NTNU
Norwegian University of
Science and Technology

Verification of a Cartesian grid method for compressible flow over moving structures

Morten Lyssand Lekven

Master of Energy and Environmental Engineering
Submission date: June 2017
Supervisor: Bernhard Müller, EPT

Norwegian University of Science and Technology
Department of Energy and Process Engineering

Verification of a Cartesian grid method for compressible flow over moving structures

Morten Lekven

June 2017

EPT-M-2017-46

MASTER THESIS

for

Student Morten Lekven

Spring 2017

Verification of a Cartesian grid method for compressible flow over moving structure
Verifikasjon av en kartesiske grid metode for kompressibel strømning rundt bevegelige strukturer

Background and objective

The Cartesian grid method has recently become a popular method in computational fluid dynamics (CFD) to compute flows over or in complex geometries. The reason lies in its simplification of grid generation. Instead of generating a body-fitted structured or unstructured grid, the geometry of a body is embedded into a simple Cartesian grid by taking the effect of the body into account at ghost points, i.e., at grid points inside the solid body near the body surface. In particular, when complex structures are moving like wings in aerodynamics or organs in biomedical applications, the Cartesian grid method relieves CFD of grid generation, which is required for body-fitted grids at every time step.

The objective of the master project is to develop, implement and verify a Cartesian grid method for 2D compressible flow over moving structures, e.g. oscillating cylinders or airfoils. An existing Cartesian grid code for compressible flow over stationary cylinders and airfoils may be used as an example. The master project is linked to the larger interdisciplinary research project *Modeling of obstructive sleep apnea by fluid-structure interaction in the upper airways*, which is a collaboration between NTNU, SINTEF and St. Olavs Hospital and funded by the Research Council of Norway.

The following tasks are to be considered:

1. Develop and implement a Cartesian grid method to numerically solve the 2D Euler equations for compressible flow over moving structures.
2. Verify the Cartesian grid method for suitable test cases of 2D inviscid compressible flow over moving structures.
3.Assess the accuracy and efficiency of the Cartesian grid method for compressible flow over moving structures by comparison with other methods.
4. Suggest improvements of the Cartesian grid method for compressible flow over moving structures.

-- " --

Within 14 days of receiving the written text on the master thesis, the candidate shall submit a research plan for his project to the department.

When the thesis is evaluated, emphasis is put on processing of the results, and that they are presented in tabular and/or graphic form in a clear manner, and that they are analyzed carefully.

The thesis should be formulated as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents etc. During the preparation of the text, the candidate should make an effort to produce a well-structured and easily readable report. In order to ease the evaluation of the thesis, it is important that the cross-references are correct. In the making of the report, strong emphasis should be placed on both a thorough discussion of the results and an orderly presentation.

The candidate is requested to initiate and keep close contact with his/her academic supervisor(s) throughout the working period. The candidate must follow the rules and regulations of NTNU as well as passive directions given by the Department of Energy and Process Engineering.

Risk assessment of the candidate's work shall be carried out according to the department's procedures. The risk assessment must be documented and included as part of the final report. Events related to the candidate's work adversely affecting the health, safety or security, must be documented and included as part of the final report. If the documentation on risk assessment represents a large number of pages, the full version is to be submitted electronically to the supervisor and an excerpt is included in the report.

Pursuant to “Regulations concerning the supplementary provisions to the technology study program/Master of Science” at NTNU §20, the Department reserves the permission to utilize all the results and data for teaching and research purposes as well as in future publications.

The final report is to be submitted digitally in DAIM. An executive summary of the thesis including title, student’s name, supervisor's name, year, department name, and NTNU's logo and name, shall be submitted to the department as a separate pdf file. Based on an agreement with the supervisor, the final report and other material and documents may be given to the supervisor in digital format.

□ Work to be done in lab (Water power lab, Fluids engineering lab, Thermal engineering lab)
□ Field work

Department of Energy and Process Engineering, 15. January 2017

Bernhard Müller
Academic Supervisor

Abstract

A simplified, low order finite volume Cartesian grid method for inviscid compressible flow over rigid, moving structures is developed and tested in two spatial dimensions to assess the treatment of the moving boundary and the potential of the Cartesian grid method for solving problems with complex boundaries. The method is second order accurate when the local Lax-Friecrichs method with MUSCL and minmod-limiter is used, and first order accurate without MUSCL. The boundary conditions are imposed via ghost points located inside the structure. The values of the ghost points G_1 are set based on the corresponding fluid points F_1, the fluid points closest to the ghost points, shifted either in the x- or y-direction or diagonally from the ghost point. Symmetry-like boundary conditions are imposed.

The density and pressure of the ghost point G_1 are set equal to the density and pressure of the corresponding fluid point F_1. The velocity of the ghost point is set such that the normal velocity component at the boundary, when determined by linear interpolation using the ghost point and a mirror point M_1, is equal to the normal velocity component of the boundary. The mirror point M_1 is located in the fluid domain, the same distance from the boundary as the ghost point, along a line passing through the ghost point G_1 and the fluid point F_1. The velocity at the fluid point M_1 is determined by linear interpolation or extrapolation, depending on the position relative to the fluid point F_1, using the fluid point F_1 and the fluid point one step further into the fluid domain, F_2.

Emerging fluid points are points that, due to the moving boundary, were ghost points in the solid domain at the previous time level, and are fluid points at the current time level. The density and pressure of such points are kept as they were at the previous time level. The normal velocity component of the emerging fluid point is set equal to the normal velocity component of the boundary, and the tangential velocity component is set equal to the tangential velocity component that the point itself had at the previous time level. Other methods for ghost and emerging fluid point treatment have been implemented, tested and found less accurate.

The method is implemented and tested for two subsonic examples in two dimensions, a moving piston and a moving cylinder. The resulting rates of convergence are as expected, two for the method with MUSCL and one for the method without MUSCL. However, as the moving boundary treatment is shown to be a limitation of the method, the accuracy of the method is expected to increase if a more sophisticated ghost point treatment is implemented. The fact that the computational effort required by boundary treatment is only a small part of the total computational effort further implies that implementing a more sophisticated method might be beneficial, yielding higher accuracy per computational effort. The computational effort required for the internal solver with MUSCL is substantially larger than without MUSCL, but as the increased accuracy is even more substantial, it is beneficial to use MUSCL.

Sammendrag

En forenklet, Kartesisk grid metode for ikke-viskøs, kompressibel strømning rundt bevegelige strukturer av lav orden har blitt utviklet og testet for todimensjonale problem for å evaluere behandlingen av den bevegelige grensen og metodens potensial for å løse problem med komplekse grenser. Metoden er av andre orden når den lokale Lax-Friedrichs metode med MUSCL og minmod-begrenser brukes, og første orden uten. Grensebetingelsene introduseres ved hjelp av spøkelsespunkter lokalisert i strukturen. Verdiene til spøkelsespunktene er basert på de korresponderende fluidpunktene, fluidpunktene nærmest spøkelsespunktene, forskjøvet enten i x- eller y-retning, eller diagonalt fra spøkelsespunktet. Symmetrilignende grensebetingelser brukes.

Densiteten og trykket ved spøkelsespunktet G_1 settes lik densiteten og trykket ved det korresponderende fluidpunktet F_1. Farten ved spøkelsespunktet settes slik at normalhastighetskomponenten ved grensen, når den interpoleres lineært mellom spøkelsespunktet og et speilpunkt, M_1, er lik normalhastighetskomponenten til grensen. Speilpunktet M_1 er i fluiddomenet, samme avstand fra grensen som spøkelsespunktet, på linjen som går gjennom spøkelsespunktet G_1 og det korresponderende fluidpunktet F_1. Farten ved speilpunktet M_1 finnes ved interpolering eller ekstrapolering, alt etter punktets posisjon i forhold til F_1, av F_1 og punktet et steg lenger inn i fluiddomenet, F_2.

Nye fluidpunkter er punkt som, på grunn av grensen i bevegelse, var spøkelsespunkter i den solide strukturen ved forrige tidsnivå, og er fluidpunkter på nåværende tidspunkt. Densiteten og trykket ved slike punkter holdes som de var ved forrige tidsnivå. Normalhastighetskomponenten til det nye punktet settes lik normalhastighetskomponenten til grensen, og tangentialhastighetskomponenten settes som tangentialhastighetskomponenten ved punktet var på forrige tidsnivå. Andre metoder for å behandle spøkelsespunkter og nye fluidpunkter har blitt implementert, testet og funnet mindre nøyaktige.

Metoden er implementert og testet for to subsoniske eksempler i to dimensjoner, et stempel og en sylinder i bevegelse. De resulterende konvergensratene er som forventet, to med MUSCL og en uten MUSCL. Da behandlingen av den bevegelige grensen har vist seg å begrensningen til metoden, forventes det at nøyaktigheten til metoden øker dersom en mer sofistikert metode for å beregne verdiene til spøkelsespunktene blir implementert. Det faktum at beregningene som behandlingen av grensen trenger bare er en liten del av beregningene som behøves totalt impliserer at implementering av en mer sofistikert metode kan være fordelaktig, og resultere i bedre nøyaktighet i forhold til mengden beregninger som behøves. Mengden beregning som trengs av metoden med MUSCL er betydelig større enn uten MUSCL, men da forskjellen i nøyaktighet er enda større, er det fordelaktig å bruke metoden med MUSCL.

Contents

Nomenclature

Δx	Grid spacing in x-direction
Δy	Grid spacing in y-direction
γ	Ratio of specific heats
$\mathbf{R}$	Residual
$\mathbf{U}$	Conservative flow variables vector
$\mathbf{V}$	Primitive flow variables vector
∇	Nabla operator
ρ	Density
$\mathbf{u}$	Velocity vector
c_p	Specific heat at constant pressure
c_v	Specific heat at constant volume
E	Specific total energy
e	Specific internal energy
h	Specific enthalpy
i_{f1}	Index of fluid point closest to the boundary
ni	Number of cells in x-direction
nj	Number of cells in y-direction
R	Specific gas constant
R_U	Universal gas constant
T	Temperature
t	Time
u	Velocity in x-direction
v	Velocity in y-direction
FVM	Finite volume method

IBM Immersed boundary method

OSAS Obstructive sleep apnea syndrome

TVD RK3 3rd order total variation diminishing Runge-Kutta method

List of Figures

List of Tables

Chapter 1

Introduction

In this section the motivation behind the development of the method is given, before the literature review gives an outline of the Cartesian grid methods for flow over stationary and moving boundaries. This section is taken from the project thesis [6].

1.1 Motivation

Compressible flow has been an active area of research within fluid dynamics for decades, covering the behaviour of fluids having variable density. Naturally, this covers all fluids, but in many cases the effects of this variability are negligible, thus modelling them as incompressible, having constant density, is acceptable [7]. The effects of variable density increase with the Mach number, the ratio of the velocity to the speed of sound, so the most relevant areas of application are those involving high-velocity flows, such as aircraft, rockets, gas pipelines, wind and gas turbines etc.

In recent years, as computational capacity has been increasing, computational fluid dynamics has become a very popular alternative to traditional experiments. For structures in motion, such as aircraft, the motion through the stagnant air is simply modelled in a reference frame attached to the aircraft by giving the air entering the computational domain at the inflow boundary a uniform velocity. By that very simple technique, rigid structures in motion are modelled with ease.

However, if the structure is not rigid, or consists of several parts in motion relative to each other, modelling is not as trivial. In such cases, for the body-fitted grid method, the grid needs to be altered for each time step, which requires some computational work, depending on how complex the geometries are and how much they change. Commercial applications include modelling of internal combustion engines, oscillations of wind turbine blades and biomechanical flow.

Modelling of obstructive sleep apnea syndrome (OSAS) by fluid-structure interaction in the upper airways is an ongoing research project investigating the effect of an operational procedure in the nose, and striving to develop a method to predict the individual patient's effect of the procedure on OSAS.

As an important part of this project, a fluid-structure interaction is to be modelled where the structure in question is the soft palate and the surrounding airways, and the fluid is air. An efficient method of modelling fluid flow around moving structures is necessary, which is the motivation behind investigating the performance of a simplified Cartesian grid method. The fact that body-fitted grid methods have to regrid after each time step because the structures have moved causes the method to require quite a lot of computational work that can be avoided with the Cartesian grid method, especially for complex boundaries. Further, as the body-fitted grid method is unable to handle collisions, and struggles more for extensively deformed grids, and the soft palate deforms quite a lot and collides with the surrounding tissue, the advantages of the Cartesian grid method are substantial here. On the other hand, the body-fitted grid method is very well known, implemented on many different platforms in commercially available software, and is thus readily available.

The Cartesian grid method, in general, has small memory requirements, and the equations are uniform in the computational domain, making it more easily implementable and parallelizable on high-performance parallel computers.

1.2 Literature review

The Immersed Boundary Method (IBM) was first introduced in 1972 by Charles Peskin, when used to simulate blood flow around the heart valves [8]. It was carried out on a Cartesian grid, and the effect of the presence of the boundary was introduced via a forcing term. Since then, the method has been further developed and adjusted in numerous ways [9]. When applied to incompressible flow, it is normally named the IBM , while for compressible flow, it is called the Cartesian grid method or the embedded boundary method [1]. When the term "Cartesian grid method" first appeared, it was simulating steady, inviscid flows with complex, embedded boundaries, but has since been extended to simulate unsteady viscous flows [9].

The Cartesian grid method is a method for numerical modelling where the computational domain is discretized on a Cartesian grid, and the boundary of the fluid domain intersects the grid arbitrarily. For stationary boundaries, the most significant advantages, compared to the more conventional body-fitted grid method, are the simplified grid generation, or meshing, and the efficient numerical methods, resulting from the fact that the grid is both orthogonal and equidistant, not requiring any calculation originating from grid transformation [10]. When it comes to moving boundaries, another advantage arises, as the grid does not require any regridding as the body-fitted grid method does. The body-fitted grid methods also require some additional information about the surface area of the interfaces between the nodes, and, for the unstructured grids, how the nodes are connected.

Another advantage of the Cartesian grid methods, according to Forrer and Jeltsch, is that it can take full advantage of fast computer architectures like vector or parallel computers [11]. On the other hand, the method is not designed to, nor able to strictly conserve mass [9].

The cut-cell method lets the boundary arbitrarily intersect cells on a stationary, Cartesian background grid. It divides the cells it intersects into two smaller cells, one on the inside of the fluid domain, a so-called cut-cell, and one on the outside, in the solid structure domain.

The cut-cell method has some of the advantages of the Cartesian grid method and the body-fitted grid method. It requires grid regeneration only in the close vicinity of the moving boundary. The boundary conditions can be trivially enforced at the boundaries, and a finite volume version of the method ensures strict global and local conservation, e.g. of mass [9]. On the other hand, the cut-cells can get very small, which can cause both numerical instabilities and oscillations [12] [13]. These can be controlled, for example by merging the small cut-cells. Further, the method is not trivial to implement, and the complexity increases with the number of spatial dimensions considered. If extended beyond the inviscid assumption, due to the complex polyhedral cells emerging, the discretization of the full Navier-Stokes equations is also complicated. However, it has been successfully implemented for three-dimensional problems with moving boundaries and seems to be a relevant method for flow over moving structures [13] [9].

To avoid the problems induced by the cut-cell method, a new boundary treatment was developed, using ghost points. A ghost point is an additional grid point, located outside the fluid domain, yet the values of the ghost points are taken into account when solving the equations, thus affecting the solution in the fluid domain. When Dirichlet boundary conditions are given, the ghost point values are set in such a way that the values in question at the boundaries get the desired value. If von Neumann boundary conditions are to be set, the values of the ghost points are set such that the approximation of the spatial derivatives of the variables in question gets the desired value.

As the simplified Cartesian grid method for moving boundaries was developed, a number of different ghost point treatments for embedded boundaries given in the literature, as well as combinations of these were used as inspiration for the proposed boundary treatments implemented and tested. The most important of these are therefore described, in short, here.

1.2.1 Cartesian grid method for stationary boundaries

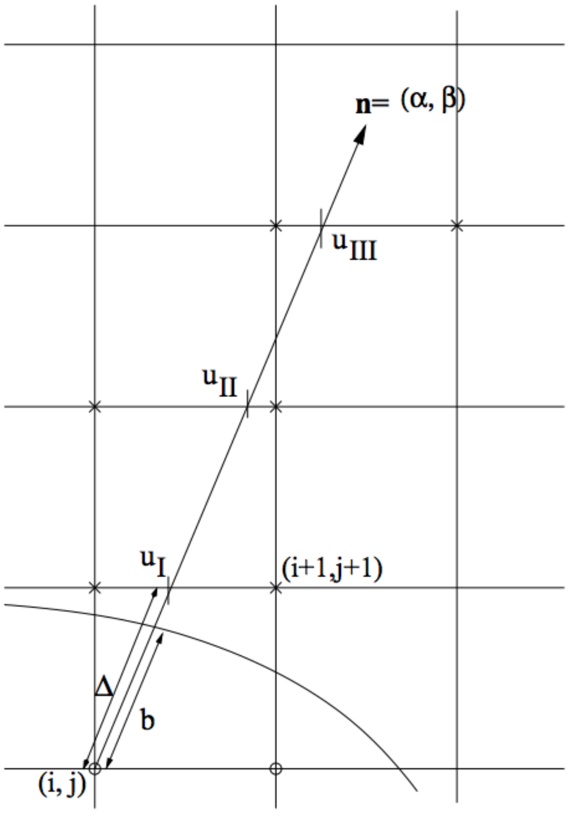

Figure 1.1: Ghost point treatment developed by Sjögreen and Petersson [1].

Björn Sjögreen and Anders Petersson [1] have developed a boundary treatment using interpolation and extrapolation to set the values at the ghost points. The method is here described for two-dimensional application.

Each ghost point has a known normal vector, a vector normal to the surface, on which the ghost point is located, as shown by Fig. 1.1. For physical, Dirichlet boundary conditions, the value u_b is known at the surface, and the ghost point value, u_g, is set by extrapolating u_b using a minmod-limited slope approximated from values at the three intersections between the grid lines in the x-direction and the normal vector, u_I, u_{II} and u_{III}. These are calculated by interpolation of the values from the closest grid points to the left and to the right of the intersection points, denoted by crosses in Fig. 1.1.

For the numerical boundary conditions, the interpolated value from the first intersection, u_I, is simply extrapolated along the normal vector, using a minmod-limited slope, with the slopes between the intersection points, $(u_I - u_{II})/\Delta$ and $(u_{II} - u_{III})/\Delta$, as input. When resulting in unphysical, negative values, such as density or pressure, at the ghost point, the interpolated value at the first intersection point, u_I, is simply copied.

As proposed by Sjögreen and Petersson, adding slope limiters to obtain zero flux boundary conditions to avoid unphysical oscillations near discontinuities, such as shocks, is necessary to achieve accurate solutions.

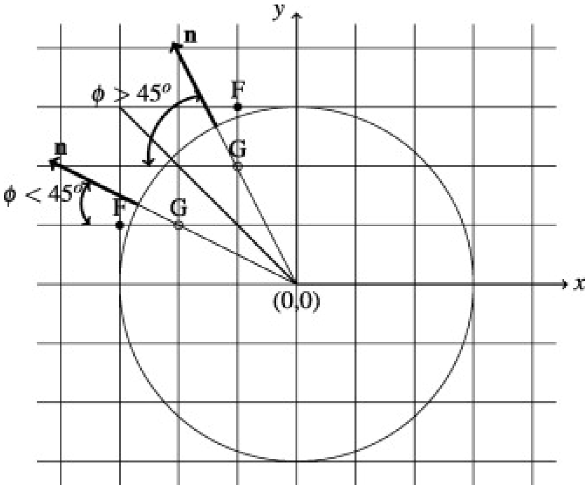

Figure 1.2: Fluid point selection developed by Farooq, demonstrated on a cylinder [2].

Asif Farooq developed, as part of his doctoral thesis, a simplified ghost point treatment for embedded boundaries in two spatial dimensions. The idea is that, as in the method of Sjögreen and Petersson, the vector normal to the surface on which the ghost point G is located, is known. Further, depending on the direction of the normal vector, a fluid point F is chosen. As demonstrated by Fig. 1.2, if the angle ϕ of the normal vector, relative to the negative x-direction, is between -45°and 45°, the fluid point to the left of the ghost point is chosen, if the angle is between 45°and 135°, the upper point is chosen, and so on. Further, as an approximation, it is assumed that the boundary is located exactly the same distance from both of these points. The boundary is set to be impermeable, such that

$$(\mathbf{u} \cdot \mathbf{n})_G = -(\mathbf{u} \cdot \mathbf{n})_F, \tag{1.2.1}$$

where $\mathbf{u}$ is the velocity vector and $\mathbf{n}$ is the unit normal vector at the surface. For the Euler equations the flow is inviscid, hence the slip-condition, given as

$$(\mathbf{u} \cdot \mathbf{t})_G = (\mathbf{u} \cdot \mathbf{t})_F, \tag{1.2.2}$$

where $\mathbf{t}$ is the unit tangential vector at the surface, is implemented. Further, the density and pressure are approximated as symmetric with respect to the solid boundary, thus resulting in a zero gradient at the boundary, given as

$$\rho_G = \rho_F \tag{1.2.3}$$

and

$$p_G = p_F. \tag{1.2.4}$$

As the relation between the pressure, velocity and energy density is given as

$$p = (\gamma - 1)(\rho E - \frac{1}{2}\rho||\mathbf{u}||^2), \tag{1.2.5}$$

and the absolute value of the velocity is equal in the fluid point and the ghost point , the energy densities are also equal,

$$(\rho E)_G = (\rho E)_F. \tag{1.2.6}$$

From equation (1.2.1) and (1.2.2), it follows that

$$u_G = u_F - 2\big[n_x u_F + n_y v_F\big]n_x \tag{1.2.7}$$

$$v_G = v_F - 2\big[n_x u_F + n_y v_F\big]n_y \tag{1.2.8}$$

where u and v are the velocities, and n_x and n_y are the normal vector components, in x- and y-direction, respectively.

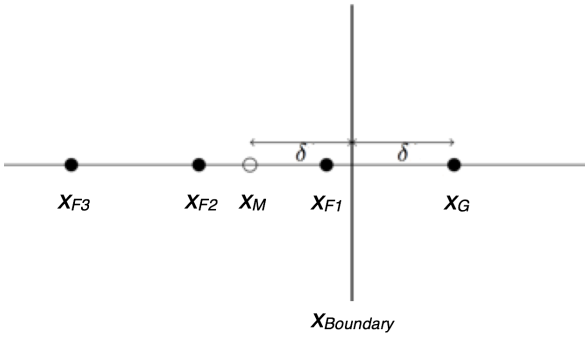

Figure 1.3: Boundary and nearby fluid points, ghost point and constructed mirror point [3].

Farooq also developed a slightly more complicated ghost point treatment for embedded boundaries in one spatial dimension, where a mirror point, located the same distance from the boundary as the ghost point, δ, is constructed. The values of the mirror point are approximated from the neighbouring points, the two or three points in the fluid domain closest to the boundary. If the mirror point is closer to the boundary than the closest fluid point, this is done by extrapolation, otherwise, it is done by interpolation. Either way, the same relation is used to determine the value, given as

$$\mathbf{U}_m = \mathbf{U}_{F2} + \frac{\mathbf{U}_{F1} - \mathbf{U}_{F2}}{\Delta x}(x_m - x_{F2}) \tag{1.2.9}$$

for two points. $\mathbf{U}$ is the vector of conservative flow variables, Δx is the grid spacing, and the subscripts represent the locations of the points relative to the boundary, as shown by Fig. 1.3.

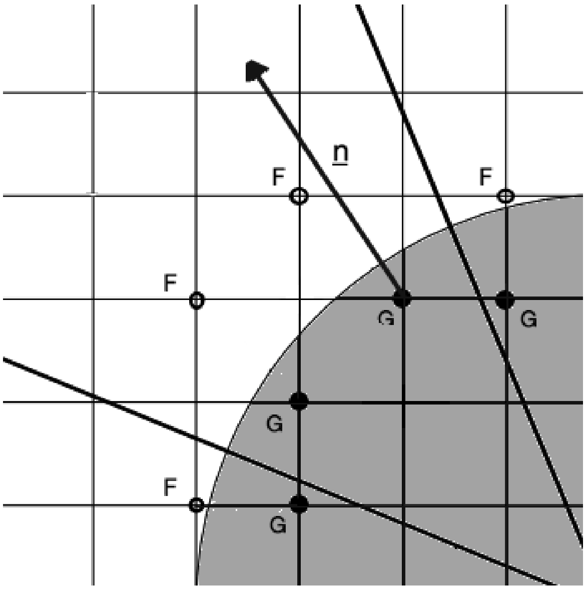

Figure 1.4: Fluid point selection developed by Skøien, demonstrated on a cylinder [4].

Are Skøien [4] improved upon Farooq's simplified treatment in his master's thesis, developing the so-called simplified ghost point treatment for embedded boundaries in two spatial dimensions. The method is based on the idea that the selected fluid point F is shifted diagonally from the ghost point G when the angle of the normal vector is closer to 45° than 90°, as shown by Fig. 1.4. The values are then transferred to the ghost point in a similar fashion to Farooq's two-dimensional method, as shown by equations (1.2.3), (1.2.4), (1.2.7) and (1.2.8).

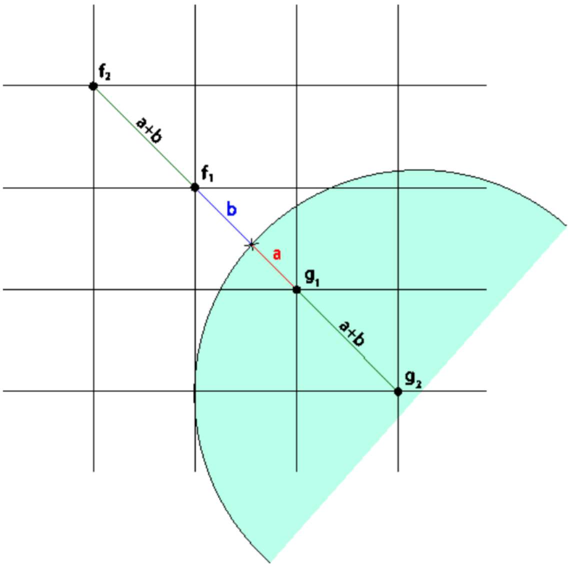

Figure 1.5: Boundary and nearby fluid points, ghost points and lengths a and b [4].

Further, Skøien developed the weighted ghost point method, adding weights based on the distance to the surface when setting the velocity of the ghost point. This is implemented as

$$\begin{aligned} u_{g1} &= -\frac{a}{b} u_{f1} \\ v_{g1} &= -\frac{a}{b} v_{f1}, \end{aligned} \tag{1.2.10}$$

where a and b are lengths defined in Fig. 1.5. As in equations (1.2.3) and (1.2.4), the density and pressure is still set such that the gradient is zero at the surface.

1.2.2 Cartesian grid method for moving boundaries

Hans Forrer and Marsha Berger [14] developed a method using so-called mirror flow, a smooth extrapolation of the flow variables, to set the values of the ghost points, such that the flow is symmetric about the boundary. For moving boundaries, this can only be achieved very locally, due to the pressure and density gradients close to the wall, resulting from the movement of the boundary.

Sirui Tan and Chi-Wang Shu [5] developed a high order moving boundary treatment for inviscid, compressible flow. It is based on an extension of the inverse Lax-Wendroff procedure described in [15]. As the solution algorithm for the internal domain is of a high order, using a seven-point stencil, three layers of ghost points need be constructed. To simplify the implementation, by enabling the possibility of treating newly emerging fluid points the same way as ghost points, another, fourth layer of ghost points is constructed. The method is based on extrapolation of the characteristic variables of the Euler equations, Lagrangian type extrapolation for smooth solutions, and weighted essentially non-oscillatory (WENO) type extrapolation for discontinuous solutions. The derivatives of the characteristic variables are approximated by Taylor expansion, and the inverse Lax-Wendroff method, using material derivatives to insert spatial derivatives for temporal derivatives, is used.

The results are accurate, achieving third order accuracy at the boundary. The algebra becomes heavy for fourth order accuracy, and was only implemented for one-dimensional problems. The method was not tested for complex geometries, which is where the Cartesian method has a promising advantage compared to the body fitted grid method due to the fact that the body fitted grid must be regenerated, at least to a certain degree, with every movement of the boundaries [10].

1.3 Scope

A simplified Cartesian method for inviscid, compressible flow is developed, tested and verified for problems in two spatial dimensions. The boundary conditions are imposed by ghost points, and several methods of setting the values of the flow variables at the ghost points and the emerging fluid points are implemented and tested to identify which combinations yield the most accurate results. To verify that the method is solving the equations correctly, the entropy error and the mass conservation error of isentropic flow problems found in literature are analysed, thus determining the order of convergence of the method. The lift and drag forces on a body are qualitatively analysed for further assurance. The errors mentioned are analysed to point out the strengths and weaknesses of the method. Two residual solvers and two time stepping methods are to be implemented, tested and compared in terms of accuracy and computational time.

1.4 Overview of content

In section 2, the governing equations will be introduced, namely, the Euler equations and the boundary conditions are briefly discussed. An outline of the chosen numerical method, the node-centred finite volume method (FVM), as well as the numerical treatment of the boundary conditions and the time discretization, is given in section 3. As this theoretical background is the same as what was relevant for the preceding project thesis, sections 2 and 3 are taken directly from the project thesis [6]. Section 4 describes the ghost point treatment, and section 5 gives an in-depth description and discussion of the various methods developed for setting the values of the flow variables of the ghost points and the emerging fluid points. Section 6 covers the numerical examples, the results, and discussions of the results. In section 7, the conclusions are given, before suggestions of future work are presented in section 8, which are quite similar to the future work suggested in the project thesis [6].

Chapter 2

Governing equations

2.1 Euler equations

The motion of the inviscid fluid with no body forces is governed by three fundamental conservation laws:

- the continuity equation, describing the conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1.1}$$

  where ρ is the density, t is the time, ∇ is the nabla operator, and $\mathbf{u}$ is the velocity vector.

- the momentum equation, describing Newton's second law of motion

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot \rho \mathbf{u}\mathbf{u} + \nabla p = 0 \tag{2.1.2}$$

  where $\rho \mathbf{u}$ is the momentum, and p is the pressure.

- the energy equation, describing the first law of thermodynamics

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho E + p)\mathbf{u} = 0 \tag{2.1.3}$$

  where ρE is the energy density.

We assume perfect gas properties, and the following equations of state apply:

$$p = \rho R T \tag{2.1.4}$$

$$R = c_p - c_v \tag{2.1.5}$$

$$\gamma = \frac{c_p}{c_v} \tag{2.1.6}$$

$$e = c_v T \tag{2.1.7}$$

where R is the specific gas constant, T is the temperature, γ is the ratio of the specific heats, 1.4 for air at ordinary temperatures, e is the specific internal energy, and c_p and c_v are the specific heat at constant pressure and volume, respectively.

Combining equations (2.1.4), (2.1.5), (2.1.6) and (2.1.7), we obtain a useful relationship for the pressure p:

$$p = (\gamma - 1)(\rho E - \frac{1}{2}\rho|\mathbf{u}|^2) \tag{2.1.8}$$

where $|\mathbf{u}|$ is the Euclidean norm of the velocity $\mathbf{u}$.

Combining (2.1.8) with the conservation laws (2.1.1) (2.1.2) and (2.1.3), we obtain the two-dimensional Euler equations in conservative form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \tag{2.1.9}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho E + p)u \end{bmatrix}, \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho E + p)v \end{bmatrix} \tag{2.1.10}$$

By rearranging equation (2.1.9), we obtain an expression for the time derivative of the conservative flow variables, the residual $\mathbf{R}$:

$$\frac{\partial \mathbf{U}}{\partial t} = -\Big[\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y}\Big] = \mathbf{R}(\mathbf{U}) \tag{2.1.11}$$

The Euler equations can be written in terms of the primitive variables, density, velocity and pressure, as shown in equation (2.1.12) for $\partial/\partial y = 0$.

$$\frac{\partial \mathbf{V}}{\partial t} + B\frac{\partial \mathbf{V}}{\partial x} = 0 \tag{2.1.12}$$

where $\mathbf{V} = (\rho,\ u,\ v,\ p)^T$ and $B_{i,j} = \partial \mathbf{F}_i/\partial \mathbf{V}_j$.

Further, the system can be written in characteristic form

$$\frac{\partial \mathbf{W}}{\partial t} + \Lambda\frac{\partial \mathbf{W}}{\partial x} = 0 \tag{2.1.13}$$

where $\Lambda = diag(u - c,\ u,\ u,\ u + c)$, a diagonal matrix containing the eigenvalues of the matrix B. The eigenvalues represent the propagation velocities of the four waves on which information is transmitted, the entropy and vorticity waves as well as the two acoustic waves. For each of the three eigenvalues, λ_l, a characteristic exists, where $dx_l/dt = \lambda_l$, such that $\frac{dw_l}{dt} = 0$, so that the characteristic variable w_l is constant. The characteristic variables are defined as

$$\partial \mathbf{W} = T^{-1}\partial \mathbf{V} \tag{2.1.14}$$

where T is the right eigenvector matrix of B, and $\mathbf{V}$ is the vector of primitive flow variables.

Fig. 2.1 shows the characteristics for subsonic flow with positive velocity in the x-direction, where C_+ and C_- are the acoustic waves, propagating with velocities $u + c$ and

$u - c$, respectively. C_0 is the characteristic of the entropy and vorticity waves, on which the entropy and vorticity remains constant, propagating with the velocity u.

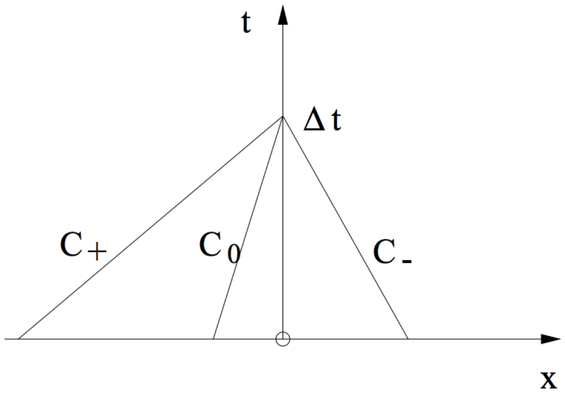

Figure 2.1: Characteristics in subsonic flow [3].

Compressible flow is grouped according by the Mach number, defined as

$$Ma = \frac{u}{c}, \tag{2.1.15}$$

where the speed of sound c is given as

$$c = \sqrt{\frac{p\gamma}{\rho}}, \tag{2.1.16}$$

u is the velocity of the fluid, p is the absolute pressure, γ is the ratio of the specific heats, and ρ is the density. When $Ma < 1$, the flow is classified as subsonic in the x-direction, and when $Ma > 1$, it is supersonic in the x-direction. This affects the propagation direction of the characteristic C_{-}, shown in Fig. 2.1, which again affects the number of physical and numerical boundary conditions that need to be prescribed.

2.2 Boundary conditions

To achieve stable and accurate solutions, the problem must be well-posed, and requires the correct amount of specified boundary conditions. For hyperbolic systems, the number of boundary conditions that must be specified is closely related to the characteristic equations mentioned in the previous section. The information is transmitted on the characteristics, and only information originating outside of the domain can be specified at the boundary as boundary conditions.

For subsonic flow at an outflow boundary, one physical and three numerical boundary conditions must be given for two dimensional problems, while for three dimensional problems, four numerical boundary conditions must be given. For supersonic flow at an outflow boundary, no characteristics are propagating into the fluid domain from the exterior, thus no physical boundary conditions must be given.

For subsonic flow at an inflow boundary, i.e., $-1 < (\mathbf{u} \cdot \mathbf{n})/c < 0$, where $\mathbf{n}$ is the outer normal unit vector, only one characteristic is propagating from the interior of the fluid domain to the exterior. Therefore, only one numerical boundary condition must be given, and three and four physical boundary conditions for two- and three dimensional problems, respectively.

Only the impermeability boundary condition at walls has been implemented here, but others, like periodic or non-reflecting boundary conditions, can also be implemented.

Chapter 3

Numerical method

3.1 Node-centered FVM

The node-centred finite volume method (FVM) divides the computational domain into cells. The values at the points, conveniently placed at the centre of the cells, represent the averages of the values in the cell, given by

$$\overline{\mathbf{U}}_P = \frac{1}{V_P} \iiint_{V_P} \mathbf{U} dV \tag{3.1.1}$$

where V_P is the volume of the cell P.

The numerical fluxes are defined at the cell faces, indicated by half-indices. The spatial-discretization of equation (2.1.11) yields:

$$\frac{\partial \mathbf{U}_{i,j}}{\partial t} = -\Big[\frac{\mathbf{F}_{i+1/2,j} - \mathbf{F}_{i-1/2,j}}{\Delta x} + \frac{\mathbf{G}_{i,j+1/2} - \mathbf{G}_{i,j-1/2}}{\Delta y}\Big] \tag{3.1.2}$$

3.2 Local Lax-Friedrichs method

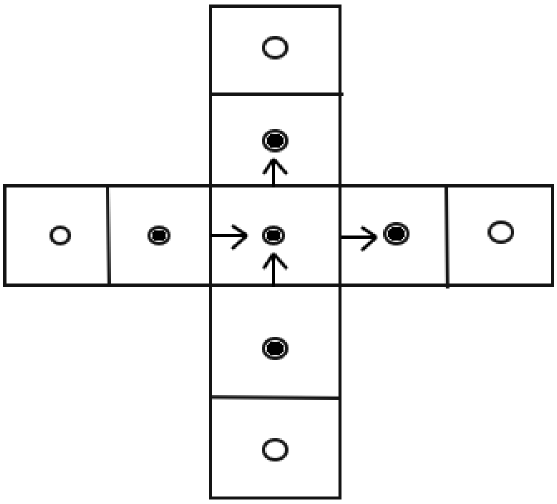

Figure 3.1: An example of a cell, the numerical fluxes indicated by arrows, and the domain of dependence of the cell, indicated by circles for the local Lax-Friedrichs method with MUSCL, and filled circles for the method without MUSCL.

To determine the numerical fluxes at the cell faces, shown by Fig. 3.1, the local Lax-Friedrichs or Rusanov method is used. It is easily implemented, being first order, and easily expanded to higher order of accuracy if necessary. In two dimensions, the fluxes are approximated by equations (3.2.1) and (3.2.2):

$$\begin{aligned}\mathbf{F}^{lLF}_{i+1/2,j} =&\frac{1}{2}[\mathbf{F}(\mathbf{U}_{i,j}) + \mathbf{F}(\mathbf{U}_{i+1,j}) - \\ & max(|u_{i,j}| + c_{i,j}, |u_{i+1,j}| + c_{i+1,j})(\mathbf{U}_{i+1,j} - \mathbf{U}_{i,j})]\end{aligned} \tag{3.2.1}$$

$$\begin{aligned}\mathbf{G}^{lLF}_{i,j+1/2} =&\frac{1}{2}[\mathbf{G}(\mathbf{U}_{i,j}) + \mathbf{G}(\mathbf{U}_{i,j+1}) - \\ & max(|v_{i,j}| + c_{i,j}), |v_{i,j+1}| + c_{i,j+1})(\mathbf{U}_{i,j+1} - \mathbf{U}_{i,j})]\end{aligned} \tag{3.2.2}$$

3.3 MUSCL with minmod limiter

To increase the accuracy of the method above, a Monotone Upwind-centered Scheme for Conservation Laws (MUSCL) with minmod limiter has been implemented. This yields second order accuracy for smooth flow except for at extrema, where the accuracy is still first order [16]. The variables $\mathbf{U}_{i,j}$ and $\mathbf{U}_{i+1,j}$ in equation 3.2.1 are replaced by

$$\mathbf{U}^{L}_{i+\frac{1}{2},j} =\mathbf{U}_{i,j} + \frac{1}{2}\text{minmod}(\mathbf{U}_{i,j} - \mathbf{U}_{i-1,j}, \mathbf{U}_{i+1,j} - \mathbf{U}_{i,j}) \tag{3.3.1}$$

$$\mathbf{U}^{R}_{i+\frac{1}{2},j} =\mathbf{U}_{i+1,j} - \frac{1}{2}\text{minmod}(\mathbf{U}_{i+2,j} - \mathbf{U}_{i+1,j}, \mathbf{U}_{i+1,j} - \mathbf{U}_{i,j}), \tag{3.3.2}$$

respectively, where the minmod limiter is defined as

$$\begin{aligned}\text{minmod}(a,b) &= \begin{cases} a, & \text{if } |a| \leq |b| \text{ and } ab > 0 \\ b, & \text{if } |b| < |a| \text{ and } ab > 0 \\ 0, & \text{if } |ab| \leq 0 \end{cases} \\ \text{minmod}(a,b) &= \text{sign}(a)\ \max(0,\ \min(|a|,\ \text{sign}(a)b)),\end{aligned} \tag{3.3.3}$$

thus choosing the least steep slope if the slopes are of the same sign, otherwise zero.

3.4 Numerical treatment of the boundary conditions

As stated in section 2.2, three numerical, and one physical, boundary conditions must be specified for subsonic, two-dimensional problems at an outflow boundary. As wall boundaries are modelled as impermeable, the normal velocity at the surface, relative to the surface, must be zero. As the surface itself might be in motion, the normal velocity of the fluid at the boundary is set equal to the normal velocity of the boundary, as a Dirichlet boundary condition:

$$(\mathbf{u} \cdot \mathbf{n})_f = (\mathbf{u} \cdot \mathbf{n})_b, \tag{3.4.1}$$

where $\mathbf{u}$ is the velocity vector, $\mathbf{n}$ is the unit normal vector at the surface, and the subscripts f and b represent the fluid domain and the boundary, respectively.

The three remaining, numerical boundary conditions are set as Neumann boundary conditions, so the gradient at the surface is given. One possible approach is to set the gradient of the density and the pressure at the boundary to zero. Further, as boundary layers do not exist for inviscid fluid flow, thus no no-slip condition, but rather a slip condition, the tangential velocity at the surface has a zero normal gradient.

$$\begin{aligned} \frac{\partial \rho_f}{\partial \mathbf{n}} &= 0 \\ \frac{\partial p_f}{\partial \mathbf{n}} &= 0 \\ \frac{\partial \mathbf{u}_f \cdot \mathbf{t}}{\partial \mathbf{n}} &= 0 \end{aligned} \tag{3.4.2}$$

where

$$\begin{aligned} \frac{\partial \rho}{\partial \mathbf{n}} &= n_x \frac{\partial \rho}{\partial x} + n_y \frac{\partial \rho}{\partial y}, \\ \mathbf{n} &= \begin{bmatrix} n_x \\ n_y \end{bmatrix}, \end{aligned} \tag{3.4.3}$$

and $\mathbf{t}$ is the unit tangential vector at the surface of the wall.

Another possible technique is to impose the numerical boundary conditions using the characteristic variables, w_1, w_2 and w_3, as discussed in section 2.1, by setting their gradients at the boundary to zero, as shown by

$$\frac{\partial w_{f,l}}{\partial \mathbf{n}} = 0, \ l = 1, 2, 3. \tag{3.4.4}$$

Yet another possible approach is to let the normal derivative of the characteristics remain constant near the boundary, thus letting the second normal derivative be zero:

$$\frac{\partial^2 w_{f,l}}{\partial \mathbf{n}^2} = 0, \ l = 1, 2, 3. \tag{3.4.5}$$

The ghost points are to be constructed with values such that the internal solver, without any special consideration at or near the boundaries, when solving for the flow variables in the next time step, or next stage in the TVD RK3, cf. section 3.5, achieves the desired effect of the boundary conditions.

3.5 Time discretization

By discretizing the residual function in space, (2.1.11), according to equation (3.1.2), a semi-discrete system of ordinary differential equations is obtained, that can be solved by time-stepping methods, either implicit or explicit, given an initial condition. The system is solved by two simple, explicit methods, the third order total variation diminishing Runge-Kutta method (TVD RK3) and the explicit Euler method.

The TVD RK3 algorithm is given as

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t \mathbf{R}(\mathbf{U}^n) \tag{3.5.1}$$

$$\mathbf{U}^{(2)} = \frac{3}{4}\mathbf{U}^n + \frac{1}{4}\mathbf{U}^{(1)} + \frac{1}{4}\Delta t \mathbf{R}(\mathbf{U}^{(1)}) \tag{3.5.2}$$

$$\mathbf{U}^{(n+1)} = \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\mathbf{U}^{(2)} + \frac{2}{3}\Delta t \mathbf{R}(\mathbf{U}^{(2)}). \tag{3.5.3}$$

In terms of memory requirement, the TVD RK3 requires two memory slots the size of the vector of the flow variables $\mathbf{U}$. It is third order accurate in time, and it has a large stability region, including the intervals $[-2.5,\ 0]$ on the real axis and $[-1.732,\ 1.732]$ on the imaginary axis [16], making it more robust, and opens up for the possibility of larger time steps. However, when it comes to computational time required, it requires the residual to be determined three times per time step, as can be seen from equation (3.5.1), (3.5.2) and (3.5.3).

The explicit Euler method is given as

$$\frac{\mathbf{U^{n+1}} - \mathbf{U^n}}{\Delta t} = \mathbf{R}(\mathbf{U^n}). \tag{3.5.4}$$

In terms of memory requirements, the explicit Euler method is very lean, as it needs to store the vector of flow variables $\mathbf{U}$ in only one location, which is updated at every time step. When it comes to computational effort, it requires the residual to be determined only once per time step. However, as it has a smaller stability region than TVD RK3, including no parts of the imaginary axis, the size of the time step is limited. Smaller time steps yield more time steps, increasing the computational time required. In terms of accuracy, it is only first order accurate.

For the method to converge, it needs to be stable, thus satisfying the Courant-Friedrichs-Lewy (CFL) condition, which is a limitation of the Courant number. For the explicit Euler method the limitation is $C_{max} = 1$, while for TVD RK3, it is $C_{max} \approx 1.5$, for the spatial scheme without MUSCL. With MUSCL, the limitation is $C_{max} \approx 1$ for TVD RK3. The CFL condition is given as

$$C_x + C_y \leq C_{max} \tag{3.5.5}$$

where

$$C_x = \frac{(u + \mid c \mid)\Delta t}{\Delta x} \tag{3.5.6}$$

and

$$C_y = \frac{(v + \mid c \mid)\Delta t}{\Delta y}. \tag{3.5.7}$$

Chapter 4

The simplified Cartesian grid method

The treatment of the moving boundaries, represented by the ghost points, is the most characteristic feature of the simplified Cartesian grid method for flow over moving structures. For the ghost points to have the desired effect on the solution of the internal solver, the ghost points must be identified, and the values of their flow variables must be updated periodically. The ghost points are, as mentioned in section 3.4, points located outside the fluid domain, but in the domain of dependence of the fluid points in the close vicinity of the boundary. The order of accuracy of the boundary treatment is limited by the number of ghost points considered, which is here one or two.

To simulate the presence of a solid structure boundary, ghost points in the vicinity of the boundary, inside the structure, are set in a specific way in order to mimic the effect of the boundary. The values of the ghost points are updated before the residual is determined and after the last stage of the time stepping method employed, as some methods determining the flow variables of the emerging fluid points depend on their values from the previous time step. The emerging fluid points are identified, and their initial conditions are set before the first stage of the time stepping method.

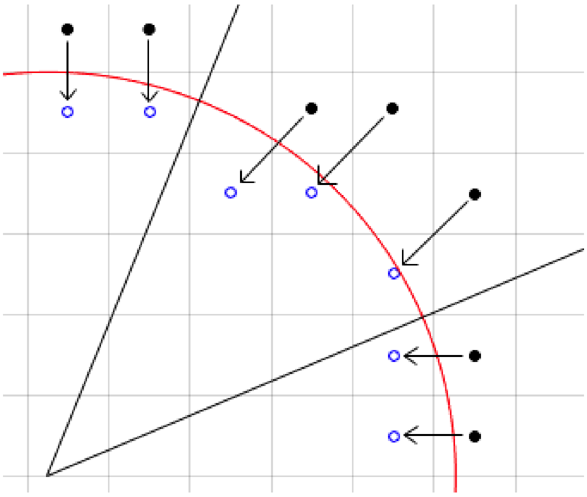

Figure 4.1: Fluid point selection developed by Skøien, demonstrated on the first quadrant of a cylinder [4]. The filled black circles are fluid points, F_1, providing information for the ghost points G_1, the small, blue circles. Their position relative to each other is determined based on the angle of the normal vectors at the surface, angle limits illustrated by the black lines, further specified in equation (4.0.1).

Identifying the ghost points is not trivial. Therefore the technique used to identify the ghost points and the fluid points is described in detail here. For an arbitrary point on the surface, the cell closest to the surface, having its centre in the solid domain, is flagged as a ghost point G_1. The ghost points are recorded in an array, creating a coherent barrier between the fluid domain and the solid domain, called the first layer of ghost points. However, as information is not transferred diagonally by the residual solvers, the points fitting the description above having a neighbouring ghost point in both x- and y-direction are redundant, therefore excluded from the array.

The corresponding fluid point F_1, the source of information for the ghost point G_1 is selected in the same way as in Skøiens method [4], described in section 1.2, and shown in Fig. 4.1. In some regions the selected fluid point F_1 is shifted diagonally from the ghost point G_1, in other regions it is shifted in the x- or y- direction, depending on the direction of the normal vector at the surface. For a ghost point with indices i, j, the indices of the selected fluid point F_1 are given as

$$(i,j)_{F_1} = \begin{cases} (i+1,j), & \text{if } -22.5^\circ < \phi \leq 22.5^\circ \\ (i+1,j+1), & \text{if } 22.5 < \phi \leq 67.5^\circ \\ (i,j+1), & \text{if } 67.5 < \phi \leq 112.5^\circ \\ (i-1,j+1), & \text{if } 112.5 < \phi \leq 157.5^\circ \\ (i-1,j), & \text{if } 157.5 < \phi \leq 202.5^\circ \\ (i-1,j-1), & \text{if } 202.5 < \phi \leq 247.5^\circ \\ (i,j-1), & \text{if } 247.5 < \phi \leq 292.5^\circ \\ (i+1,j-1), & \text{if } 292.5 < \phi \leq 337.5^\circ \end{cases} \tag{4.0.1}$$

where ϕ is the angle of the vector normal to the surface. The fluid point F_2, required for some the methods setting the ghost point values based on mirror points, cf. section 5, is located along the same line as F_1, one step further into the fluid domain from F_1.

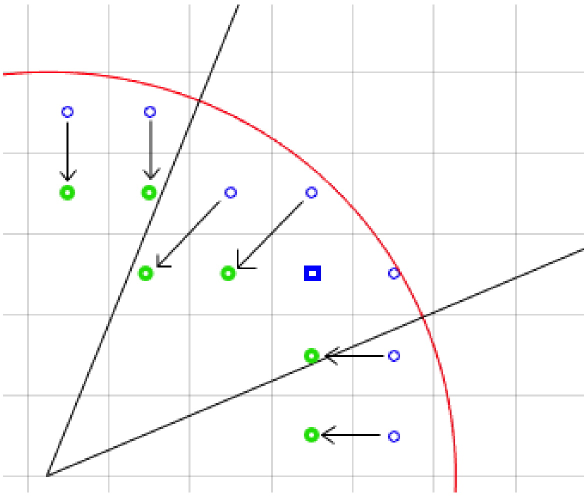

Figure 4.2: Example of the second layer of ghost points for a curved surface, the first quadrant of a cylinder. The green circles represent the second layer ghost points, G_2, the blue circles represent the first layer ghost points, G_1, and the blue square represents a hole.

When the internal solver with MUSCL, described in section 3.3, is utilized, two layers of ghost points are required. One more ghost point is created one step further into the solid domain from each ghost point in the first layer, as shown in Fig. 4.2. This step can be in x- or y-direction, or both, depending on the direction of the normal vector at the surface. This induces a few holes that need special treatment and some redundant ghost points that do not require any attention, but cause some unnecessary computation. The holes occur where the redundant cells mentioned above are excluded in the regions where the ghost points and the corresponding fluid points are shifted diagonally, as illustrated by a blue square in Fig. 4.2. These are simply identified, recorded, and treated like first layer ghost points. The second layer ghost point below the hole, the blue square, in Fig. 4.2 is only within the domain of influence of the fluid point two steps to the right, and is treated as a second layer ghost point originating from the point to the right, not the ghost point shifted diagonally.

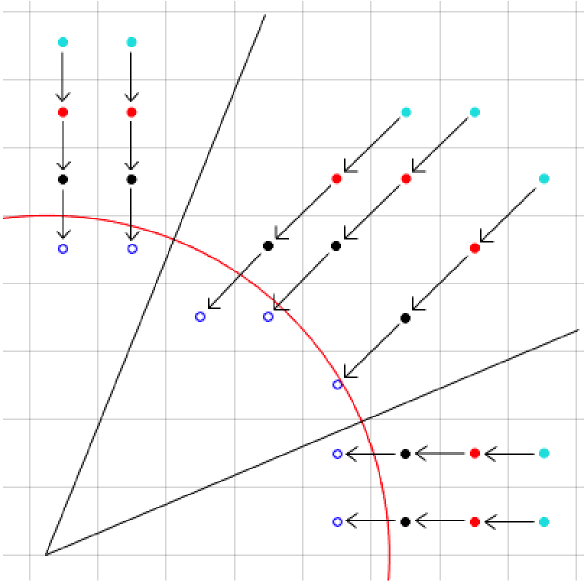

Figure 4.3: Example of the second and third layer of selected fluid points for a curved surface. The red circles mark the second layer fluid points, F_2, and the cyan circles mark the third layer fluid points, F_3.

When another layer of ghost points is constructed, which is necessary when the internal solver with MUSCL is utilized, and a mirror point method, cf. section 5, is used, another fluid point, F_3, is required. It is identified in the same manner as F_2, one step further into the fluid domain, as shown in Fig. 4.3.

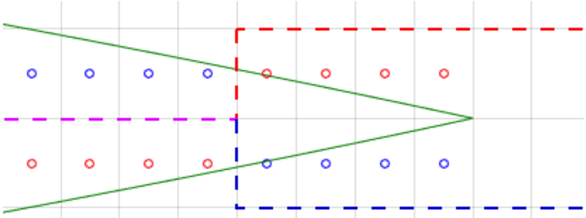

Figure 4.4: Example of the first layer of ghost points for a slender body, near the trailing edge of an airfoil. The blue circles are the ghost points representing the upper surface, and the red circles are the ghost points representing the lower surface. The stippled lines represent the boundaries of the two smaller, temporary computational domains.

For slender bodies, such as the surface of an airfoil near the trailing edge, some of the ghost points are in the fluid domain, as demonstrated by Fig. 4.4, causing issues as these points in the data structure are occupied by the flow variables of the fluid points. The solution here is to divide the computational domain along the stippled lines, yielding two smaller, temporary domains with some overlap, ensuring that the domain of influence of every fluid cell is conserved. The upper computational domain is limited by the blue and purple lines, and the lower domain is limited by the red and purple lines. When the residuals of the fluid cells have been determined for the two smaller computational domains independently, they are combined, yielding a data structure containing the residuals of all the fluid cells. The domain of influence includes only one neighbouring cell in each direction in this example. The method will readily handle larger domains of influence, but requires a bigger overlap.

4.1 The moving cylinder

An example covered in detail in section 6.2 is a 2D simplification of a moving rigid cylinder moving in the x-direction. Developing an algorithm generating the first layer of ghost points proved not to be trivial, and is therefore covered in detail here. This is done once for every time step, and the resulting ghost point locations are compared to those of the previous time step to identify emerging fluid points.

The algorithm is based on symmetry about the x-axis, as the cylinder moves only in the x-direction, and the fact that the x-axis is located at the interface between two layers of cells in the computational domain, as shown by Fig. 4.5. Here, also the y-axis is located at the interface between two layers of cells. If that is not the case, equation (4.1.5) does not apply.

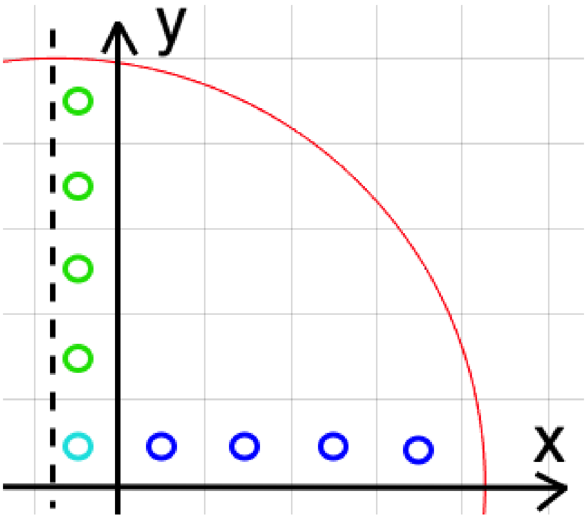

Figure 4.5: The surface of the first quadrant of the circle in red, the x-values of the base coordinates, represented by small, blue circles and the y-values of the base coordinates, represented by green circles. The coordinates of the circle in cyan is in both base coordinate arrays. Here, $t = 0.25$ and $nj = 40$, cf. section 6.2.

For the first quadrant of the circle, an array containing the x-values of the centre of all the cells whose centre is within the first quadrant of the circle is constructed. An equivalent array is constructed for the y-values. These are called base coordinates, and are marked with blue and green circles in Fig. 4.5, respectively. Here, only the x-position of the blue circles and the y-position of the green circles are important, as their position in y- and x-direction, respectively, is arbitrary.

Next, the corresponding surface coordinates for each base coordinate are determined according to

$$R^2 = (x - x_0)^2 + (y - y_0)^2, \tag{4.1.1}$$

where x_0 and y_0 are the coordinates of the centre of the cylinder, and R is the radius of the circle. Every pair consisting of a base coordinate and a surface coordinate is now a point on the surface of the circle, marked by the blue and green lines intercepting the circle in Fig 4.6. The points marked by the green lines have surface coordinates in the x-direction and base coordinates in the y-direction. The opposite is true for the points marked by the blue lines.

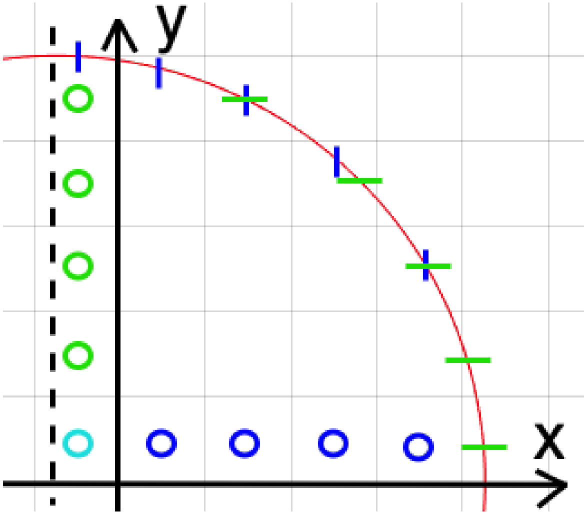

Figure 4.6: The positions of the surface points are given by the interceptions of the coloured lines and the red circle.

Next, the indices of the data points, which are located at the centre of the cells, are determined as

$$i = \begin{cases} \text{round}(\frac{x}{dx} + i_0 - 0.5), & \text{if } \phi \leq 67.5^\circ \\ \text{round}(\frac{x}{dx} + i_0), & \text{if } \phi > 67.5^\circ \end{cases} \tag{4.1.2}$$

and

$$j = \begin{cases} \text{round}(\frac{y}{dy} + j_0), & \text{if } \phi \leq 22.5^\circ \\ \text{round}(\frac{y}{dy} + j_0 - 0.5), & \text{if } \phi > 22.5^\circ \end{cases} \tag{4.1.3}$$

where ϕ is the angle of the vector normal to the surface, and i_0 and j_0 are the pseudoindices of origo given as

$$i_0 = (i_{x=dx} + i_{x=-dx})/2 \tag{4.1.4}$$

and

$$j_0 = (j_{y=dy} + j_{y=-dy})/2, \tag{4.1.5}$$

where dx and dy are given by the number of grid cells in x- and y-direction, ni and nj, respectively, and the size of the computational domain.

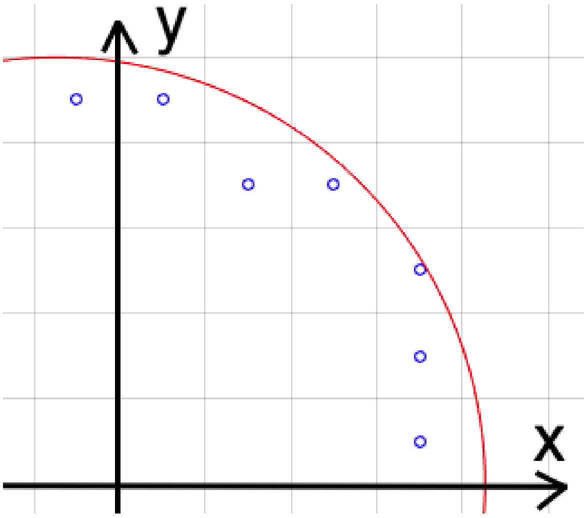

Figure 4.7: The positions of the first layer ghost points, represented by the blue circles, located in the middle of the ghost cells.

This yields a list of potential ghost point indices including some double points, where a point is included twice, and some redundant points, having neighbouring points in both x- and y-direction. These are removed, and the remaining points are sorted according to the angle of their surface vector, determining in which direction their corresponding fluid points F are located, thus determining their indices as

$$i_F = \begin{cases} i_G + 1, & \text{if } \phi \leq 67.5^\circ \\ i_G, & \text{if } \phi > 67.5^\circ \end{cases} \tag{4.1.6}$$

and

$$j_F = \begin{cases} j_G, & \text{if } \phi \leq 22.5^\circ \\ j_G + 1, & \text{if } \phi > 22.5^\circ. \end{cases} \tag{4.1.7}$$

The resulting points are marked by small, blue circles in Fig. 4.7

The process is analogous for the second quadrant, and the ghost point and corresponding fluid point indices for the third and fourth quadrant are determined by mirroring about the x-axis. For this example, it is not necessary to store the coordinates of the surface points, as these are easily determined from the indices of the ghost points, but for other, more complex surfaces, that might be necessary. Only an ordered array of indices and the number of ghost points having their corresponding fluid points in each direction of the eight possible directions, as well as the number of layers of ghost points, are stored. That is enough information to identify all ghost and fluid points.

Chapter 5

Ghost point and emerging fluid point values

The various methods used to set the flow variables of the ghost points and the emerging fluid points are described in detail here. Two specific problems are addressed, how the flow variables of the ghost points are set, and how they are set for the newly emerging fluid points. To investigate the performance of the various methods, the following approaches have been implemented and tested. Results are given in section 6.

5.1 Ghost point values

When the local Lax-Friedrichs method without MUSCL is applied, the stencil of the method at a fluid point includes only one neighbouring point in each direction, thus only one layer of ghost points is required. When MUSCL is applied, the stencil of the method includes two neighbouring points in each direction, and two layers of ghost points must be constructed. The methods setting the flow variables of the ghost points do not distinguish between the layers of ghost points, except for which fluid points are taken into account. A first layer ghost point G_1 is influenced by fluid points F_1 and in some cases F_2, while a second layer ghost point G_2 is influenced by fluid points F_2 and in some cases F_3.

As discussed in section 2.2, for subsonic flow problems in two dimensions, three numerical and one physical boundary conditions must be given at an outflow boundary. Separating the numerical and physical boundary conditions is not as trivial as in one dimension, as the velocity component normal to the boundary is the physical boundary condition, and the velocity component parallel with the boundary is a numerical boundary condition.

Three different approximations of the velocity of the ghost points, and two different ways to set the density and pressure of the ghost points have been implemented and tested.

5.1.1 Velocity

a: The first method, named the very simple method, is primarily used to show that the method functions as planned, and as a basis for comparing the other methods. Assuming that the ghost point is located at the boundary, the normal velocity component $(\mathbf{u}\cdot\mathbf{n})_{G_1}$ is set equal to the normal velocity component of the boundary, $(\mathbf{u}\cdot\mathbf{n})_B$, while the tangential velocity component $(\mathbf{u}\cdot\mathbf{t})_{G_1}$ is set equal to the tangential velocity component of the fluid point, $(\mathbf{u}\cdot\mathbf{t})_{F_1}$. These conditions lead to a linear system of equations, yielding the conditions

$$\begin{aligned} u_{G_1} &= n_x[n_x u_B + n_y v_B] + n_y[n_y u_{F_1} - n_x v_{F_1}] \\ v_{G_1} &= n_y[n_x u_B + n_y v_B] + n_x[n_x v_{F_1} - n_y u_{F_1}], \end{aligned} \tag{5.1.1}$$

where n_x and n_y are the x- and y-components of the normal vector at the surface, u and v are the x- and y-components of the velocity, and the subscripts B, G_1 and F_1 represent the boundary, the first layer ghost point and the corresponding fluid point, respectively, as shown in Fig. 5.1. The derivation of equation (5.1.1) is given in appendix A.

b: The second method, called the simple method, is based on the boundary treatment Asif Farooq et al. developed [2], as described in section 1.2.1. Assuming the boundary is located the same distance from the ghost point, G_1, and the fluid point selected, F_1, the velocity is set according to the boundary conditions, given by equations (3.4.1) and (3.4.2). By requiring

$$\frac{1}{2}((\mathbf{u}\cdot\mathbf{n})_{F_1} + (\mathbf{u}\cdot\mathbf{n})_{G_1}) = (\mathbf{u}\cdot\mathbf{n})_B \tag{5.1.2}$$

and

$$(\mathbf{u}\cdot\mathbf{t})_{G_1} = (\mathbf{u}\cdot\mathbf{t})_{F_1} \tag{5.1.3}$$

the conditions yield

$$\begin{aligned} u_G =& u_{F1} - 2\big[n_x u_{F_1} + n_y v_{F_1}\big]n_x + 2\big[n_x u_B + n_y v_B\big]n_x \\ v_G =& v_{F1} - 2\big[n_x u_{F_1} + n_y v_{F_1}\big]n_y + 2\big[n_x u_B + n_y v_B\big]n_y \end{aligned} \tag{5.1.4}$$

where u and v are the x- and y-components of the velocity, and n_x and n_y are the unit normal vector components, in x- and y-direction, respectively. The derivation of equation (5.1.4) is given in appendix A.

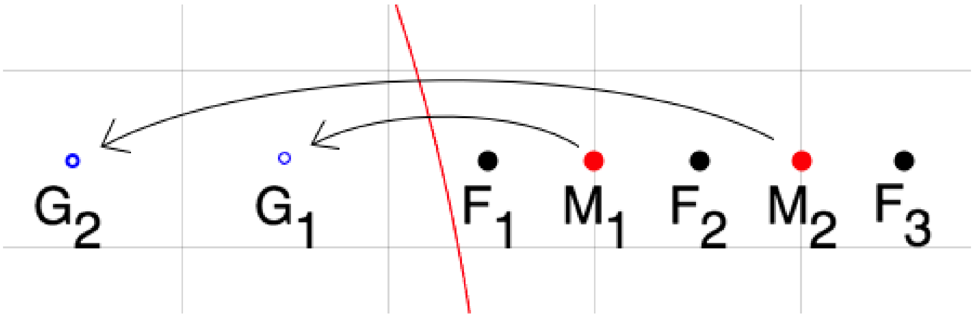

Figure 5.1: Mirror point location demonstrated on a curved surface with two layers of ghost points, with the surface in red, the ghost points as blue circles, the fluid points as black dots and the mirror points as red dots.

c: For the mirror point method, a mirror point M_1, located on the line passing through the ghost point G_1 and the fluid points F_1 and F_2, is constructed. It is positioned the same distance from the boundary as the ghost point G_1, as shown by Fig. 5.1. The primitive flow variables are interpolated or extrapolated, depending on the position of the boundary relative to the selected fluid points, to approximate the flow variables at the mirror point, as given by

$$\mathbf{V}_{M_1} = \mathbf{V}_{F_1} + \frac{\mathbf{V}_{F_2} - \mathbf{V}_{F_1}}{\Delta x} \delta_{F_1, M_1}, \tag{5.1.5}$$

where δ_{F_1,M_1} is the distance between the mirror point and the fluid point F_1. By the same reasoning as for the simple method, b, using the mirror point M_1 where the fluid point F_1 was used, the velocity components are set as

$$\begin{aligned} u_{G_1} &= u_{M_1} - 2\big[n_x u_{M_1} + n_y v_{M_1}\big] n_x + 2\big[n_x u_B + n_y v_B\big] n_x \\ v_{G_1} &= v_{M_1} - 2\big[n_x u_{M_1} + n_y v_{M_1}\big] n_y + 2\big[n_x u_B + n_y v_B\big] n_y. \end{aligned} \tag{5.1.6}$$

The derivation of equation (5.1.6) is given in appendix A.

5.1.2 Density and pressure

I: The first method is named the simple method. The boundary is assumed to be located the same distance from the ghost point G_1 and the fluid point F_1. Further, it is assumed that the normal surface vector is parallel to the line passing through the ghost point G_1 and the fluid point F_1, such that equation (3.4.2) can be approximated by setting the density ρ_{G_1} and pressure p_{G_1} according to

$$\begin{aligned} \rho_{G_1} =& \rho_{F_1} \\ p_{G_1} =& p_{F_1}. \end{aligned} \tag{5.1.7}$$

By applying the pressure relation (2.1.8), the following relation for the total energy density $(\rho E)_G$ is obtained:

$$(\rho E)_{G_1} = (\rho E)_{F_1} + \frac{1}{2} \rho_{F_1} (|\mathbf{u}_{G_1}|^2 - |\mathbf{u}_{F_1}|^2), \tag{5.1.8}$$

where $|\mathbf{u}| = \sqrt{u^2 + v^2}$ is the Euclidean norms of the velocity vector.

II: The primitive variables at the mirror point M given by equation (5.1.5) are used. The density ρ and pressure p of the ghost point are set equal to the values at the mirror point, given as

$$\begin{aligned} \rho_{G_1} &= \rho_{M_1} \\ p_G &= p_{M_1}. \end{aligned} \tag{5.1.9}$$

By applying the pressure relation (2.1.8), the total energy density relation

$$(\rho E)_{G_1} = (\rho E)_{M_1} + \frac{1}{2} \rho_{M_1} (|\mathbf{u}_{G_1}|^2 - |\mathbf{u}_{M_1}|^2) \tag{5.1.10}$$

is obtained.

5.2 Emerging fluid point values

Emerging fluid points are fluid points that were ghost points at the previous time step, and have emerged as fluid points at the current time step as the boundary of the structure moved past the centre of the ghost cell. These emerging fluid points require a set of physical initial conditions, preferably close to the values of the exact solution. The initial conditions are set after the boundary has moved past the centre of the cell, but before the residual is determined for the first stage of the time stepping method . For simplicity, the notation in this section is non-conventional, as superscript n normally refers to the resulting value after all stages of the time stepping method, but here the values with superscript n are used as input in first stage of the time stepping method. The values with superscript $n-1$ have the conventional notation, resulting from the final stage of the time stepping method at time level $n-1$. Three different methods have been developed to set the density ρ and pressure p, and four methods to set the velocity $\mathbf{u}$ of the emerging fluid points.

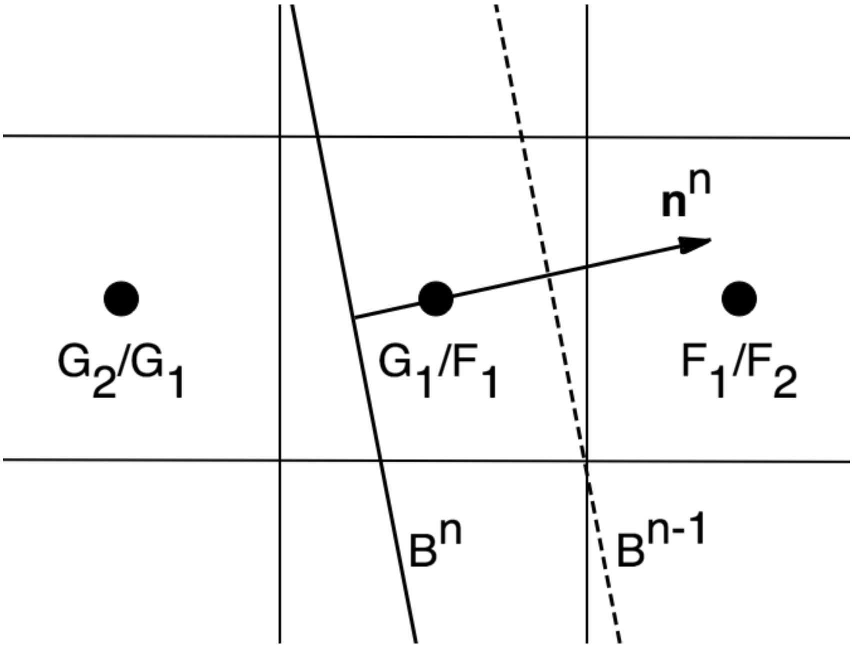

Figure 5.2: Illustration of the concept of emerging fluid points, showing the boundary B at two different times, $n-1$ and n, three data points and their respective notations at the times $n-1$ and n.

5.2.1 Density and pressure

1: The first method is called the simple method with current values. The values are simply kept at their current values as the ghost point is transformed into a fluid point,

$$\begin{bmatrix} \rho \\ p \end{bmatrix}_{F_1}^{n} = \begin{bmatrix} \rho \\ p \end{bmatrix}_{G_1}^{n-1}, \tag{5.2.1}$$

implying that the method chosen to set the density and pressure of the ghost points at the previous stage of the time stepping method has direct impact on the accuracy of this approximation. When this method is used, the ghost point values are updated after the last stage, or only stage, of the time stepping method, marginally increasing the computational effort required, but increasing the accuracy. Note that the cell ${G_1}^{n-1}$ is the same cell as ${F_1}^n$, cf. Fig. 5.2.

2: For this method, the simple method with neighbouring values, the current values of the neighbouring fluid point F_2, as shown in Fig. 5.2, are used. These were last updated as the internal solver solved the internal flow field for the last time step, $n-1$.

$$\mathbf{V} = \begin{bmatrix} \rho \\ p \end{bmatrix}_{F_1}^{n} = \begin{bmatrix} \rho \\ p \end{bmatrix}_{F_1}^{n-1} \tag{5.2.2}$$

Note that the cell ${F_1}^n$ is not the same cell as ${F_1}^{n-1}$, cf. Fig. 5.2.

3: As discussed in section 3.4, one way of setting the numerical boundary conditions is to let the gradients of the characteristics normal to the boundary be zero, as in equation (3.4.4). In this method, the characteristic method for emerging fluid point density and pressure, gradients of the characteristics w_1 and w_2, given as

$$\partial w_1 = \partial p - \rho c \partial(\mathbf{u} \cdot \mathbf{n}) \tag{5.2.3}$$

$$\partial w_2 = \partial s, \tag{5.2.4}$$

are set to zero. Here, s is the entropy, given as

$$s = c_v * ln\left(\frac{p}{\rho^\gamma}\right), \tag{5.2.5}$$

where γ is the ratio of the specific heats c_p and c_v, where c_p and c_v are the specific heats at constant pressure and volume, respectively [7]. Equations (5.2.3) and (5.2.4) yield the conditions

$$p_{F_1}^n = p_{F_2}^{n-1} + (n_x(u_1^n - u_2^n) + n_y(v_1^n - v_2^n))\sqrt{(\gamma p_{F_2}^{n-1} \rho_{F_2}^{n-1})} \tag{5.2.6}$$

$$\rho_{F_1}^n = \rho_{F_2}^{n-1}(p_{F_1}^n / p_{F_2}^{n-1})^{1/\gamma}, \tag{5.2.7}$$

for the emerging fluid point density and pressure. $p_{F_1}^n$, determined by equation (5.2.6), is used in equation (5.2.7). The derivations of equations (5.2.6) and (5.2.7) are given in appendix A.

5.2.2 Velocity

The velocity components u and v are set by one of the following methods:

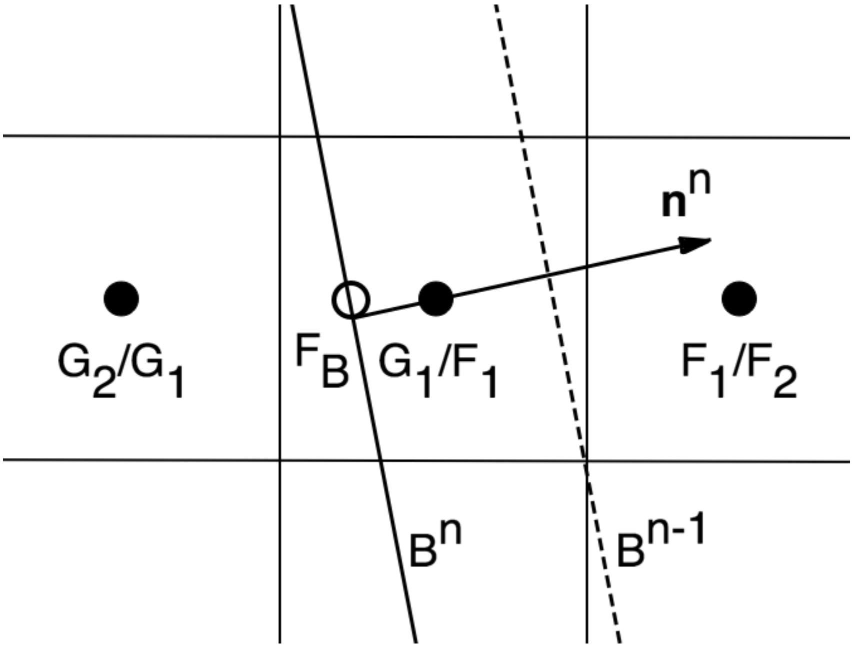

Figure 5.3: Illustration of the concept of emerging fluid points, showing the boundary B at two different times, $n-1$ and n, three data points and their respective notations at the different times, as well as the point F_B, which is in the fluid domain, infinitely close to the boundary.

A: The first method, the simple method based on current velocity, assumes that the emerging fluid point is positioned infinitely close to the moving boundary, at point F_B, as shown by Fig. 5.3. The normal velocity component $(\mathbf{u}\cdot\mathbf{n})_{F_1}$ is set equal to the normal velocity component of the boundary $(\mathbf{u}\cdot\mathbf{n})_B$, while the tangential velocity component $(\mathbf{u}\cdot\mathbf{t})_{F_1}$ is set equal to the preliminary tangential velocity component, thus neglecting the velocity change over the distance of which the wall has surpassed the newly emerged fluid point, as shown in Fig. 5.2. The preliminary velocity, $\mathbf{u}' = [u',\ v']^T$, is set as

$$\begin{bmatrix} u' \\ v' \end{bmatrix}_{F_1}^{n} = \begin{bmatrix} u \\ v \end{bmatrix}_{G_1}^{n-1}, \tag{5.2.8}$$

using the existing velocities of the point in question, set by the ghost point treatment after the last stage, or the only stage, of the time stepping method.

These conditions lead to a linear system of equations, yielding the conditions

$$\begin{aligned} u_{F_1} &= \big[n_y u'_{F_1} - n_x v'_{F_1}\big]n_y + \big[n_x u_B + n_y v_B\big]n_x \\ v_{F_1} &= \big[-n_y u'_{F_1} + n_x v'_{F_1}\big]n_x + \big[n_x u_B + n_y v_B\big]n_y \end{aligned} \tag{5.2.9}$$

for the velocity components u and v of the emerging fluid point. The derivation of equation (5.2.9) is given in appendix A.

B: The second method, the simple method based on the neighbouring velocity, also assumes that the emerging fluid point is positioned infinitely close to the moving boundary, at point F_B, as shown by Fig. 5.3. The velocity is set as in A, by equation (5.2.9), but in this case the preliminary velocity is set as

$$\begin{bmatrix} u' \\ v' \end{bmatrix}^{n}_{F_1} = \begin{bmatrix} u \\ v \end{bmatrix}^{n-1}_{F_1}, \tag{5.2.10}$$

using the existing velocity components of the neighbouring point, resulting from the solution from the internal solver in the previous time step.

C: The third method, the interpolating method based on current velocity, takes the velocity change over the distance by which the wall has surpassed the emerging fluid point into account. The velocity is set by linearly interpolating between the adjacent fluid point F_2 and a point in the fluid domain infinitely close to the boundary, F_B, yielding a more accurate approximation.

First, the preliminary velocity $\mathbf{u}'$ is set based on the current values at the point, as in equation (5.2.8). Next, the velocity components of the fluid infinitely close to the boundary, u_{F_B} and v_{F_B}, are determined as

$$\begin{aligned} u_{F_B} &= \big[n_y u'_{F_1} - n_x v'_{F_1}\big] n_y + \big[n_x u_B + n_y v_B\big] n_x \\ v_{F_B} &= \big[-n_y u'_{F_1} + n_x v'_{F_1}\big] n_y + \big[n_x u_B + n_y v_B\big] n_y \end{aligned} \tag{5.2.11}$$

The velocity of the emerging fluid point is then determined as

$$\mathbf{u}_{F_1} = \mathbf{u}_{F_B} + (\mathbf{u}_{F_2} - \mathbf{u}_{F_B})\frac{\delta_{B,F_1}}{\delta_{B,F_2}}, \tag{5.2.12}$$

where δ_{B,F_1} is the distance between the boundary and the emerging fluid point F_1, and δ_{B,F_2} is the distance between the boundary and the neighbouring fluid point, F_2.

D: The fourth method, the interpolating method based on neighbouring velocity, also takes the velocity change over the distance by which the wall has surpassed the emerging fluid point into account. The velocity is determined in the same way as above, in method C, but the preliminary velocity is set as in method B, thus combining the equations (5.2.10), (5.2.11) and (5.2.12).

Chapter 6

Numerical examples

To verify that the method is solving the correct equations, and show that the method is convergent, two examples are given here. The performance of the method is evaluated in terms of accuracy and computation time required.

6.1 Retracted piston

The first two-dimensional example presented by Tan and Shu [5] to demonstrate their method is a two-dimensional simplification of a piston being retracted to the left in a tube of uniform cross-sectional area, as illustrated by Fig. 6.1, with a fixed wall at $x = 1$. The volume of the closed space between the left moving piston and the right wall increases, decreasing the density of the confined gas. The piston starts at $x = 0.5$ at $t = 0$, and moves with velocity $u_B = -0.5\cos(t)$. The ratio of the specific heats of the gas, γ, is set to 1.4.

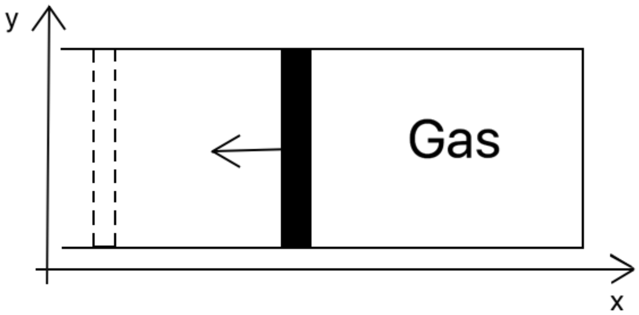

Figure 6.1: Sketch showing the piston and the gas filled compartment.

The domain $x = [0,\ 1]$ is divided into $ni \times nj$ cells, such that the grid spacings Δx and Δy are given by $\Delta x = 1/ni$ and $\Delta y = 1/nj$, respectively. Here, $ni = nj$. This example does not involve any complex boundaries, and the arbitrary way the boundary intercepts the Cartesian grid does not pose any major problems, as the normal surface vector of the moving boundary is always $[1\ 0]^T$.

The initial conditions are defined as

$$\begin{aligned}
\rho(x,y,0) &= 1 + 0.2\cos(2\pi(x-0.5)) + 0.1\cos(2\pi(y-0.5)), && (6.1.1)\\
u(x,y,0) &= x-1, && (6.1.2)\\
v(x,y,0) &= y(y-1)\cos(\pi x), && (6.1.3)\\
p(x,y,0) &= \rho(x,y,0)^{\gamma}, && (6.1.4)
\end{aligned}$$

thus introducing some non-uniformity in the y-direction. All quantities are assumed to be dimensionless.

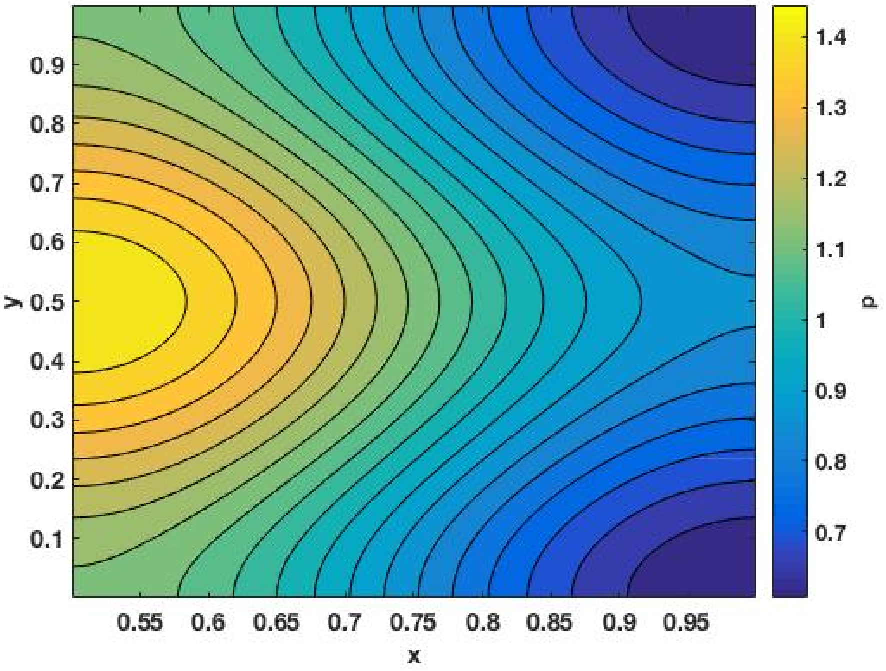

Figure 6.2: Initial pressure p for all fluid points with $nj = 640$, as given by equation (6.1.4).

The CFL-number, as defined by equations (3.5.5), (3.5.6) and (3.5.7), is set to 0.6, so that the results are comparable to the results of Tan and Shu [5], unless otherwise specified.

All combinations of time marching methods, internal solvers, and ghost point and emerging fluid point methods have been tested to determine which ones were the most promising, and two combinations, one using the MUSCL, and one not using the MUSCL, have been thoroughly analysed. The combinations are given in Table 6.6. The simpler approximations yield the best results, except for ghost point velocity, which is surprising, but in line with the results obtained for one dimensional examples in the project work [6]. The choice of method setting the ghost point velocity has the biggest impact on the errors analysed.

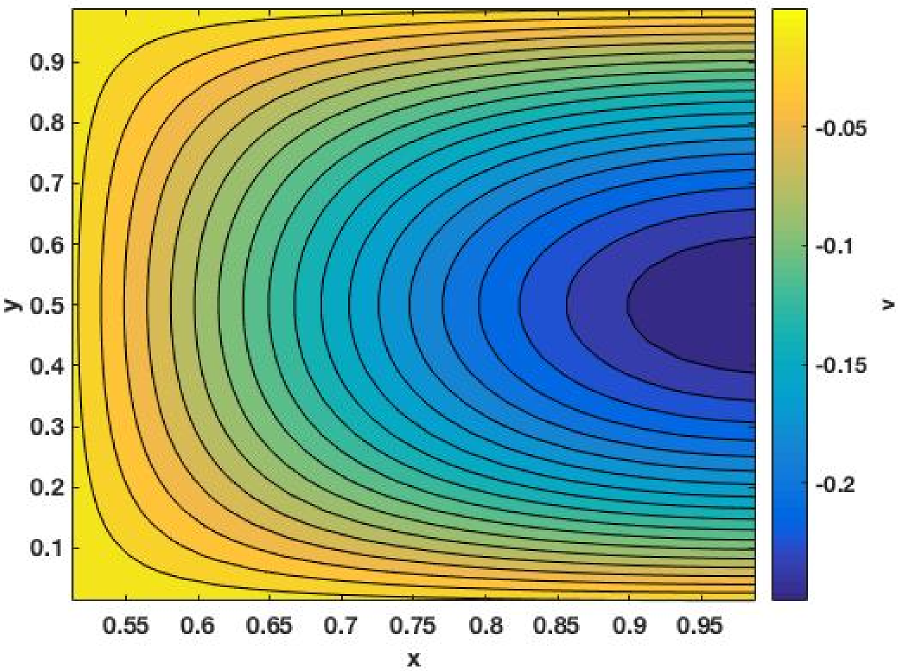

Figure 6.3: Initial velocity v for all fluid points with $nj = 640$,as given by equation (6.1.3).

Spatial scheme	w/MUSCL	wo/MUSCL
Time stepping method	TVD RK3	Explicit Euler
ρ and p of ghost points	I	I
Velocity of ghost points	b	c
ρ and p of emerging fluid points	2	1
Velocity of emerging fluid points	A	A

Table 6.1: The combination of methods using MUSCL that performs the best in terms of entropy and mass conservation error and rate of convergence, and the best combination not using MUSCL.

The resulting primitive variables, the density ρ, the velocity components u and v, and the pressure p, of all the fluid points, at time $t = 0.5$ are shown by Figs. 6.4, 6.5, 6.6 and 6.7, respectively. The x-component of the velocity u is equal to the velocity of the piston near the piston and zero close to the stationary wall, in line with the boundary condition, as given by equation (3.4.1). The initial velocity component in y-direction v is negative everywhere, as shown by equation (6.1.3) and Fig. 6.6, but with a higher magnitude near the stationary wall on the right, driving a clockwise rotation. This rotation is counteracted in the area near the lower left corner by the local maximum of the initial pressure p close to the piston, shown by Fig. 6.2. The same maximum drives the clockwise rotation near the upper left corner, causing pressure rise, which again causes some local positive u values around time $t = 0.25$. The velocity component u is non-uniform in y-direction due to this rotation. The density ρ and pressure p is also higher in the lower right corner due to this initial v. Further, the denisty ρ and the pressure p is lower on the left side, close to the piston, because the piston is moving to the left, increasing the volume of the fluid domain.

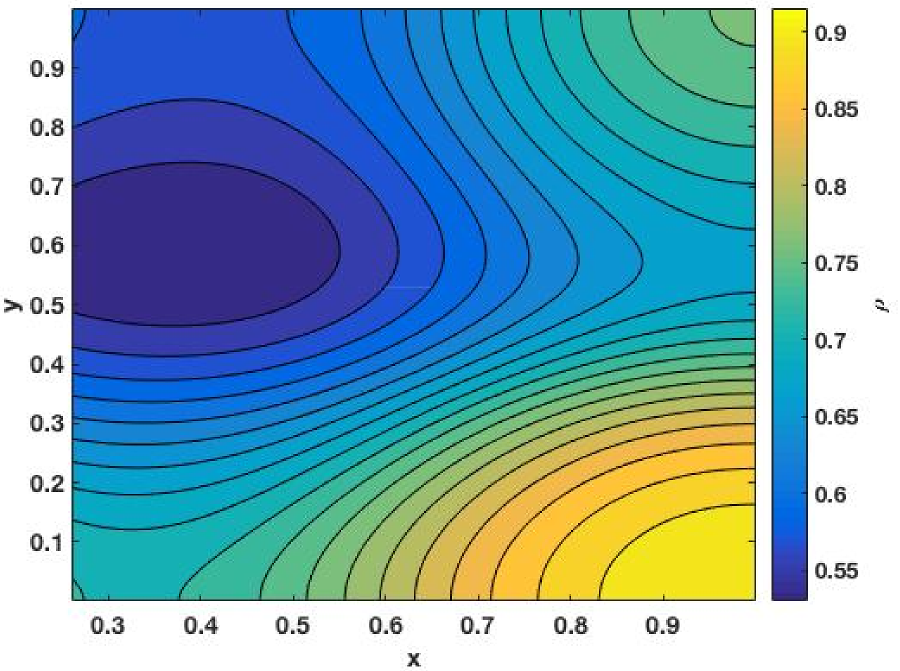

Figure 6.4: Density ρ for all fluid points at time $t = 0.5$ with $nj = 640$, obtained using the combination of methods with MUSCL.

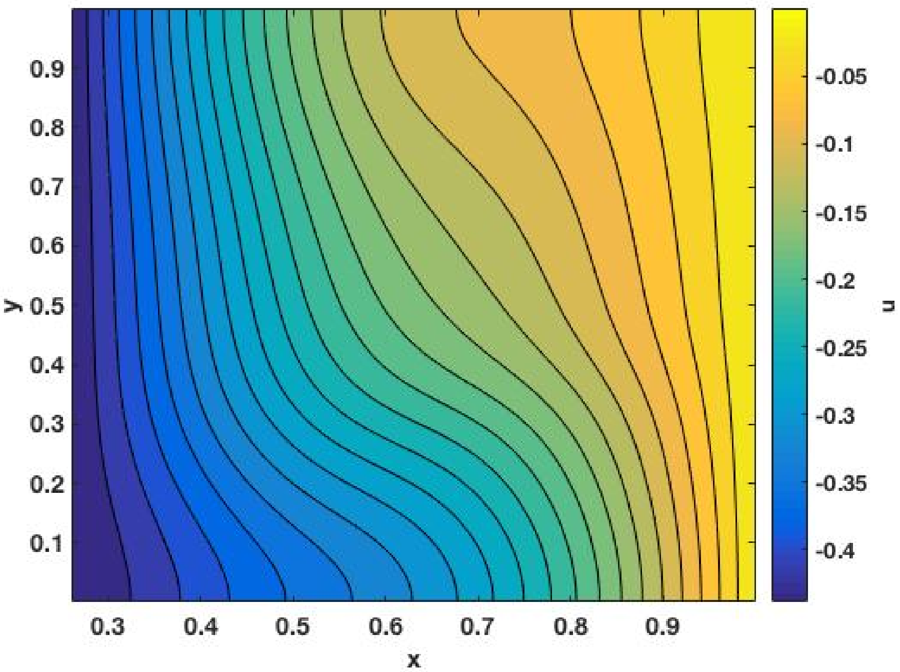

Figure 6.5: Velocity u for all fluid points at time $t = 0.5$ with $nj = 640$, obtained using the combination of methods with MUSCL.

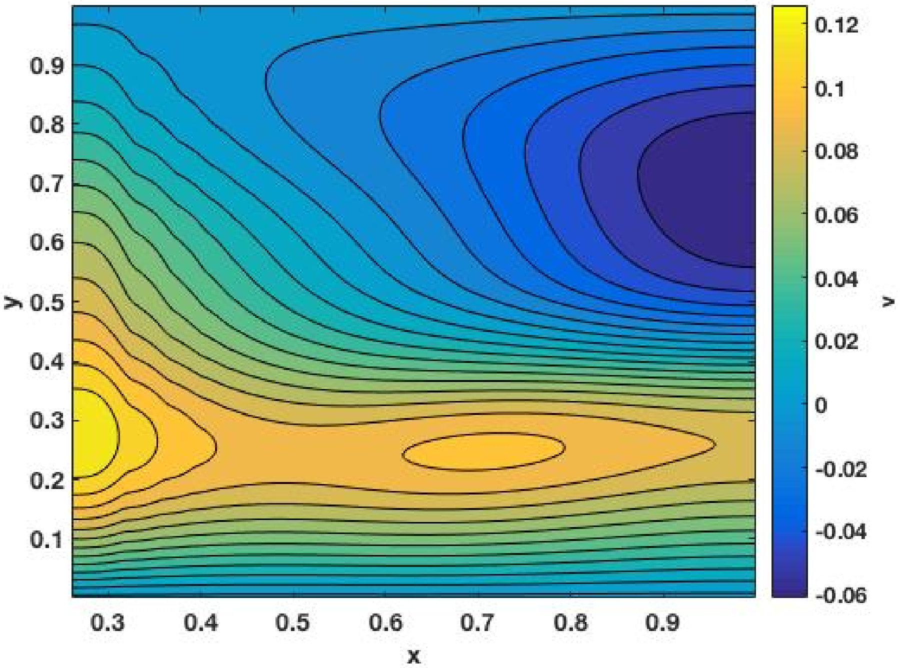

Figure 6.6: Velocity v for all fluid points at time $t = 0.5$ with $nj = 640$.

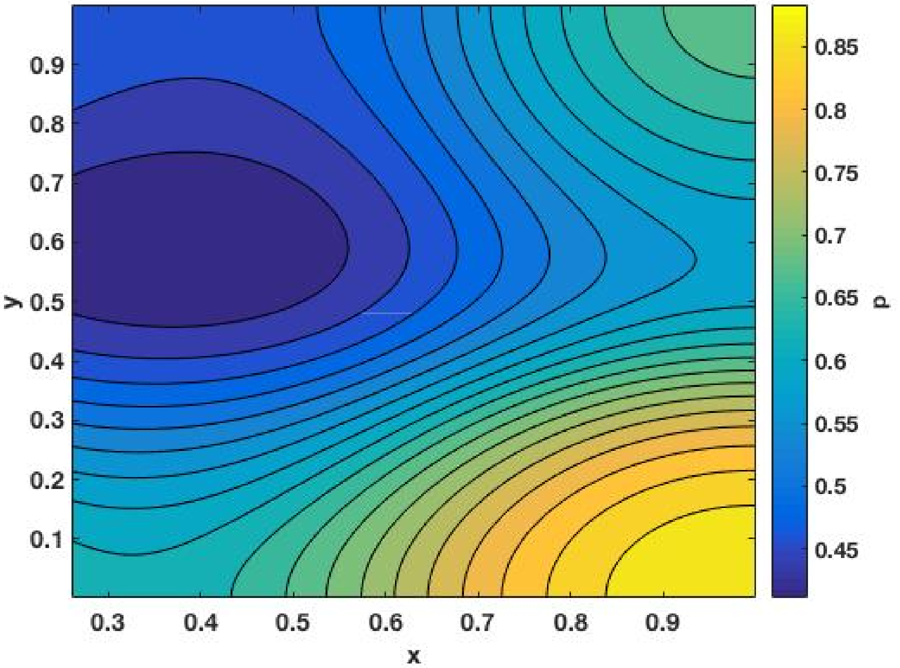

Figure 6.7: Pressure p for all fluid points at time $t = 0.5$ with $nj = 640$.

The minimum speed of sound, as given by equation (2.1.16), in this example is $c = 1.05$, remarkably larger than the maximum absolute velocity encountered, $\mathbf{u} = 0.6$, thus confirming that the subsonic flow assumption is valid.

The highest stable CFL-numbers of the different combinations of methods are given in Table 6.2. As expected, the TVD RK3 time stepping method is stable for slightly bigger time steps than the explicit Euler method, and the spatial scheme not using MUSCL is stable for slightly bigger time steps than the method employing it.

Spatial scheme	Temporal scheme	Highest stable CFL-number
wo/MUSCL	TVD RK3	1.4
	Explicit Euler	1.1
w/MUSCL	TVD RK3	1.3
	Explicit Euler	1.0

Table 6.2: Stable CFL-numbers of the combinations of spatial schemes and time stepping methods.

6.1.1 Entropy error

The entropy is expected to stay constant as the flow is inviscid, and no shocks occur. An entropy-related term, s', is also, for the exact solution of the problem, staying constant. The initial conditions are chosen such that the initial magnitude of the entropy-related term is one, such that the relative entropy error is equal to the absolute entropy error. The deviation is, as in [5], monitored to measure the convergence of the method. The entropy s is given as

$$s = c_v * ln\Big(\frac{p}{\rho^\gamma}\Big), \tag{6.1.5}$$

where c_v is the specific heat at constant volume [7]. The entropy-related term s' is given as

$$s' = \frac{p}{\rho^\gamma}, \tag{6.1.6}$$

and their relation is given by

$$s' = e^{\frac{s}{c_v}} \; . \tag{6.1.7}$$

The error is determined as the norm of a vector containing the entropy-related error of all points in the fluid domain, given as

$$||s' - s'_0||_2 = \sqrt{\Delta x \Delta y \sum_{j=1}^{nj} \sum_{i=i_{F1}}^{ni} \Big(\frac{p_{i,j}}{\rho_{i,j}^\gamma} - 1\Big)^2}. \tag{6.1.8}$$

However, as the error is given by Tan and Shu [5] as the 1-norm, given as

$$||s' - s'_0||_1 = \Delta x \Delta y \sum_{j=1}^{nj} \sum_{i=i_{F1}}^{ni} \Big|\frac{p_{i,j}}{\rho_{i,j}^\gamma} - 1\Big|, \tag{6.1.9}$$

and the infinity norm, given as

$$||s' - s'_0||_\infty = max_{i\in[F1 \;\; ni],j\in[1 \;\; nj]}\Big(\Big|\frac{p_{i,j}}{\rho_{i,j}^\gamma} - 1\Big|\Big), \tag{6.1.10}$$

these are also determined.

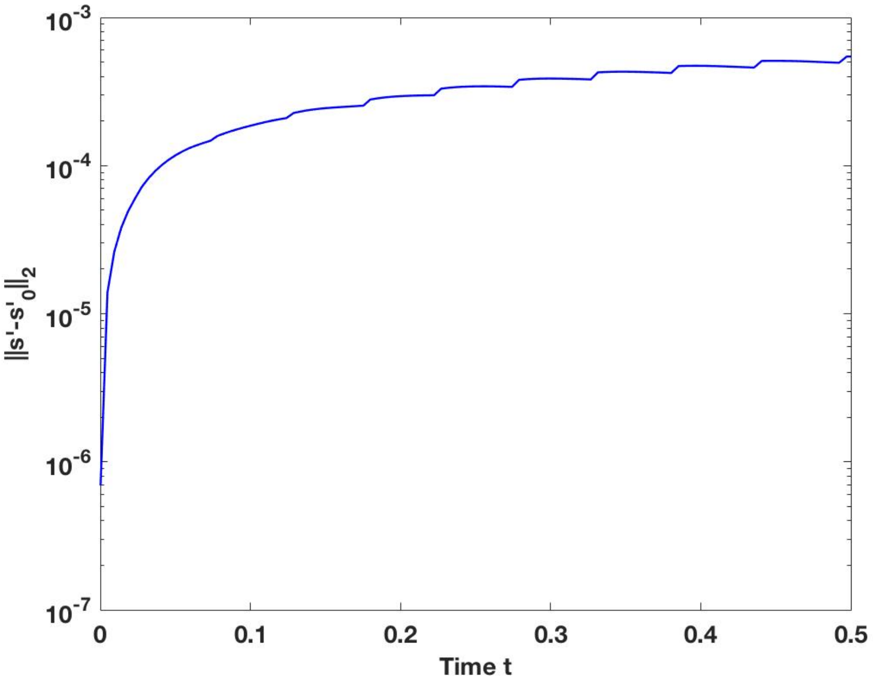

Figure 6.8: Entropy error $||s' - s'_0||_2$, as given by equation (6.1.8), with MUSCL and $nj = 40$.

Due to the errors given by equations (6.1.8) and (6.1.9) being area dependent, they seem to increase substantially when a new column of fluid points emerge, as shown by Fig. 6.11. This is due to an area error, as the total area of the fluid cells is different from the exact area of the fluid domain. The area of fluid cells closest to the boundary A_{F1} varies between $0.5A_{F_2}$ and $1.5A_{F_2}$, as illustrated by Figs. 6.9 and 6.10.

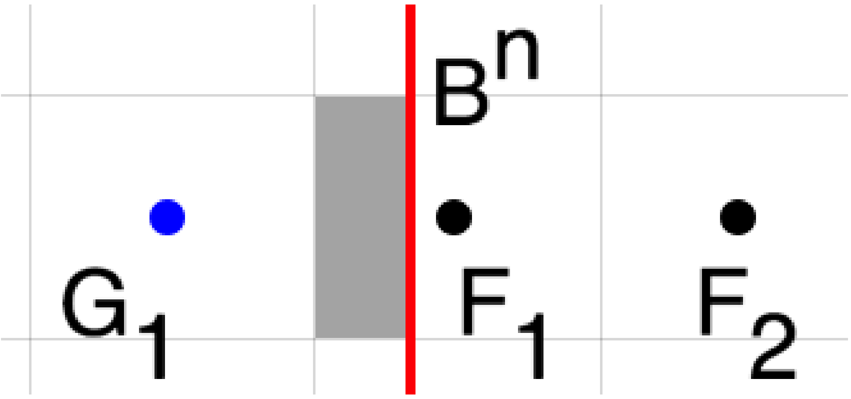

Figure 6.9: Illustration of the concept of the area error. The exact area of cell F_1 is smaller than that assumed by equations (6.1.8), (6.1.9) and (6.1.14). The grey area represents the deviation.

As this area error itself is not of particular interest here, correcting for it to reveal the remaining sources of entropy error is favourable. As determining the exact area of these cells is trivial, utilizing the exact area when determining the post-processed errors is possible. The post-processed errors $||s' - s'_0||_{2,p}$ and $||s' - s'_0||_{1,p}$ are given as

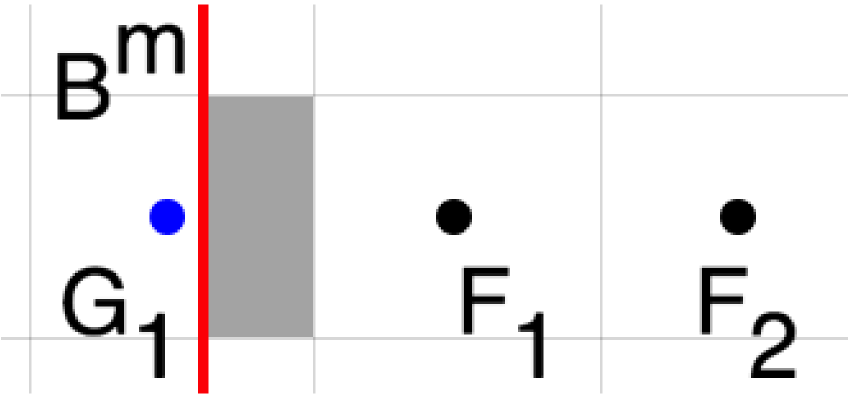

Figure 6.10: Illustration of the concept of the area error where the exact area of cell F_1 is larger than that assumed by equations (6.1.8), (6.1.9) and (6.1.14). The grey area represents the deviation.

$$||s' - s'_0||_2 = \sqrt{\Delta y \sum_{j=1}^{nj} \left(\Delta x \sum_{i=i_{F2}}^{ni} \left(\frac{p_{i,j}}{\rho_{i,j}^{\gamma}} - 1 \right)^2 + \left(\frac{\Delta x}{2} + x_{F1} - x_B \right) \left(\frac{p_{F1,j}}{\rho_{F1,j}^{\gamma}} - 1 \right)^2 \right)}, \tag{6.1.11}$$

and

$$||s' - s'_0||_1 = \Delta y \sum_{i=1}^{ni} \left(\Delta x \sum_{j=j_{F2}}^{nj} \left| \frac{p_{i,j}}{\rho_{i,j}^{\gamma}} - 1 \right| + (\frac{\Delta x}{2} + x_{F1} - x_B) \left| \frac{p_{i,j}}{\rho_{i,j}^{\gamma}} - 1 \right| \right), \tag{6.1.12}$$

respectively.

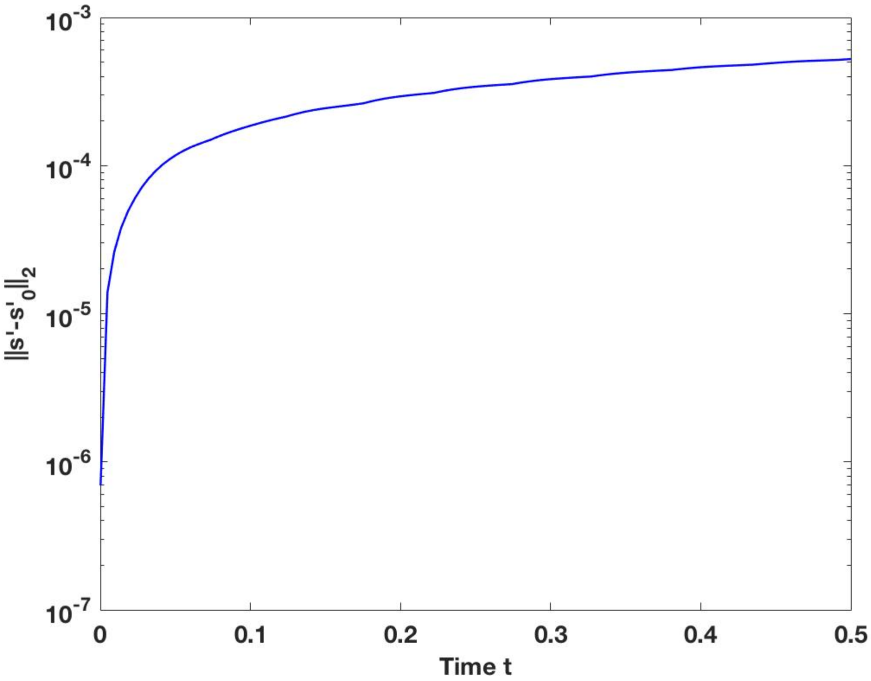

Figure 6.11: The post-processed entropy error $||s' - s'_0||_2$, as given by equation (6.1.11), with MUSCL and $nj = 40$.

As Fig. 6.11 shows, the post-processed entropy error $||s' - s'_0||_{2,p}$ increases smoothly with time. The sources of error include error due to the simplified approximation of the values at the ghost point points, truncation error resulting from the discretization in space and time, including numerical diffusion.

The order of convergence given in Tables 6.3 and 6.4 is given as

$$P = \frac{\left(\frac{err(\Delta x)}{err\left(\frac{\Delta x}{n}\right)}\right)}{ln(n)}, \tag{6.1.13}$$

where P is the order of convergence, err is the error with a given grid spacing, and $1/n$ is the factor by which the grid spacing changes from one grid to the next, for which the order of convergence is determined.

	Third order [5]		wo/MUSCL		w/MUSCL	
nj	$\|\|s' - s'_0\|\|_1$	P	$\|\|s' - s'_0\|\|_1$	P	$\|\|s' - s'_0\|\|_1$	P
80	2.50E-08		1.82E-03		9.02E-05	
160	1.10E-09	4.50	9.40E-04	0.95	2.42E-05	1.90
320	9.70E-11	3.50	4.80E-04	0.97	6.60E-06	1.87
640	9.87E-12	3.30	2.42E-04	0.99	1.73E-06	1.93

Table 6.3: Post-processed entropy errors $||s' - s'_0||_1$ and corresponding convergence rates given by equations (6.1.12) and (6.1.13), respectively, for the simplified methods with and without the MUSCL and for Tan and Shu's third order method [5], for comparison.

The post-processed entropy error $||s' - s'_0||_1$ is lower for the results with MUSCL than without, but higher than for the results obtained by Tan and Shu [5], as expected. Further, the convergence rates are very close to the expected limits of two and one, with and without MUSCL, respectively, as shown in Table 6.3.

	Third order [5]		wo/MUSCL		w/MUSCL	
nj	$\|\|s' - s'_0\|\|_\infty$	P	$\|\|s' - s'_0\|\|_\infty$	P	$\|\|s' - s'_0\|\|_\infty$	P
80	3.28E-07		5.42E-03		9.35E-04	
160	3.06E-08	3.42	3.16E-03	0.78	4.38E-04	1.09
320	6.17E-09	2.31	1.44E-03	1.13	1.65E-04	1.41
640	7.06E-10	3.13	8.96E-04	0.69	8.42E-05	0.97

Table 6.4: Post-processed entropy errors $||s' - s'_0||_\infty$ and corresponding convergence rates given by equations (6.1.10) and (6.1.13) for the simplified methods with and without the MUSCL and for Tan and Shu's third order method [5], for comparison.

The maximum entropy error, as given by equation (6.1.10), is located in the internal part of the fluid domain in initial part of the simulation. For $t > 0.15$, it is located in the newly emerged fluid point in the upper left corner, probably due to large gradients in that area around $t = 0.15$. As this is still true at the time $t_{\text{end}} = 0.5$, as shown by Fig. 6.12, the error can be analyzed to determine the accuracy of the boundary treatment.

As Table 6.4 shows, the convergence rate of the error is not as high as expected. The lower than expected convergence rate may be due to the errors embedded in the approximations of the ghost point values, thus a limitation of the method , assuming that the most accurate ghost and emerging fluid point treatments possible within the constraints of the method are used here, which is probably not really the case, but not very far from it. Changing the method, by introducing a more sophisticated ghost point treatment more similar to the method developed by Sjögreen and Petersson [1], as discussed in section 1.2, might increase the order of accuracy of the boundary treatment.

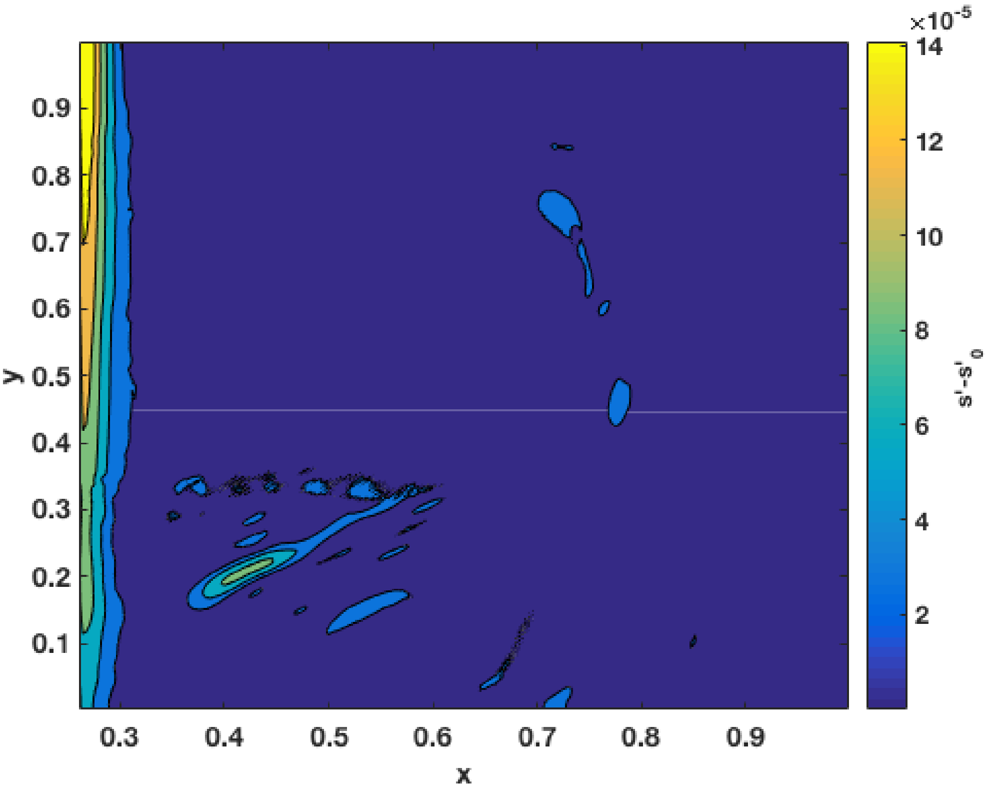

Figure 6.12: The entropy error $s' - s'_0$ for all the fluid cells at the time $t = 0.5$ for $nj = 320$ and $CFL = 1.2$ using the method with MUSCL as given by Table 6.1.

As Fig. 6.13 shows, the post-processed entropy error tends towards zero as the grid spacings Δx and Δy tend towards zero, showing that the method is consistent in terms of entropy error.

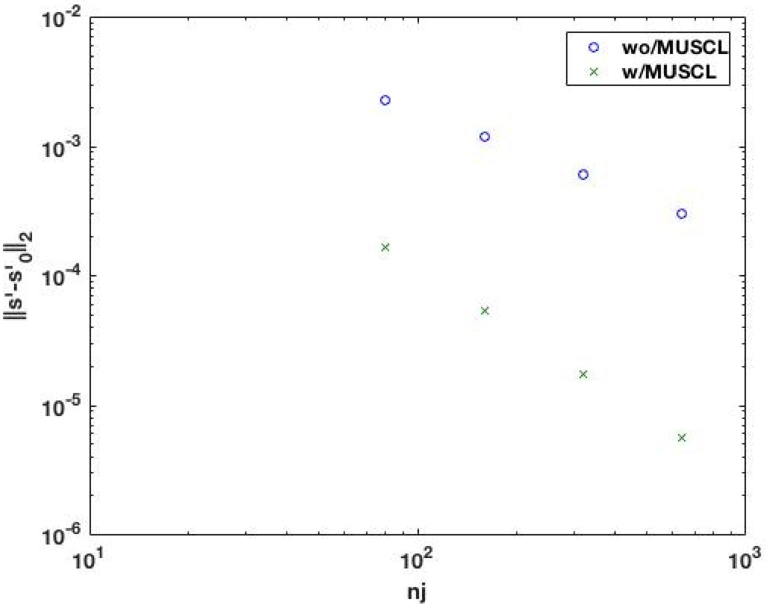

Figure 6.13: The post-processed entropy error $||s' - s_0'||_2$, determined at the time $t = 0.5$ using two different combinations of methods, for different grid spacings Δx and Δy.

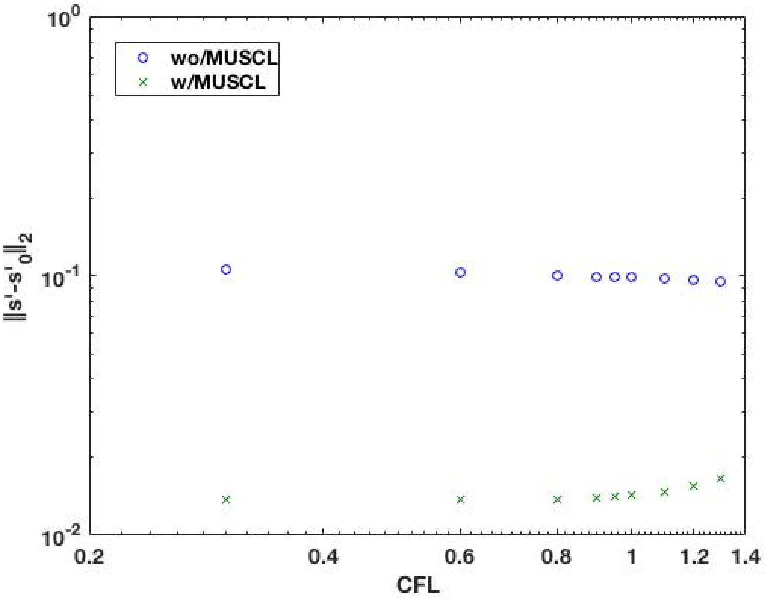

Figure 6.14: The post-processed entropy error $||s' - s_0'||_2$ for different CFL-numbers, determined at the time $t = 0.5$ for $nj = 40$.

The simulations performed to identify the combinations of methods yielding the smallest errors were performed with $CFL = 0.6$. However, as Fig. 6.14 shows, the effect of changing the CFL-number on the entropy error is insignificant, thus setting the CFL-number higher is beneficial, as the number of time steps is lower for higher CFL-numbers. Further, this implies that the numerical error in the spatial discretizarion is dominant, as changing the size of the time steps show little impact.

6.1.2 Mass conservation error

As the boundaries are modelled as impermeable and the mass flux across them is zero, the mass of the system is constant. However, as mentioned in section 1.2, the Cartesian grid method is known not to conserve mass. The mass is determined as

$$m = \Delta y \Delta x \sum_{j=1}^{nj} \sum_{i=i_{F1}}^{ni} \rho_{i,j}, \tag{6.1.14}$$

and relative mass conservation error is given as $(m-m_0)/m_0$ where $m_0 = 0.5$ is the initial mass of the system. As Fig. 6.15 shows, the error is oscillatory, primarily due to the area of fluid cells adjacent to the boundary having a variable area that is assumed to be constant, as discussed in section 6.1.1. By introducing a post-processed mass conservation error, taking the exact area of the fluid cells into account, given as $(m - m_0)/m_0$, where m is given as

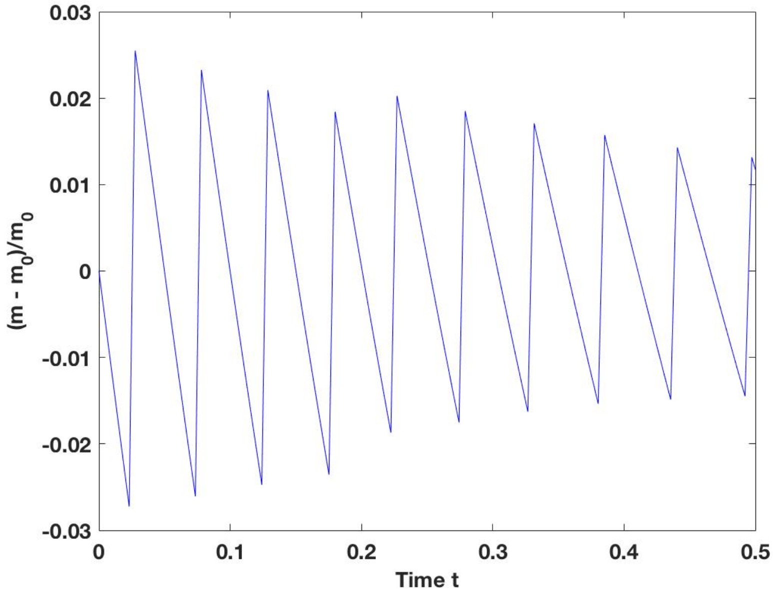

Figure 6.15: Mass conservation error $(m - m_0)/m_0$, as given by equation (6.1.14), with MUSCL and $nj = 40$.

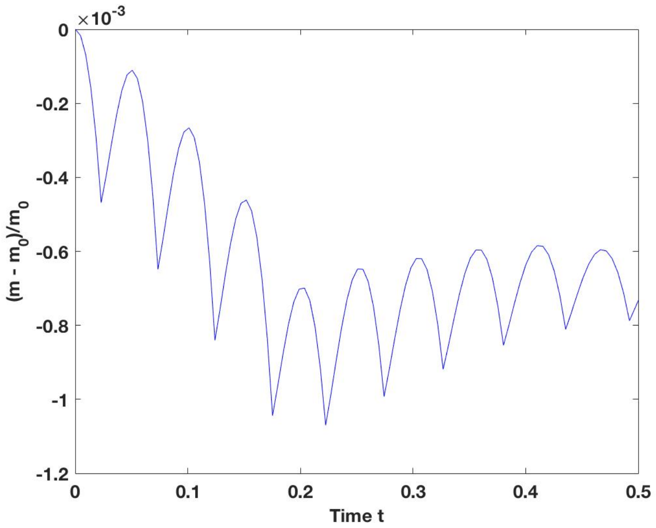

Figure 6.16: The post-processed mass conservation error $(m{-}m_0)/m_0$ as given by equation (6.1.15) with MUSCL and $nj = 40$.

$$m = \Delta y \sum_{j=1}^{nj} \Big(\Delta x \sum_{i=i_{F2}}^{ni} \rho_{i,j} + \big(\frac{\Delta x}{2} + x_{F1} - x_B\big)\rho_{F1,j} \Big), \tag{6.1.15}$$

the error is reduced by one order of magnitude, as shown by Figs. 6.15 and 6.16. The post-processed mass conservation error has a characteristic, periodic shape, shown by Fig. 6.16. The mass of the system, as given by equation (6.1.15), is largest when the area of the fluid cell adjacent to the moving boundary is equal to the area of the other fluid cells. The mass increases when the area is smaller, and decreases when the area is larger, and the rate of change appears to be proportional to the deviation in area. The total mass is a result of the density and the area of the fluid cells, and the density results from the numerical flux terms ρu and ρv, as equations (2.1.9) and (2.1.10) shows. The flux terms at the boundary results from the density and the velocity of the cells adjacent to the boundary, the ghost cells and the fluid cells. The ghost point velocity method used, specified in Table 6.1, assumes the boundary to be located the same distance from the ghost point and the adjacent fluid point, as given by equation (5.1.4), which is true when the area of the fluid cell adjacent to the boundary is equal to the area of the other fluid cells. When the area is very small, in the beginning of every cycle, right after the boundary has overstepped a ghost point, making it a newly emerged fluid point, the velocity of the ghost point is set too high, as the velocity of the boundary is negative. The flux term ρu at the left boundary of the fluid cell adjacent to the boundary, which should be zero, is larger than zero, making the mass increase. When the area is too large, at the end of the cycle, the velocity is set too low, causing a negative flux term, which again causes a decrease in the mass.

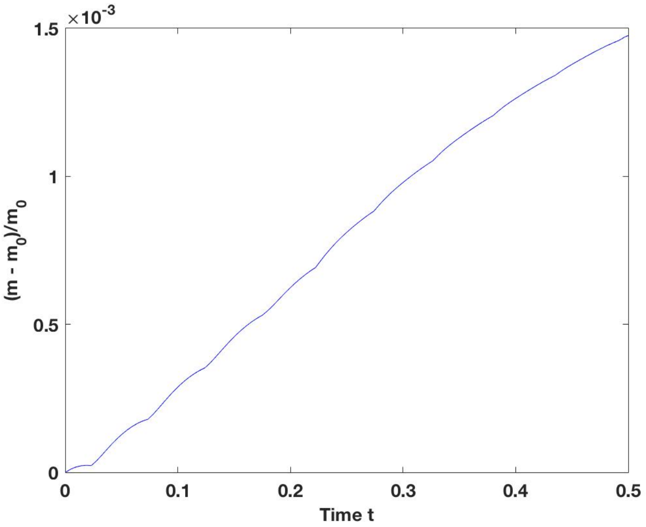

Figure 6.17: The post-processed mass conservation error $(m-m_0)/m_0$ as given by equation (6.1.14) with the mirror point ghost point velocity method, c, with MUSCL and $nj = 40$.

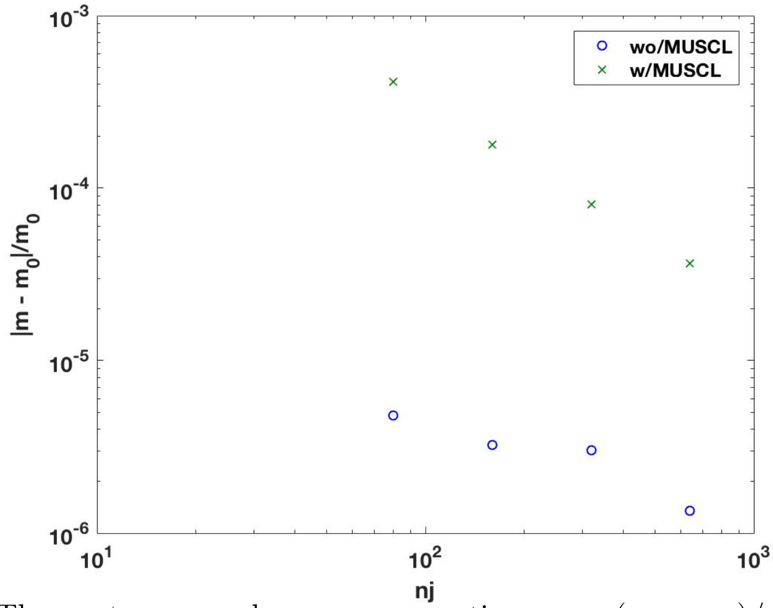

Figure 6.18: The post-processed mass conservation error $(m - m_0)/m_0$, determined at the time $t = 0.5$ using two different combinations of methods, as given in Table 6.1, for different grid spacings.

As Fig. 6.18 shows, the post-processed mass conservation error is significantly smaller without MUSCL than with MUSCL, while the convergence rate is higher with MUSCL. This is unexpected, implying that the method without MUSCL is more accurate than the one with MUSCL.

When employing a different method of setting the velocity of the ghost points, method c, based on the mirror points, as given by equations (5.1.5) and (5.1.6), the mass conservation error is slightly larger. However, the characteristic shape has disappeared, as shown in Fig. 6.17, as expected.

As Fig. 6.18 shows, the post-processed mass conservation error tends towards zero as the grid spacings Δx and Δy tend towards zero, implying that the method is consistent also in terms of the mass conservation error.

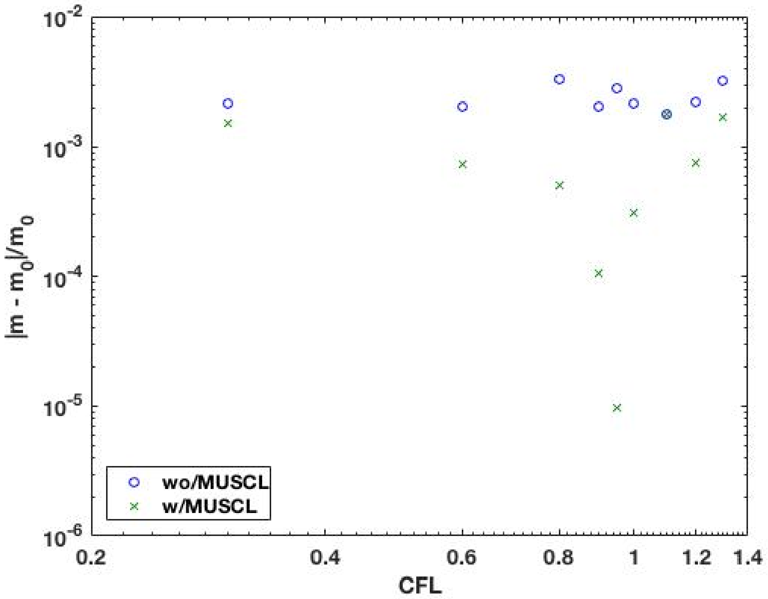

Figure 6.19: The post-processed mass conservation error $(m - m_0)/m_0$ for different CFL-numbers, determined at the time $t = 0.5$ for $nj = 40$.

As Fig. 6.35 shows, the change in post-processed mass conservation error $(m-m_0)/m_0$ when increasing the CFL-number seems significant for the method using MUSCL. However, as Fig. 6.20 shows, the interesting data point is coincidental, and does not imply that the result is close to the exact solution. This compromises the results given in Fig. 6.35, as the resulting mass conservation errors are all coincidental.

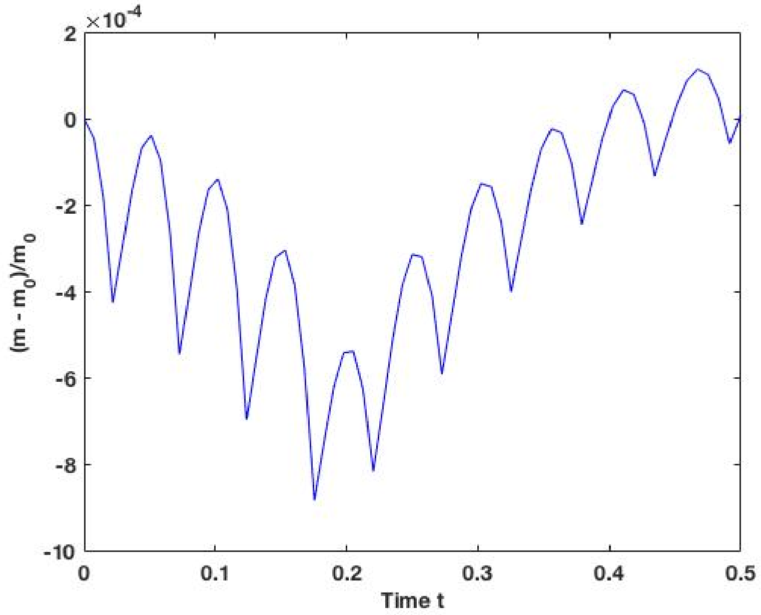

Figure 6.20: The post-processed mass conservation error $(m - m_0)/m_0$ for $CFL = 0.95$ at time $t = [0\ 0.4]$ for $nj = 60$. The magnitude of the error is varying a lot, and by coincidence, the end time error is very small.

However, by recording the maximum mass conservation error obtained, more informative and reliable results may be obtained, shown in Fig. 6.21. This shows that the CFL-number has a low impact on the post-processed mass conservation error, thus the error resulting from the spatial discretization is dominant, in line with the entropy error. As the entropy error and mass conservation error are indicators of how close the result is to the exact solution, the fact that they behave similarly in terms of grid refinement and changes in the CFL-number is promising.

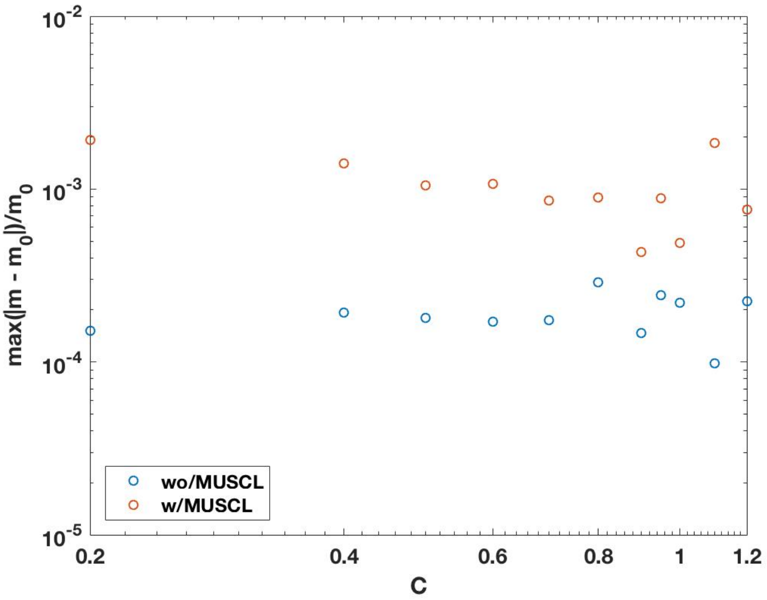

Figure 6.21: The maximum post-processed mass conservation error $(m - m_0)/m_0$ for different CFL-numbers, for $nj = 40$.

6.1.3 Computational time

The simulations are performed using MATLAB 2016a on an Apple MacBook Pro ”Core i5” 2.7 13” produced early in 2015, which, as the name suggests, has a 2.7 GHz dual-core processor.

For a constant CFL-number, the time step, as given by equations (3.5.5), (3.5.6) and (3.5.7), is inversely proportional to the number of grid points in each direction, $\Delta t \propto 1/nj$. Thus, the number of time steps required to reach a certain time t_{end} is proportional to the number of grid points in each direction nj. Further, the number of operation required per time step is linearly dependent of the number of grid points, nj^2. The complexity of the method is therefore $\mathcal{O}(n^3)$, where $n = nj$. The computational time is given as $w\mathcal{O}(n^3)$, where w is a constant resulting from the number of floating point operations of the method per grid point and the computational power of the computer. If the method is improved by increasing the order of accuracy of the boundary treatment and/or the internal solver, w increases. This is beneficial when more accurate solutions are required, as smaller grids will be sufficient, but not when less accurate solutions are sufficient. For two methods of the same complexity, one with a larger w and a larger convergence rate than the other, there will be a ”break-even” point in terms of grid size. For larger grids the former yields better accuracy per computational effort, and for smaller grids, the latter will be advantageous.

ni x nj	80x80	160x160	320x320	640x640
wo/MUSCL	2.73	15.97	87.27	573.18
w/MUSCL	12.53	63.35	571.78	5288.65
wo/MUSCL w/TVD RK3	8.80	28.42	159.41	1462.73

Table 6.5: CPU time required for the different combinations of methods, measured using the tic-toc functionality in MATLAB. Note that, as the methods w/MUSCL and wo/MUSCL use different time stepping methods, another method, wo/MUSCL, but with TVD RK3, not inluded in Table 6.1, has been included here.

The computational time required is expected to increase by a factor of 8 when the number of grid points in each direction is doubled, for every step to the right in Table 6.5. That does not match very well with the resulting computational times given in Table 6.5. The trend is that the increase is smaller for the smaller grid sizes, and slightly larger for the last step. This discrepancy can be explained by the computational times of the smaller grids being dominated by operations that require the same amount of computational time independently of the grid size, such as function calls. Further, the method using TVD RK3 requires between two and three times as much computational time as the one using the explicit Euler method, which is expected, since it involves three stages, cf. section 3.5.

The method with MUSCL and TVD RK3 requires substantially more computational time for the larger grids than the method without MUSCL with TVD RK3, about three times as much for the larger grids. As the computational time of the method with MUSCL and TVD RK3 is eight times higher than for the method without MUSCL with the explicit Euler method, comparing the results from the method with MUSCL with the results from the method without MUSCL using a grid with twice as many grid points in each direction can be justified.

As the goal of any numerical method is to achieve a result as accurately as possible per computational effort and memory required, comparing the methods under the constraint of equal computational time, when optimal CFL-numbers are used, is beneficial. As Table. 6.5 shows, the computational time of the method without MUSCL, with the explicit Euler method, for $nj = 640$ is equal to the one for the method with MUSCL, for $nj = 320$. As Tables 6.3 and 6.4 and Figs. 6.13 and 6.18 shows, the method with MUSCL yields far better accuracy.

With optimal CFL-number, as given in Table 6.2, similar results are obtained. The size of the time steps increase proportionally to the CFL-numbers, decreasing the computational times, and the magnitude of the 2-norm of the entropy error and the mass conservation error do not change significantly.

By analyzing the method using the profile timing functionality in MATLAB, it is revealed that less than 10 % of the computational time is spent on the boundary treatment. For the method performing the best in terms of accuracy per computational time, less than 3 % of the computational time is spent on the boundary treatment, and the relative amount decreases for increasing grid size. Further, as the convergence rates of the infinity norm, as given in Table. 6.4, are below the limitations of two and one, for the method with MUSCL and without MUSCL, respectively, it is likely that the boundary treatment is the limitation of the method.

It is obvious from thorough testing that some combinations of ghost point treatments yield far better results, thus implementing ghost point treatments that yield more accurate values at the ghost points should be beneficial for the accuracy of the method. Therefore, it would probably be advantageous for the performance of the method to utilize a more sophisticated ghost point treatment.

6.2 Moving cylinder

The next example presented by Tan and Shu [5] is a two dimensional simplification of a rigid cylinder, a circle, moving in an enclosed square with rigid walls, as illustrated by Fig. 6.22. The circle has a radius $R = 1$, and the walls at the boundaries of the computational domain are positioned at $x = \pm 4$ and $y = \pm 4$. The domain is divided into $ni \times nj$ cells, such that the grid spacings Δx and Δy are given by $\Delta x = 8/ni$ and $\Delta y = 8/nj$, respectively. Here, $ni = nj$ for all calculations. This example is expected to be more challenging for the method, as the boundary intercepts the Cartesian grid in an arbitrary way.

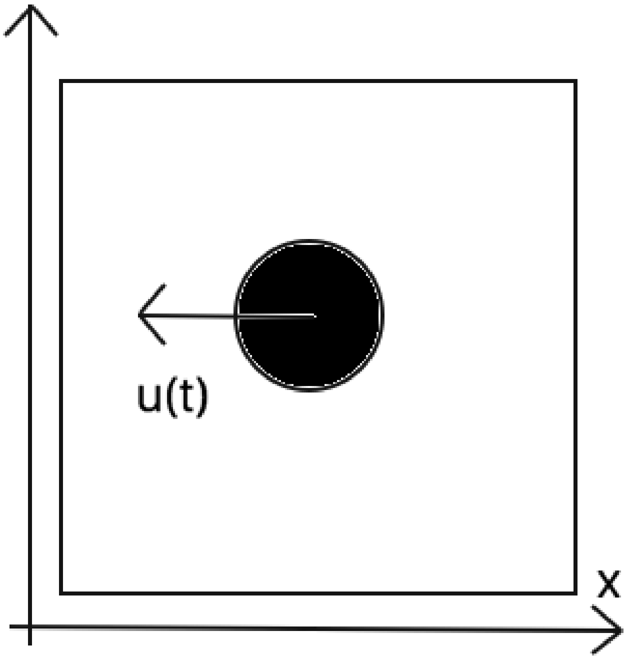

Figure 6.22: Illustration of the moving cylinder.

The cylinder moves horizontally with variant velocity, as the position of the centre of the cylinder is given as $\mathbf{x}_c(t) = (-0.5\sin t, 0)$. The ratio of the specific heats of the gas, γ, is set to 1.4. The initial conditions are given as

$$\begin{aligned} \rho(x,y,0) &= 1, \\ u(x,y,0) &= -0.5\tilde{u}(x,y), \\ v(x,y,0) &= 0.5\tilde{v}(x,y), \\ p(x,y,0) &= 1, \end{aligned} \tag{6.2.1}$$

where

$$\tilde{u}(x,y) = \lambda_1(x,y)u_1(x,y) + \lambda_2(x,y)^2, \tag{6.2.2}$$

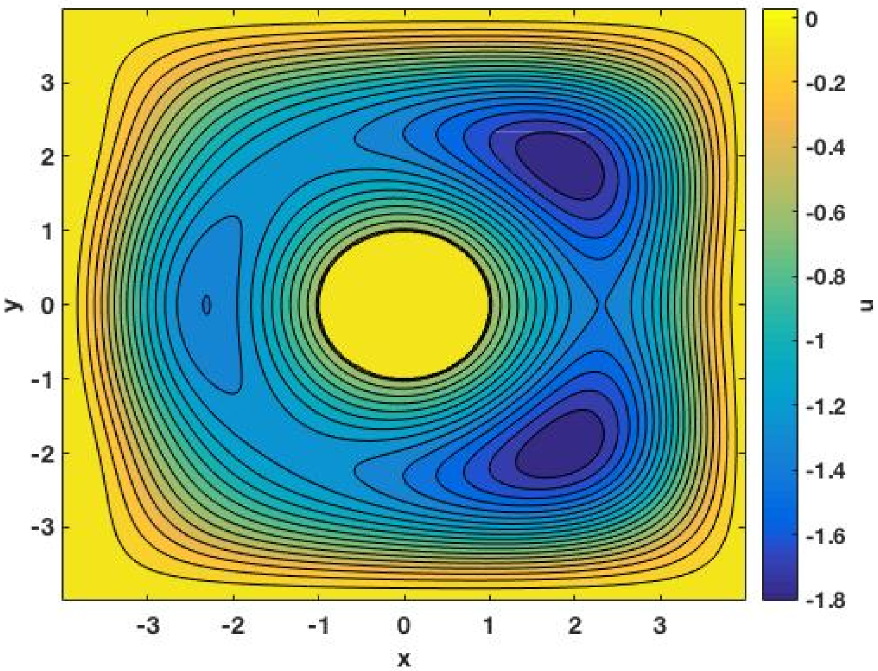

Figure 6.23: Initial velocity component u for all fluid points with $nj = 640$.

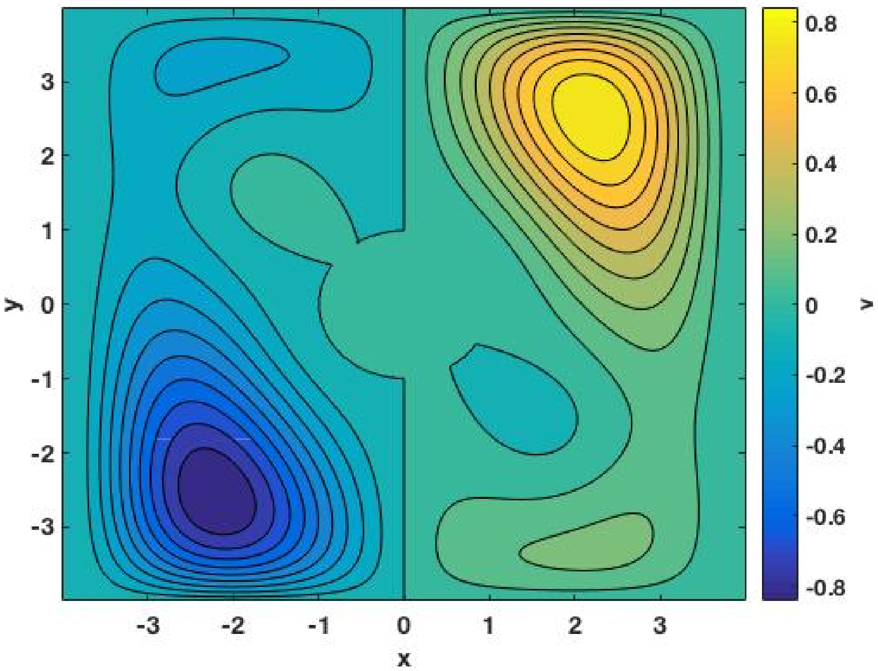

Figure 6.24: Initial velocity component v for all fluid points with $nj = 640$.

where

$$\begin{aligned} \lambda_1(x,y) &= \frac{\big(4\sqrt{2}-1\big)\big(\sqrt{x^2+y^2}-1\big)}{\big(\sqrt{16+y^2}-1\big)\big(\sqrt{16+x^2}-1\big)}, \\ \lambda_2(x,y) &= \frac{\sqrt{x^2+y^2}\big(x^2-16\big)\big(y^2-16\big)}{\big(\frac{x^2}{x^2+y^2}-16\big)\big(\frac{y^2}{x^2+y^2}-16\big)}, \qquad (6.2.3) \\ u_1(x,y) &= \sin\big(\frac{\pi}{4}x\big)\sin^2\big(\frac{\pi}{4}y\big), \end{aligned}$$

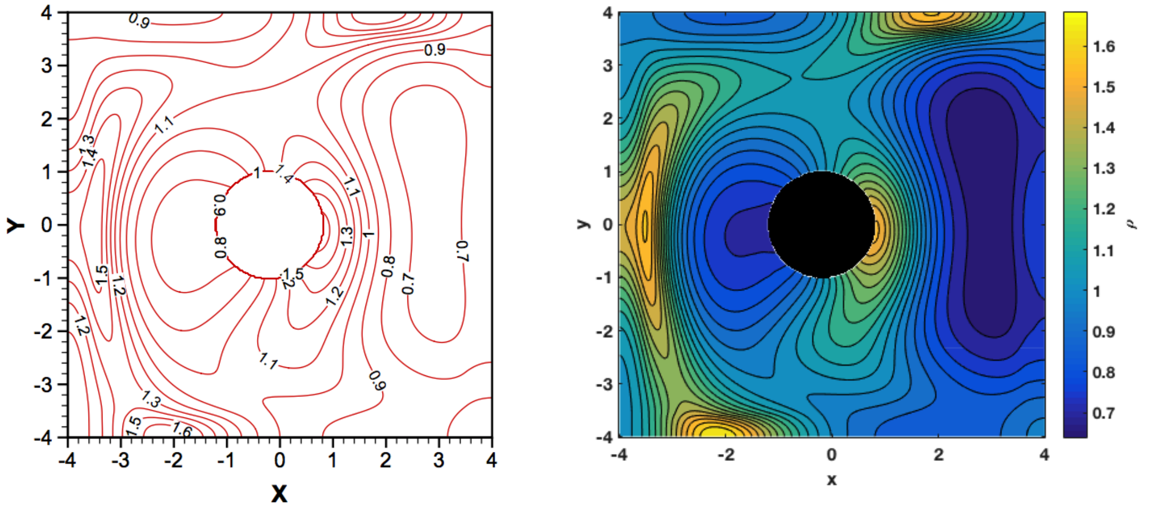

Figure 6.25: Density contours from Tan and Shu [5] on the left, and the simplified method with MUSCL on the right, $\Delta x = \Delta y = 1/40, t = 0.4$.

and

$$\tilde{v}_1(x,y) = \lambda_1(x,y)v_1(x,y) + \lambda_2(x,y)v_2(x,y), \tag{6.2.4}$$

where

$$\begin{aligned} v_1(x,y) &= \sin^2\left(\frac{\pi}{4}x\right)\sin\left(\frac{\pi}{4}y\right), \\ v_2(x,y) &= \frac{1}{16}(x^2+y^2-1)\sin\left(\frac{\pi}{4}x\right) \end{aligned} \tag{6.2.5}$$

which is consistent with the boundary conditions given in equation (3.4.1). Also here all quantities are assumed to be dimensionless.

All combinations of time marching methods, internal solvers, and ghost point and emerging fluid point methods have been tested to determine which ones were the most promising, and two combinations, one using the MUSCL, and one not using the MUSCL, have been thoroughly analysed. The combinations are given in Table 6.6. The CFL-number is set to 0.6 for the results to be comparable with those presented by Tan and Shu [5], unless otherwise specified.

Spatial scheme	w/MUSCL	wo/MUSCL
Time stepping method	TVDRK3	Explicit Euler
ρ and p of ghost points	I	I
Velocity of ghost points	c	c
ρ and p of emerging fluid points	3	1
Velocity of emerging fluid points	A	C

Table 6.6: The combination of methods using MUSCL that performs the best in terms of entropy and mass conservation error and rate of convergence, and the best combination not using MUSCL.

It is evident from Fig. 6.25 that the result determined by the simplified method is visually identical to the result presented by Tan and Shu [5], indicating that the simplified method is solving the correct set of equations.

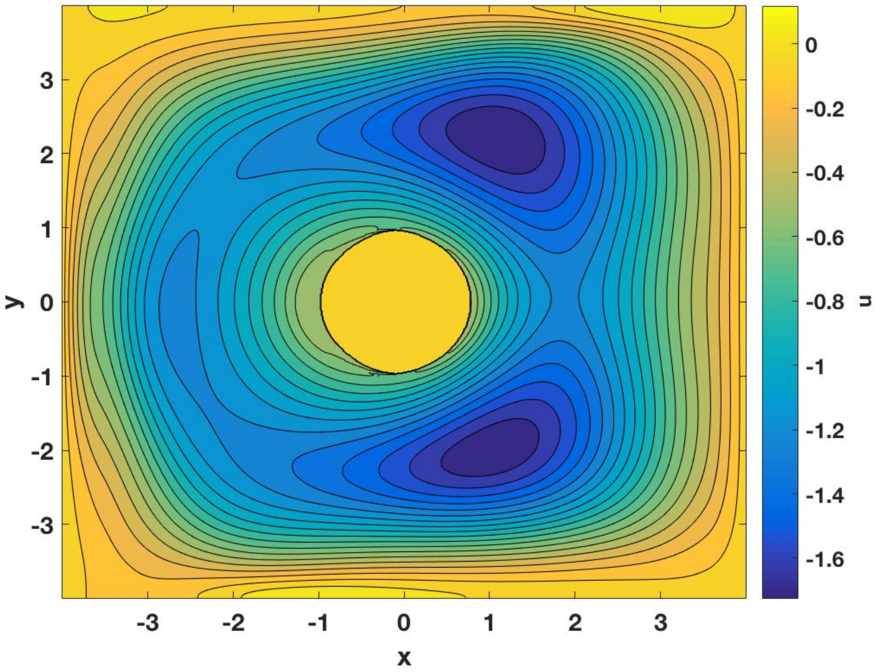

Figure 6.26: Velocity component u for all fluid points with $nj = 640$ at time $t = 0.4$, obtained using the combination of methods with MUSCL.

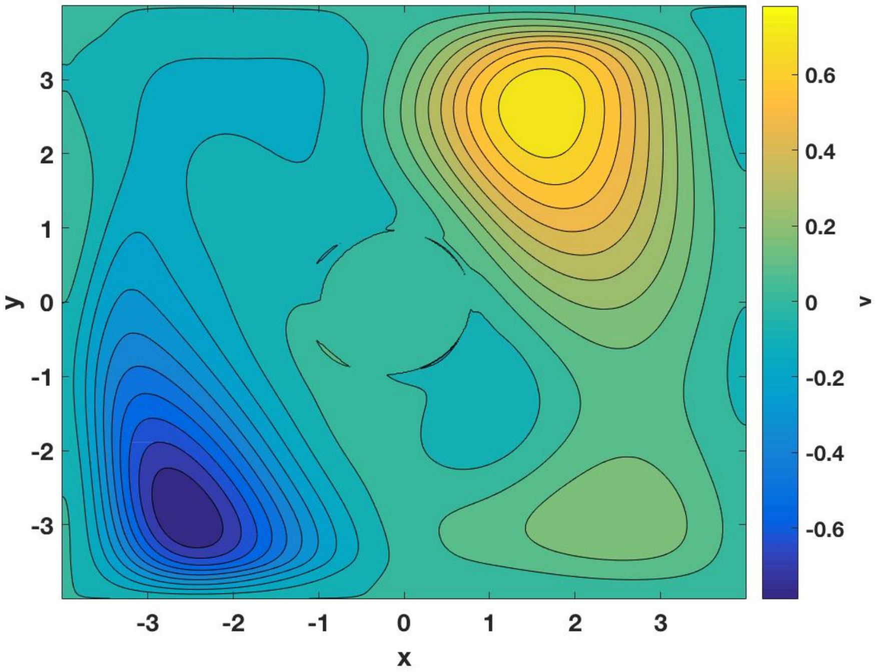

Figure 6.27: Velocity component v for all fluid points with $nj = 640$ at time $t = 0.4$, obtained using the combination of methods with MUSCL.

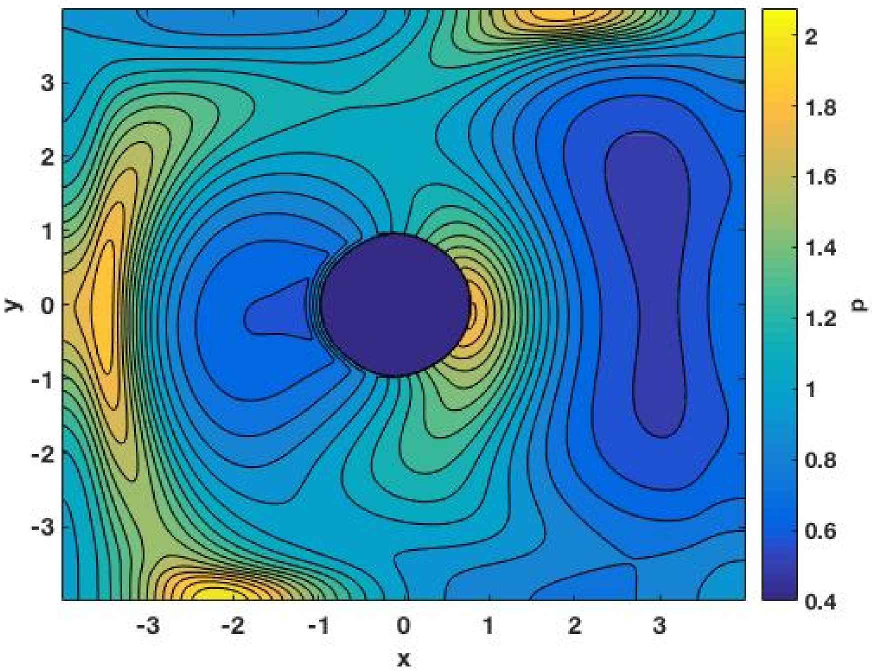

Figure 6.28: Pressure p for all fluid points with $nj = 640$ at time $t = 0.4$, obtained using the combination of methods with MUSCL.

The remaining primitive variables, the velocity components u and v, and the pressure p, of all the fluid points at time $t = 0.4$ are shown by Figs. 6.26, 6.27 and 6.28, respectively.

By intuition, the resulting pressure and density contours do not look correct at first, especially in the close vicinity of the cylinder. However, by studying the initial velocity, shown by Figs. 6.23 and 6.24, it can be explained. The initial velocity component u very close to the surface of the cylinder is equal to its initial velocity. Further, the velocity component u is all negative. The x-component of the velocity of the cylinder is given as $-0.5\cos t$, which means that it slows down, making the gas behind it, to the right, catch up with it, causing a rise in density and pressure on the right side. The initial value of the y-component of the velocity, v, is positive near the upper right corner, and negative near the lower left corner. As these get closer and closer to the wall, the density and pressure increase in front of them, retarding them slightly. The two minima of the velocity component u are simply transported to the left by advection, affected only slightly by the presence of the cylinder. The initial velocity component u is also negative on the left side of the cylinder. This part of the gas is slowed down by the rigid wall, resulting in an increase in pressure and density which causes a shock later on, at $t > 0.4$. The initial density and pressure is uniform, thus not interesting.

When it comes to robustness in terms of stable CFL-numbers, very similar results to the ones given in table 6.2 are obtained for this example. The minimum speed of sound, as given by equation (2.1.16), in this example is $c = 1.08$, smaller than the maximum absolute velocity, as shown by Fig. 6.23. However, this does not cause any trouble, as it is not on the boundary of the computational domain. Further, this shows that the method is robust enough to handle supersonic flow.

6.2.1 Entropy error

The initial conditions are also here chosen such that initially, the entropy related term $s' = 1$. The deviation of the entropy is utilized to measure the relative error developing over time. The area-correction of the entropy error done as post-processing for the previous example is not done here, as the exact area of the fluid cells close to the boundary is more complicated to determine, and because it is not necessary to show the consistency of the method. Therefore, the entropy errors $||s' - s'_0||_2$, $||s' - s'_0||_1$ and $||s' - s'_0||_\infty$, as given by equations (6.1.8), (6.1.9) and (6.1.10), respectively, are used.

	Third order [5]		wo/MUSCL		w/MUSCL	
nj	$\|\|s' - s'_0\|\|_1$	P	$\|\|s' - s'_0\|\|_1$	P	$\|\|s' - s'_0\|\|_1$	P
40	2.81E-02		9.87E-01		1.70E-01	
80	2.80E-03	3.33	5.28E-01	0.90	4.23E-02	2.01
160	1.39E-04	4.33	2.74E-01	0.95	1.05E-02	2.02
320	3.79E-06	5.20	1.40E-01	0.97	2.60E-03	2.01
640	1.95E-07	4.28	7.06E-02	0.99	6.50E-04	2.00

Table 6.7: The entropy error $||s' - s'_0||_1$ at time $t = 0.4$ and corresponding convergence rates given by equations (6.1.9) and (6.1.13) for the simplified methods with and without the MUSCL, and for Tan and Shu's third order method [5] for comparison.

The entropy error $||s' - s'_0||_1$ is lower for the results with MUSCL than without, but higher than for the results obtained by Tan and Shu [5], as expected. Further, the convergence rates are very close to the expected limits of two and one, with and without MUSCL, respectively, as shown in Table 6.7.

	Third order [5]		wo/MUSCL		w/MUSCL	
nj	$\|\|s' - s'_0\|\|_\infty$	P	$\|\|s' - s'_0\|\|_\infty$	P	$\|\|s' - s'_0\|\|_\infty$	P
40	7.71E-03		5.21E-02		1.42E-02	
80	1.21E-03	2.67	3.20E-02	0.70	4.34E-03	1.71
160	1.08E-04	3.49	1.81E-02	0.82	1.92E-03	1.17
320	8.49E-06	3.67	9.69E-03	0.90	8.37E-04	1.20
640	1.48E-06	2.52	5.03E-03	0.95	4.08E-04	1.04

Table 6.8: The entropy error $||s' - s'_0||_\infty$ and corresponding convergence rates given by equations (6.1.10) and (6.1.13) for the simplified methods with and without the MUSCL and for Tan and Shu's third order method [5] for comparison.

As for the previous example, the maximum entropy error at time $t = t_{\text{end}} = 0.4$ is located in the close vicinity of the moving boundary, as shown by Fig. 6.29. Therefore, the infinity norm of the entropy error, $||s' - s'_0||_\infty$, can be analyzed to determine the order of the boundary treatment. As Table 6.8 shows, the convergence rate of the boundary treatment is not as high as expected. Therefore, as discussed in more detail in the previous example, the method might benefit from a more sophisticated ghost point treatment.

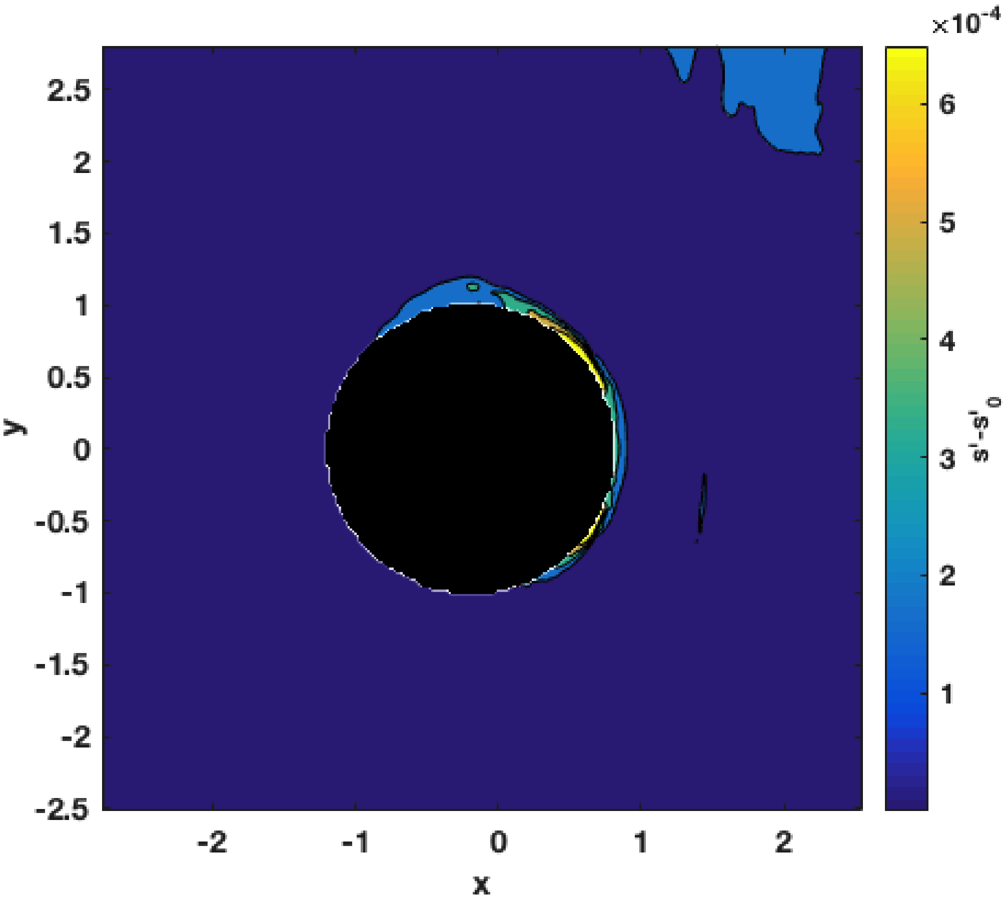

Figure 6.29: The entropy error $s' - s'_0$ for all the fluid cells at the time $t = 0.4$ for $nj = 320$ and $CFL = 1.0$ using the method with MUSCL, as given by Table 6.1.

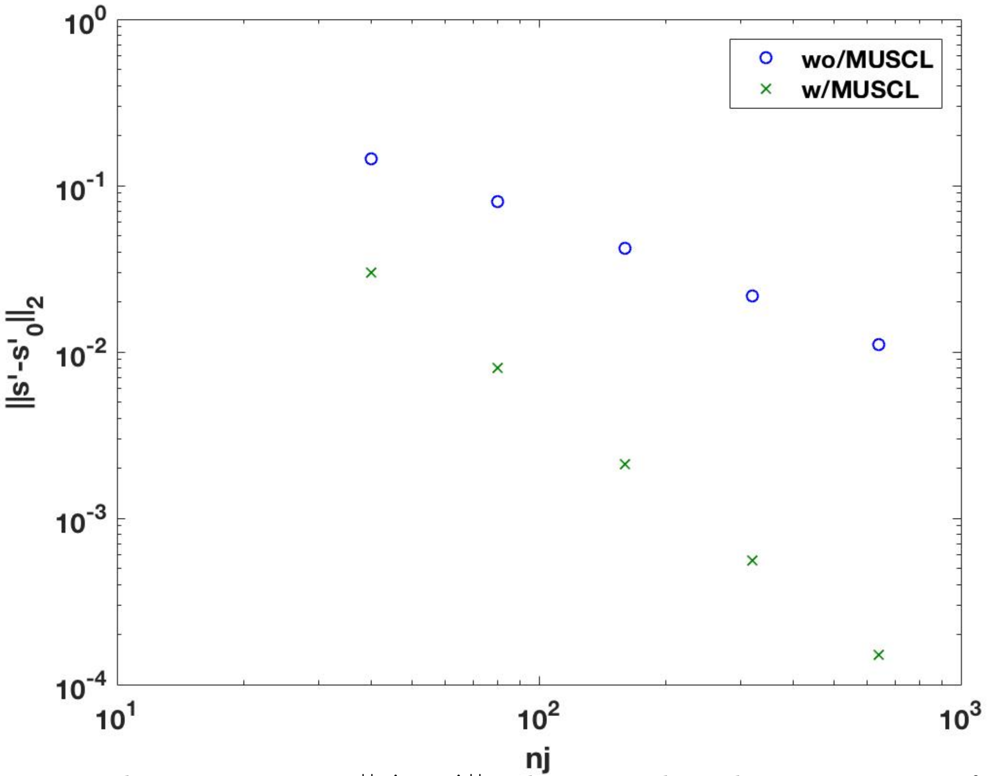

Figure 6.30: The entropy error $||s' - s'_0||_2$, determined at the time $t = 0.4$, for different numbers of grid points in each direction, nj.

As for the previous example, the resulting entropy error trend shown in Fig. 6.30 is as expected, showing that the method is consistent for both the method with and without MUSCL. As Fig. 6.30 and Tables 6.7 and 6.8 show, the method with MUSCL yields a higher accuracy and a higher order of convergence in terms of the entropy error, as expected.

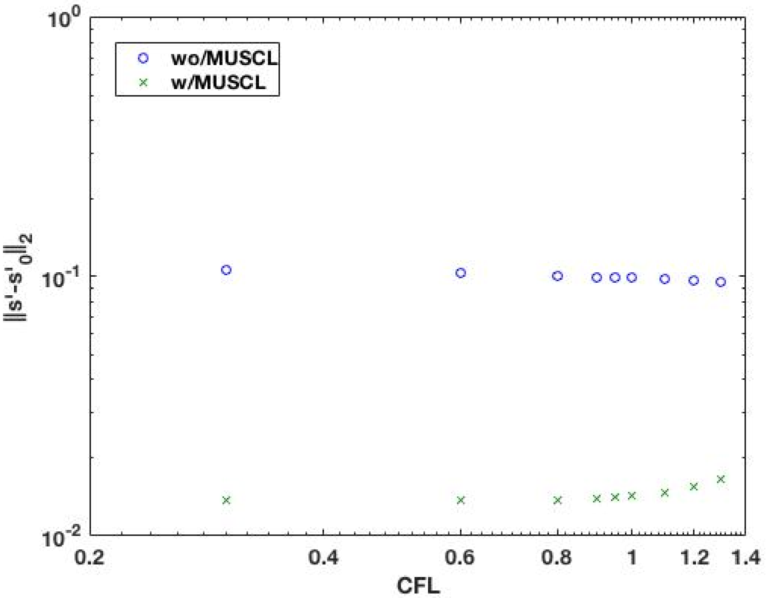

Figure 6.31: The entropy error $||s'-s'_0||_2$ for different CFL-numbers using the two methods specified in Table 6.6, for $nj = 60$. Other grid spacings yield similar results.

As Fig. 6.31 shows, the change in entropy error $||s' - s'_0||_2$ when increasing the CFL-number is insignificant. Thus setting the CFL-number as high as possible, as long as it is still stable, will contribute to achieving a solution as close to the exact solution per computational effort as possible.

6.2.2 Mass conservation error

As for the previous example, the mass does not remain constant, yielding a mass conservation error. The mass of the system is determined as

$$m = \Delta x \Delta y \sum_{i,j \in \text{Fluid points}} \rho_{i,j}, \tag{6.2.6}$$

and the relative mass conservation error is given as

$$(m - m_0)/m_0, \tag{6.2.7}$$

where $m_0 = 8^2 - \pi$ is the initial mass of the system.

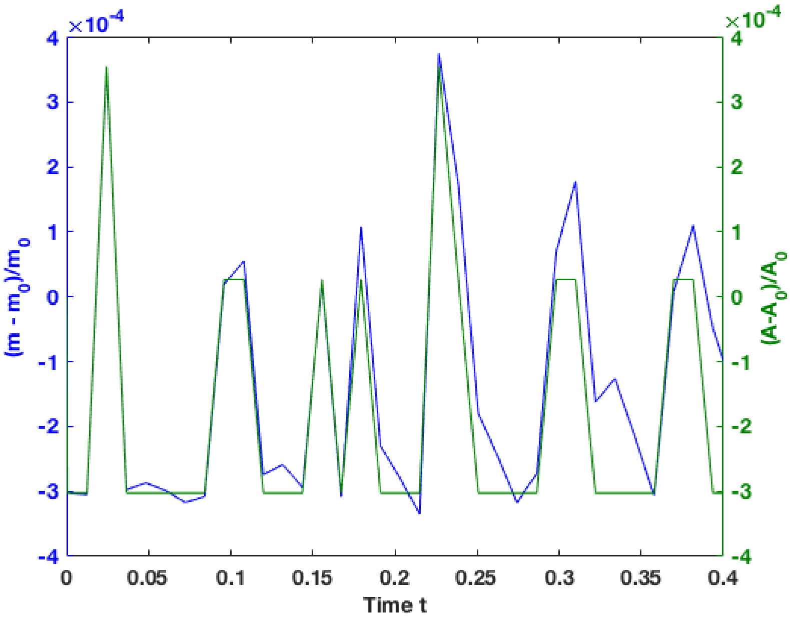

Figure 6.32: Mass conservation error and area error for $nj = 80$, wo/MUSCL.

As Fig. 6.32 shows, the mass conservation error varies a lot. However, it is closely correlated with the area error, the difference between the exact area of the fluid domain and the total area of all the fluid cells. The area error, given as $(A - A_0)/A_0$, where A is the total area of all the fluid cells, and $A_0 = 8^2 - \pi$ is the exact area of the computational domain, is illustrated by Fig. 6.33.

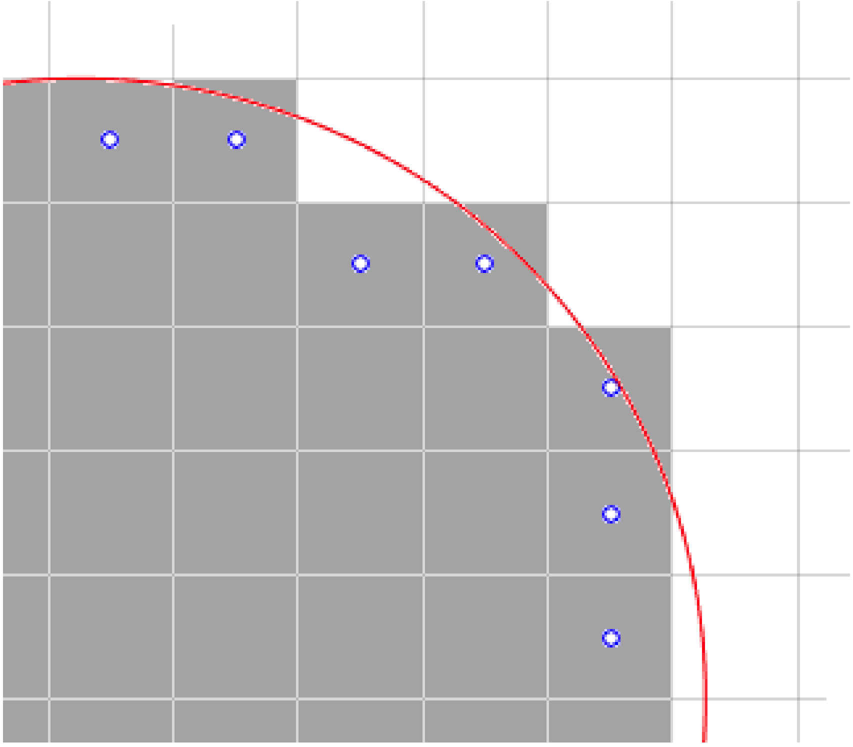

Figure 6.33: Illustration of the area error for a part of the moving boundary. The white area is the area of the fluid cells, and the exact area of the fluid domain is the area outside the red boundary. The difference between the total grey area outside the red line and the total white area inside the red line is the area error in this area.

The number of fluid cells is arbitrary, as the number of ghost cells turning into fluid cells and fluid cells turning into ghost cells is arbitrary, making the area error vary quite a bit, but yielding only discrete values, multiples of $\Delta x \Delta y$. The correlation between the area error and the mass conservation error, shown by Fig. 6.32, implies that the area error is the main source of mass conservation error. To remove the part of the error caused by the area error, a simple post-processing is introduced, given as

$$m = \Delta x \Delta y \sum_{i,j \in \text{Fluid points}} \rho_{i,j} + \overline{\rho_{G_1}}(A_{F_0} - A_F), \tag{6.2.8}$$

where $\overline{\rho_{G_1}}$ is the average density of the first layer ghost points, A_{F_0} is the exact area of the fluid domain, and A_F is the current area of the fluid domain. The average density of the first layer of ghost points is approximately equal to the average density of the fluid points adjacent to the surface of the cylinder.

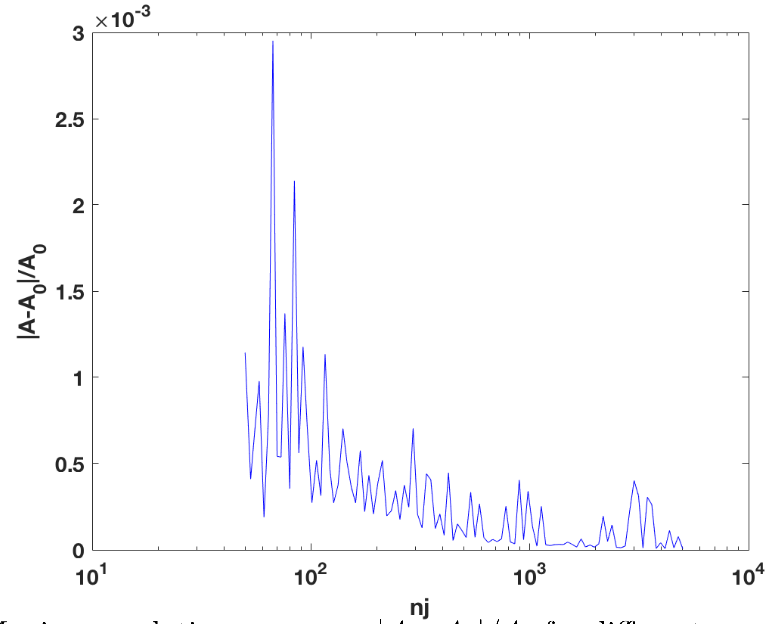

Figure 6.34: Maximum relative area error $|A - A_0|/A_0$ for different numbers of grid points in each direction nj.

The area error is embedded in the method, and as Fig. 6.34 shows, it tends to zero, but slowly. The area error shows that the possible resolution of the boundary is limited. However, the resolution close to the boundary is not critical when solving inviscid flow problems, but will cause significant problems if viscosity is taken into account. For viscous flow, the resolution of the boundary layer is important to accurately determine the shear force acting between the fluid and the structure, and due to large gradients close to the boundaries.

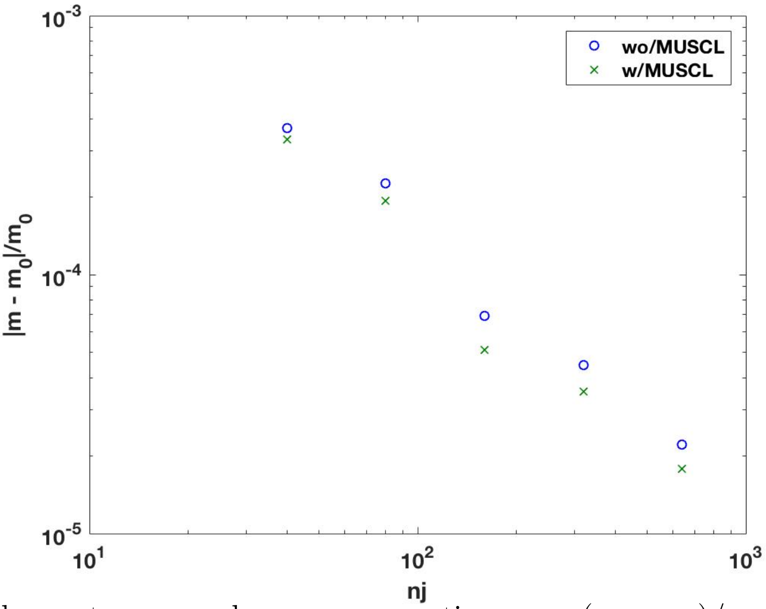

Figure 6.35: The post-processed mass conservation error $(m - m_0)/m_0$, where m is given by equation (6.2.8), determined at the time $t = 0.4$, for different grid spacings Δx.

As Fig. 6.35 shows, the post-processed mass conservation error tends to zero as the grid spacings Δx and Δy tend to zero, thus showing that the method is consistent in terms of the mass conservation error, also for this example. Further, the method using MUSCL yields slightly more accurate mass conservation than the method without MUSCL. The data points do not fit very well on a straight line, as they do for the entropy error. This is probably do to the primitive post-processing. As Fig. 6.37 shows, the mass conservation error has the same characteristic, chaotic features as the mass conservation error without post-processing, as shown by Fig. 6.32, implying that the post-processing is unable to properly remove the error caused by the area error. That makes sense, as the post-processing is very simple.

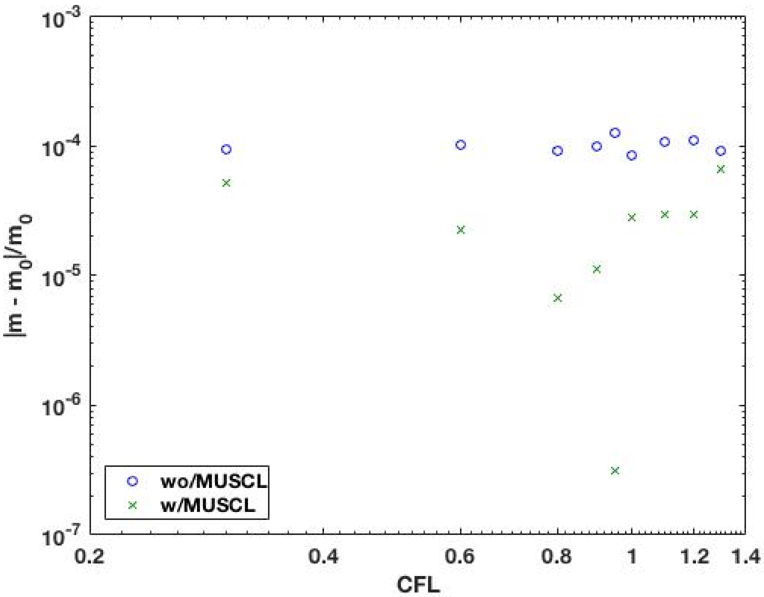

Figure 6.36: The post-processed mass conservation error $(m - m_0)/m_0$ for different CFL-numbers using the two methods specified in Table 6.6, for $nj = 60$. Other grid spacings yield similar results.

As Fig. 6.36 shows, the change in post-processed mass conservation error $(m-m_0)/m_0$ when increasing the CFL-number seems significant for the method using MUSCL, as the error is very low when $CFL = 0.95$. However, as Fig. 6.37 shows, the data point is coincidental, and does not imply that the result is close to the exact solution.

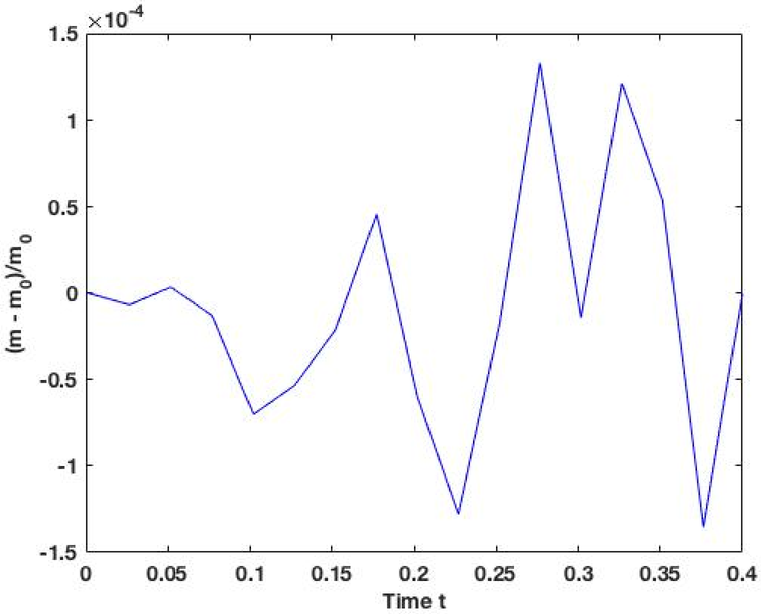

Figure 6.37: The post-processed mass conservation error $(m - m_0)/m_0$ for $CFL = 0.95$ for time $t = [0\ 0.4]$ for $nj = 60$. The magnitude of the error is varying a lot, and by coincidence, the end time error is very small.

The method is empirically shown to be stable and consistent for stable CFL-numbers, as both the entropy error and mass conservation error tends towards zero as the grid spacings, and thus the time step, due to constant CFL numbers, tends towards zero. Therefore, by Lax' equivalence theorem, it is convergent. If necessary, a more accurate post-processing could be introduced, but it is not necessary here.

6.2.3 Lift and drag

For invisid flow, the only forces acting from the fluid on the solid structure are pressure forces. As the cylinder is moving in the negative x-direction, the lift and drag forces are defined as

$$\mathbf{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} = - \sum_{i \in \{\text{Ghost points}\}} p_i \begin{bmatrix} n_x \\ n_y \end{bmatrix}_i \Delta A_i, \tag{6.2.9}$$

where F_x and F_y is the drag and lift forces, respectively. The pressure of the ghost point is a valid approximation of the pressure at the surface, as the normal derivative of the pressure at the surface is set to zero, as given equation (3.4.2), approximated by the ghost point treatment methods. The area, given as $A = Rd\phi$, is approximated as

$$A = R \sum_{i \in \text{Ghost points}} (\phi_{\mathbf{n}_{i+1}} - \phi_{\mathbf{n}_i}), \tag{6.2.10}$$

where $\phi_{\mathbf{n}}$ is the angle of the surface normal vector. Further, both the pressure and the normal surface vector is approximated by the arithmetic mean, yielding

$$\mathbf{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} = -\frac{R}{4} \sum_{i \in \{\text{Ghost points}\}} (p_{i+1} + p_i) \left(\begin{bmatrix} n_x \\ n_y \end{bmatrix}_{i+1} + \begin{bmatrix} n_x \\ n_y \end{bmatrix}_i \right) (\phi_{\mathbf{n}_{i+1}} - \phi_{\mathbf{n}_i}), \tag{6.2.11}$$

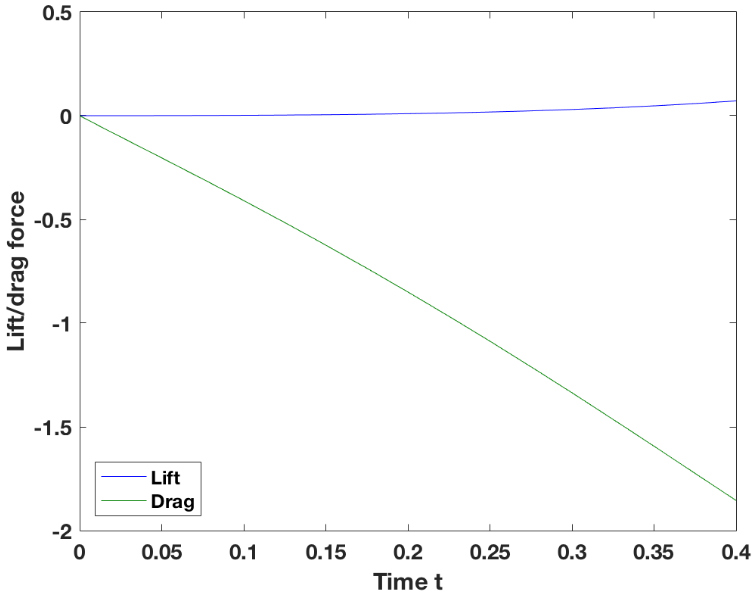

Figure 6.38: Lift and drag force exerted on the cylinder at the time $t = [0,\ 0.4]$.

As Fig. 6.38 shows, the drag is negative, thus the flow field is exerting a positive force in the negative x-direction, in the same direction as the cylinder is moving. Initially, this seems counter intuitive. However, this is due to the initial conditions of the problem, as the initial velocity of the cylinder is $u = -0.5$, and the surrounding velocity field is such that the boundary condition given by equation (3.4.1) is satisfied, making the relative velocity zero at all boundaries. Further, as Fig. 6.23 shows, the velocity field shows all negative velocities in the x-direction. The negative u-velocity values to the right of the cylinder will catch up with the cylinder, moving by advection, as time goes by and the cylinder slows down. The density and pressure builds up behind the cylinder as a result of the cylinder slowing down and the boundary condition, the velocity component normal to the boundary being zero. As more gas catches up to the cylinder, the pressure builds up further, making the negative drag force increase further. On the left side of the cylinder, the opposite happens. When the cylinder slows down, the velocity of the gas in front of it gets higher, more negative, relative to the velocity of the cylinder. This results in rarefaction, and the density and pressure decreases.

The time interval analysed is quite small considering the length scale of the example and the magnitude of the velocity, as can easily be noticed by looking at how short the initial extrema of the velocity component u are transported. Only the initial velocity field close to the cylinder has a major impact on the drag force. As Fig. 6.23 shows, the initial velocity component u gets more negative as the radius increases, the magnitude of the x-component of the velocity is proportional to the radius in this area. This is easily seen from the even spacing of the contour lines. As time goes by, more and more energetic fluid is stagnated by the boundary condition, making the pressure rise further, explaining why the drag force keeps increasing.

As Figs. 6.24, 6.28 and 6.25 show, the initial velocity in the y-direction includes two extrema, a maximum in the upper right corner and a minimum in the lower left corner. These two extrema cause two local spikes in density and pressure. As the cylinder moves towards the left, it moves closer to the extremum on the left, and further away from the extremum on the right, causing a small pressure difference between the upper and lower side of the cylinder, causing a small lift force, as shown by Fig. 6.38.

6.2.4 Computational time

The simulations are performed on the same computer as the previous example, specified in section 6.1.3.

ni x nj	40x40	80x80	160x160	320x320	640x640
wo/MUSCL	0.36	0.71	2.41	10.32	97.36
w/MUSCL	1.15	2.95	18.06	85.69	796.72
wo/MUSCL w/TVD RK3	0.81	1.66	4.62	25.00	245.93

Table 6.9: CPU time required for the different combinations of methods measured using the tic-toc functionality in MATLAB. Note that, as the methods w/MUSCL and wo/MUSCL given in Table 6.6 use different time stepping methods, another method, wo/MUSCL, but with TVD RK3, not inluded in Table 6.6, has been included here.

As for the previous example, the computational time is expected to increase by a factor of 8 when the number of grid points is doubled. The same trend is present here as was observed for the computational time of the previous example, the increase in computational time per step to the right is smaller for smaller grid sizes, and larger for the larger grid sizes.

	Relative amount of computation in boundary treatment	
nj	w/MUSCL	wo/MUSCL
40	.55	.52
80	.45	.46
160	.24	.24
320	.077	.22
640	.053	.18

Table 6.10: Relative amount of computational effort required by the boundary treatment for the two combinations of methods given in Table 6.6.

Here, the relative amount of the computational effort required for the boundary treatment is significantly larger than for the previous example, as given by Table 6.10. Further, as can be seen in Table 6.10, a trend is obvious. The relative amount of computational effort required for the boundary treatment is declining with increasing grid size. As the boundaries have a fixed length, the number of ghost points is proportional to the number of grid points in each direction, nj. The fluid domain, however, is of a fixed area, so the number of fluid cells is proportional to nj^2. As the computational effort required per ghost cell and per fluid cell per time step is fixed, this trend is logical. However, as the grid gets very large, the trend seems to stagnate.

As discussed in section 6.1.3, increasing the accuracy of a method is not always beneficial in terms of accuracy per computational effort. Here, the discussion is not about increasing the order of the method or the boundary treatment, but to increase the ghost point treatment. The increase in computational effort required if a more sophisticated ghost point treatment is utilised will be more significant for smaller grids than larger grids, relatively speaking. How large the increase is, depends on how sophisticated the new ghost point treatment is. A ghost point treatment more similar to the method developed by Sjögreen and Petersson [1], as discussed in section 1.2, would probably increase the computational effort required for the ghost point treatment significantly. However, the increased accuracy might be significant enough to increase the amount of accuracy per computational effort for the method, at least for the larger grids. Implementing a more accurate boundary treatment, similar to that of Tan and Shu [5], would greatly increase the accuracy of the boundary treatment, but would not be advantageous, as the accuracy of the internal solver would be limiting the accuracy of the method.

Chapter 7

Conclusions

The simplified Cartesian grid method for flow over moving structures is shown to be convergent for problems with smooth solutions, as the entropy error and the mass conservation error tend towards zero as the grid is refined. Further, by comparing the results with examples found in literature, it is verified that the method is solving the Euler equations correctly. The simple methods for setting ghost point and emerging fluid point density and pressure, as well as the emerging fluid point velocity, have proven most accurate. For the ghost point velocity, the mirror point method yields the most accurate results in most cases, and the simple method in some cases. Further, the method with MUSCL yields more accurate results than the method without MUSCL.

The method with MUSCL also yields the most accurate results per computational time, both at optimal CFL-conditions and otherwise. The highest stable CFL-numbers are most beneficial, as the magnitude of the errors remain relatively constant for different CFL-numbers, while the decrease in computational time is substantial.

The resulting rates of convergence are close to the expected ones, i.e. one and two for the 2-norm without and with MUSCL, respectively. However, for the infinity norm of the entropy error, the convergence rate is only 0.8 and 1.2 without and with MUSCL, respectively. The maximum error has been shown to be located in the close vicinity of the moving boundary. The lower than expected convergence rate of the infinity norm seems to be a limitation of the method, resulting from the simplified approximation of ghost point values.

The ghost point treatment represents a small portion of the computational effort. As the moving boundary treatment is shown to be a limitation of the accuracy of the method, implementing a more sophisticated ghost point treatment might increase the accuracy of the method per computational effort.

Chapter 8

Future work

Further testing of ghost point treatment methods might yield some better methods, as the convergence rate of the infinity norm of the entropy error shows that there is room for improvement of the boundary treatment.

A ghost point velocity method that might yield better accuracy is based on the simple method, b, given here. Instead of assuming that the boundary is located the same distance from the ghost point G and the fluid point F, the actual position of the boundary is taken into account. The normal velocity component $(\mathbf{u}\cdot\mathbf{n})_G$ is set as

$$(\mathbf{u}\cdot\mathbf{n})_{G_1} = \frac{(\mathbf{u}\cdot\mathbf{n})_B - \frac{\delta_{GB}}{\delta_{GF}}(\mathbf{u}\cdot\mathbf{n})_{F_1}}{1-\frac{\delta_{GB}}{\delta_{GF}}}, \tag{8.0.1}$$

where δ_{GF} is the distance between the ghost point and the fluid point, and δ_{GF} is the distance between the ghost point and the boundary.

When the mirror point method is used for the ghost point velocity, a lot of information that can be utilized to set the emerging fluid point velocity is available. As the exact position of the boundary at the previous time level is known, so is the exact position of the mirror point. For the examples given here, the velocity of the boundary is low compared to the time step, which means that such a method would yield the velocity of a point very close to the emerging fluid point in space, but at the previous time level.

When the location of the mirror point M_2 is between the fluid points F_1 and F_2, the fluid points F_1 and F_2 should be used to set the values of the mirror point, not the fluid points F_2 and F_3, as was done here. This will change the calculation of the flow variables from an extrapolation to an interpolation.

Here, the method used to set the values at the ghost points are the same for the first and second layer. Using different methods for the first and second layer ghost points might yield better results.

More sophisticated methods of setting the ghost point values, such as interpolation with more input values, similar to Sjøgren and Petterson's method, cf. section 1.2, would almost certainly yield more accurate results, but would require longer computational time. However, as the computational time required for the boundary treatment is small compared to the total computational time, increasing the level of complexity of the boundary treatment is advantageous.

Other possible methods of increasing the complexity of the boundary treatment is by utilizing bilinear interpolation or least squares interpolation, both with a significantly larger domain of influence for the ghost point values.

The next step of investigating the potential of the simplified Cartesian grid method is implementing and testing the method for more complicated two-dimensional problems, for example in supersonic or transonic conditions.

The next step would be adding viscous terms, as the range of fluid dynamic problems where the inviscid approximation is useful is quite limited. As turbulent flow is complicated, to say the least, investigating the performance of the method simulating laminar flow is beneficial.

As most industrial applications of CFD for flow over complex geometries are in three dimensions of space, the method should be tested in 3D, in case unexpected problems should arise. But, as this includes considerable effort implementing the method, the above-mentioned investigations should be done before this

Extending the method to include local grid refinement for areas of particular interest should be possible, using a quadtree data structure. As these areas move, as the structure moves, the area having a finer grid should also move, but it does not have to move with every time step. Calculating, or at least estimating, the optimal frequency of which the grid, or at least parts of the grid, should be redefined with new areas of finer and coarser grids to achieve the highest accuracy should be possible.

If the results show that the method has potential of competing with the body-fitted grid methods, an automated grid generator must be developed, for example taking .stl files at different time levels as input, and interpolating between the time levels to find the exact positions of the boundaries. The ray-tracing algorithm of Bibs and Kamath, would probably be useful to determine the exact location of the boundary in [17]. For mechanical systems, where the effect of the boundary on the flow is interesting, such as for an internal combustion engine, that would work. For an autonomous system, such as the fluid-structure interaction of the soft palate and the surrounding airflow, a multiphysics coupling is required.

Chapter 9

Acknowledgements

Thanks are given to Bernhard Müller for all his input and guidance, Martin Larsson for thorough feedback on the project work, Martin Galta and Tone Lauglo for their contributions as latex gurus, and Simen Haave for spell checking assistance.

This thesis is a continuation of the project work in [6]. It is basis for 30 SP. MATLAB is used for implementation and testing.

Bibliography

[1] Sjögreen, B. and Petersson, N.A. A Cartesian embedded boundary method for hyperbolic conservation laws. *Communications in Computational Physics*, 2(6):1199–1219, 2007.

[2] Farooq, M.A., Skøien, A.A., and Müller, B. Cartesian grid method for the compressible Euler equations using simplified ghost point treatments at embedded boundaries. *Computers & Fluids*, 82:50–62, 2013.

[3] Farooq, M.A. Cartesian grid method for compressible flow simulation, PhD Thesis, NTNU. 2012.

[4] Skøien, A.A. Cartesian grid method for the compressible Navier-Stokes equations, Master's Thesis, NTNU, 2012.

[5] Tan, S. and Shu, E.W. A high order moving boundary treatment for compressible inviscid flows. *Journal of Computational Physics*, 230(15):6023–6036, 2011.

[6] Lekven, M. Cartesian grid method for compressible flow over moving structures, project work, NTNU, 2016.

[7] White, F.M. *Fluid Mechanics.* McGraw-Hill, 6th edition, 2009.

[8] Peskin, C.S. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics*, 10(2):252–271, 1972.

[9] Mittal, R. and Iaccarino, G. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[10] Pletcher, R.H., Tannehill, J.C. and Anderson, D. *Computational fluid mechanics and heat transfer.* CRC Press, 3rd edition, 2013.

[11] Forrer, H. and Jeltsch, R. A higher-order boundary treatment for Cartesian-grid methods. *Journal of Computational Physics*, 140(2):259–277, 1998.

[12] Schneiders, L., Hartmann, D., Meinke, M. and Schröder, W. An accurate moving boundary formulation in cut-cell methods. *Journal of Computational Physics*, 235:786–809, 2013.

[13] Yang, G., Causon, D.M. and Ingram, D.M. Calculation of compressible flows about complex moving geometries using a three-dimensional Cartesian cut cell method. *International Journal for Numerical Methods in Fluids*, 33(8):1121–1151, 2000.

[14] Forrer, H. and Berger, M. Flow simulations on Cartesian grids involving complex moving geometries. In *Hyperbolic problems: theory, numerics, applications*, pages 315–324. Springer, 1999.

[15] Tan, S. and Shu, E.W. Inverse Lax-Wendroff procedure for numerical boundary conditions of conservation laws. *Journal of Computational Physics*, 229(21):8144–8166, 2010.

[16] Müller, B. Introduction to computational fluid dynamics, Lecture notes, NTNU, 2015.

[17] Bihs, H. and Kamath, A. A combined level set/ghost cell immersed boundary representation for floating body simulations. *International Journal for Numerical Methods in Fluids*, 2016.

Appendices

Appendix A

Derivation of various equations

A.1 Velocity of the ghost points, method a

$$
\begin{aligned}
(\mathbf{u}\cdot\mathbf{n})_{G_1} =& (\mathbf{u}\cdot\mathbf{n})_B && \text{(A.1.1)}\\
(\mathbf{u}\cdot\mathbf{t})_{G_1} =& (\mathbf{u}\cdot\mathbf{t})_{F_1} && \text{(A.1.2)}\\
\mathbf{n} =& [n_x \quad n_y] && \text{(A.1.3)}\\
\mathbf{t} =& [n_y \quad -n_x] && \text{(A.1.4)}
\end{aligned}
$$

Using equations (A.1.3) and (A.1.4) in (A.1.1) and (A.1.2) to get

$$
\begin{aligned}
u_{G_1}n_x + v_{G_1}n_y &= u_B n_x + v_B n_y && \text{(A.1.5)}\\
u_{G_1}n_y - v_{G_1}n_x &= u_{F_1}n_y - v_{F_1}n_x, && \text{(A.1.6)}
\end{aligned}
$$

which combines to form

$$
\begin{bmatrix} n_y & -n_x \\ n_x & n_y \end{bmatrix} \begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} u_{F_1}n_y - v_{F_1}n_x \\ u_B n_x + v_B n_y \end{bmatrix}. \tag{A.1.7}
$$

By multiplying from the left by

$$
\begin{bmatrix} n_y & -n_x \\ n_x & n_y \end{bmatrix}^{-1} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \tag{A.1.8}
$$

such that

$$
\begin{aligned}
\begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} &= \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \begin{bmatrix} u_{F_1}n_y - v_{F_1}n_x \\ u_B n_x + v_B n_y \end{bmatrix}, && \text{(A.1.9)}\\
\begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} &= \begin{bmatrix} n_x(n_x u_B + n_y v_B) + n_y(n_y u_{F1} - n_x v_{F1}) \\ n_y(n_x u_B + n_y v_B) + n_x(n_x v_{F1} - n_y u_{F1}) \end{bmatrix} && \text{(A.1.10)}
\end{aligned}
$$

is obtained.

A.1.1 Velocity of ghost points, method b

$$\frac{1}{2}((\mathbf{u}\cdot\mathbf{n})_{F_1} + (\mathbf{u}\cdot\mathbf{n})_{G_1}) = (\mathbf{u}\cdot\mathbf{n})_B \tag{A.1.11}$$

$$(\mathbf{u}\cdot\mathbf{t})_{G_1} = (\mathbf{u}\cdot\mathbf{t})_{F_1} \tag{A.1.12}$$

(A.1.11) yields

$$(\mathbf{u}\cdot\mathbf{n})_{G_1} = 2(\mathbf{u}\cdot\mathbf{n})_B - (\mathbf{u}\cdot\mathbf{n})_{F_1} \tag{A.1.13}$$

Using equations (A.1.3) and (A.1.4) such that

$$\begin{bmatrix} n_y & -\,n_x \\ n_x & n_y \end{bmatrix} \begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} u_{F_1}n_y - v_{F_1}n_x \\ -u_{F_1}n_x - v_{F_1}n_y + 2u_Bn_x + 2v_Bn_y \end{bmatrix} \tag{A.1.14}$$

is obtained. Then, multiplying from the left by

$$\begin{bmatrix} n_y & -\,n_x \\ n_x & n_y \end{bmatrix}^{-1} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \tag{A.1.15}$$

such that

$$\begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \begin{bmatrix} u_{F_1}n_y - v_{F_1}n_x \\ -u_{F_1}n_x - v_{F_1}n_y + 2u_Bn_x + 2v_Bn_y \end{bmatrix}, \tag{A.1.16}$$

$$\begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} u_{F1} - 2\big(n_xu_{F_1} + n_yv_{F_1}\big)n_x + 2\big(n_xu_B + n_yv_B\big)n_x \\ v_{F1} - 2\big(n_xu_{F_1} + n_yv_{F_1}\big)n_y + 2\big(n_xu_B + n_yv_B\big)n_y \end{bmatrix} \tag{A.1.17}$$

is obtained.

A.1.2 Velocity of ghost points, method c

$$\frac{1}{2}((\mathbf{u}\cdot\mathbf{n})_{M_1} + (\mathbf{u}\cdot\mathbf{n})_{G_1}) = (\mathbf{u}\cdot\mathbf{n})_B \tag{A.1.18}$$

$$(\mathbf{u}\cdot\mathbf{t})_{G_1} = (\mathbf{u}\cdot\mathbf{t})_{M_1} \tag{A.1.19}$$

(A.1.18) yields

$$(\mathbf{u}\cdot\mathbf{n})_{G_1} = 2(\mathbf{u}\cdot\mathbf{n})_B - (\mathbf{u}\cdot\mathbf{n})_{M_1} \tag{A.1.20}$$

Using equations (A.1.3) and (A.1.4) such that

$$\begin{bmatrix} n_y & -\,n_x \\ n_x & n_y \end{bmatrix} \begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} u_{M_1}n_y - v_{M_1}n_x \\ -u_{M_1}n_x - v_{M_1}n_y + 2u_Bn_x + 2v_Bn_y \end{bmatrix} \tag{A.1.21}$$

is obtained. Then, multiplying from the left by

$$\begin{bmatrix} n_y & -\,n_x \\ n_x & n_y \end{bmatrix}^{-1} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \tag{A.1.22}$$

such that

$$\begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \begin{bmatrix} u_{M_1} n_y - v_{M_1} n_x \\ -u_{M_1} n_x - v_{M_1} n_y + 2u_B n_x + 2v_B n_y \end{bmatrix}, \tag{A.1.23}$$

$$\begin{bmatrix} u_{G_1} \\ v_{G_1} \end{bmatrix} = \begin{bmatrix} u_{F1} - 2\big(n_x u_{M_1} + n_y v_{M_1}\big) n_x + 2\big(n_x u_B + n_y v_B\big) n_x \\ v_{F1} - 2\big(n_x u_{M_1} + n_y v_{M_1}\big) n_y + 2\big(n_x u_B + n_y v_B\big) n_y \end{bmatrix} \tag{A.1.24}$$

is obtained.

A.1.3 Density and pressure of emerging fluid points, method 3

$$\partial p - \rho c \partial(\mathbf{u} \cdot \mathbf{n}) = 0 \tag{A.1.25}$$

$$\partial s = 0 \tag{A.1.26}$$

$$0 = p_{F_1} - p_{F_2} - (\rho c)_{F2}((\mathbf{u} \cdot \mathbf{n})_{F_1} - (\mathbf{u} \cdot \mathbf{n})_{F_2}) \tag{A.1.27}$$

$$p_{F_1} = p_{F_2} + (n_x(u_1 - u_2) + n_y(v_1 - v_2))(\rho c)_{F_2} \tag{A.1.28}$$

$$p_{F_1} = p_{F_2} + (n_x(u_1 - u_2) + n_y(v_1 - v_2))\sqrt{(\gamma p_{F_2} \rho_{F_2})} \tag{A.1.29}$$

as

$$(\rho c) = \sqrt{\frac{\gamma p}{\rho}} \rho \tag{A.1.30}$$

$$(\rho c) = \sqrt{\gamma p \rho} \tag{A.1.31}$$

$$\partial s = s_{F_1} - s_{F_2} \tag{A.1.32}$$

$$\frac{p_{F_2}}{\rho_{F_2}^{\gamma}} = \frac{p_{F_1}}{\rho_{F_1}^{\gamma}} \tag{A.1.33}$$

$$\frac{\rho_{F_1}/\rho_0}{\rho_{F_2}/\rho_0}^{\gamma} = \frac{p_{F_1}/p_0}{p_{F_2}/p_0} \tag{A.1.34}$$

$$\frac{\rho_{F_1}/\rho_0}{\rho_{F_2}/\rho_0} = \frac{p_{F_1}/p_0}{p_{F_2}/p_0}^{1/\gamma} \tag{A.1.35}$$

$$\rho_{F_1} = \rho_{F_2} \frac{p_{F_1}}{p_{F_2}}^{1/\gamma} \tag{A.1.36}$$

A.1.4 Velocity of emerging fluid points, method A

$$(\mathbf{u} \cdot \mathbf{n})_{F_1} = (\mathbf{u} \cdot \mathbf{n})_B \tag{A.1.37}$$

$$(\mathbf{u} \cdot \mathbf{t})_{F_1} = (\mathbf{u}' \cdot \mathbf{t})_{F_1} \tag{A.1.38}$$

$$\mathbf{n} = [n_x \quad n_y] \tag{A.1.39}$$

$$\mathbf{t} = [n_y \quad -n_x] \tag{A.1.40}$$

Using equations (A.1.39) and (A.1.40) in (A.1.37) and (A.1.38) to get

$$u_{F_1}n_x + v_{F_1}n_y = u_Bn_x + v_Bn_y \tag{A.1.41}$$
$$u_{F_1}n_y - v_{F_1}n_x = u'_{F_1}n_y - v'_{F_1}n_x, \tag{A.1.42}$$

which combines to form

$$\begin{bmatrix} n_y & -n_x \\ n_x & n_y \end{bmatrix} \begin{bmatrix} u_{F_1} \\ v_{F_1} \end{bmatrix} = \begin{bmatrix} u'_{F_1}n_y - v'_{F_1}n_x \\ u_Bn_x + v_Bn_y \end{bmatrix}. \tag{A.1.43}$$

By multiplying from the left by

$$\begin{bmatrix} n_y & -n_x \\ n_x & n_y \end{bmatrix}^{-1} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \tag{A.1.44}$$

such that

$$\begin{bmatrix} u_{F_1} \\ v_{F_1} \end{bmatrix} = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \end{bmatrix} \begin{bmatrix} u'_{F_1}n_y - v'_{F_1}n_x \\ u_Bn_x + v_Bn_y \end{bmatrix}, \tag{A.1.45}$$
$$\begin{bmatrix} u_{F_1} \\ v_{F_1} \end{bmatrix} = \begin{bmatrix} n_y(n_yu'_{F1} - n_xv'_{F1}) + n_x(n_xu_B + n_yv_B) \\ n_x(n_xv'_{F1} - n_yu'_{F1}) + n_y(n_xu_B + n_yv_B) \end{bmatrix} \tag{A.1.46}$$

is obtained.

A.1.5 Velocity of ghost points, method in section 8

The normal component of the velocity at the ghost point G_1 is set such that interpolating the normal component of the velocity at the ghost point G_1 and the fluid point F_1 yields the normal component of the velocity at the boundary B.

$$(\mathbf{u}\cdot\mathbf{n})_B = (\mathbf{u}\cdot\mathbf{n})_{G_1} + \big((\mathbf{u}\cdot\mathbf{n})_B = (\mathbf{u}\cdot\mathbf{n})_{F_1} - (\mathbf{u}\cdot\mathbf{n})_B = (\mathbf{u}\cdot\mathbf{n})_{G_1}\big)\frac{\delta_{GB}}{\delta_{GF}}, \tag{A.1.47}$$

where δ_{GB} is the distance between the ghost point and the boundary, and δ_{GF} is the distance between the ghost point G_1 and the fluid point F_1. Solving for $(\mathbf{u}\cdot\mathbf{n})_{G_1}$ yields

$$(\mathbf{u}\cdot\mathbf{n})_{G_1} = \frac{(\mathbf{u}\cdot\mathbf{n})_B - \frac{\delta_{GB}}{\delta_{GF}}(\mathbf{u}\cdot\mathbf{n})_{F_1}}{1 - \frac{\delta_{GB}}{\delta_{GF}}}. \tag{A.1.48}$$

A.2 Risk assessment form

NTNU	Hazardous activity identification process	Prepared by	Number	Date
HSE		HSE section	HMSRV2601E	09.01.2013
		Approved by		Replaces
		The Rector		01.12.2006

Unit: *(Department)* EPT **Date:** 8/6-17

Line manager: Olav Bolland

Participants in the identification process (including their function): Bernhard Müller (supervisor)

Short description of the main activity/main process: Master project for student Morten Lekven. Verification of a Cartesian grid method for compressible flow over moving structures.

Is the project work purely theoretical? (YES/NO): YES *Answer "YES" implies that supervisor is assured that no activities requiring risk assessment are involved in the work. If ~~YES~~ NO, briefly describe the activities below. The risk assessment form need not be filled out.*

Signatures: *Responsible supervisor:* B Müller *Student:* M Lekven

ID nr.	Activity/process	Responsible person	Existing documentation	Existing safety measures	Laws, regulations etc.	Comment