of the cylinder. The magnitude of the skin friction depends on the viscosity and the velocity of the gas. We can calculate the skin friction coefficient by the following formula.

\[ C_f = \frac{1}{Re} \frac{\partial V}{\partial n} \frac{1}{2 \rho \infty u_\infty^2}, \]  

(4.2)

where \( V \) is the velocity tangential to the wall, and is given as

\[ V = -n_2 u + n_1 v. \]  

(4.3)

Figure 4.5 shows the skin friction coefficient along the surface of the cylinder as a function of the x-coordinate.
Chapter 4. Results

4.2 Flow past disks

In our second example we simulated a two dimensional flow past three disks at Mach 3, computed by the embedded boundary method. The computational domain is $[-4,4] \times [-4,4]$ with 305x305 grid points. The disks are centered at (-1.3,-1.0), (-1.0,0.8), and (1.3,0.2), with radius 0.35, 0.3, and 0.4 respectively. This example shows one of the best features with the embedded boundary technique; it is very easy to create complex geometry in the computational domain. The Mach number and $\gamma$ are the same as in the first example, 3 and 1.4, respectively, but we have changed the Reynolds number to 1000. Figure 4.6 shows the density contours for the supersonic flow around three disks. Figure 4.7 shows a contour plot of $u$. Finally, in figure 4.9 we show that there are almost no limit to how many disks you can put in the domain or how you want configuration the domain. Table 4.1 show how different numbers of disks affects the overall execution time. We see that increasing the number of disks in the domain only give us a small increase of the execution time.

<table>
<thead>
<tr>
<th>Disks</th>
<th>Ghost points</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>419</td>
<td>3711.69</td>
</tr>
<tr>
<td>3</td>
<td>220</td>
<td>3683.14</td>
</tr>
<tr>
<td>1</td>
<td>116</td>
<td>3570.18</td>
</tr>
</tbody>
</table>

Table 4.1. Execution time, after 4000 time steps, for different number of disks in the domain
Figure 4.6. Density contour for supersonic flow past three disks
Chapter 4. Results

![Contour plot showing the shear layers](image1)

**Figure 4.7.** Contour plot showing the shear layers

![Quiver plot of the velocity around the disks](image2)

**Figure 4.8.** Quiver plot of the velocity around the disks
Chapter 4. Results

Figure 4.9. Density plot

4.3 Parallel performance

A part of this thesis work was to try to run the Euler solver, developed in [4], on a parallel computer. The code uses the MPI library to run on parallel computers with distributed memory. The domain is distributed on the parallel machine automatically. The algorithm tries to make the distribution of grid points as evenly as possible, with local processor patches as close to square as possible. Figure 4.10 shows the behavior of the execution time for the $305 \times 305$ problem as a function of numbers of processors.

The goal in parallel computing is to use $p$ processors to execute a program $p$ times faster than it executes on a single processor. The ratio of the sequential execution time to parallel execution time is called speedup.

\[
\text{Speedup} = \frac{1 \text{ processor execution time}}{p \text{ processors execution time}} \tag{4.4}
\]

The speedup is shown in Figure 4.11 and it is seen that the code scales well at least up to 64 processors for the chosen problem size.
Chapter 4. Results

Figure 4.10. Execution time, after 1000 time steps, as a function of number of processors.

Figure 4.11. Speedup
Chapter 5

Conclusion

A numerical flow simulation of the Navier-Stokes equations with the embedded boundary method has been developed. The code is based on the work done in [4], and is developed in C++ and Fortran. The viscous terms of the Navier-Stokes equations have been approximated by second order finite difference schemes. We have examined how successful it was to use the embedded boundary technique for the Navier-Stokes equations. From the discussion in section 3.4 we expect that we can get higher accuracy with body fitted grid. The accuracy of the mixed derivatives don’t hold second order accuracy. A clear advantage with the embedded boundary method over body fitted grid is that we can “easily” add complex geometrical objects in the domain.

Further developments include generalizations to three space dimensions.
References


