

# Kartesiske grid metode for kompressibel strømning over oscillerende vingeprofiler.

**Carsten Berge Helverschou**

Master i ingeniørvitenskap og IKT  
Innlevert: juni 2013  
Hovedveileder: Bernhard Müller, EPT

Norges teknisk-naturvitenskapelige universitet  
Institutt for energi- og prosessteknikk



# Cartesian Grid Method for Compressible Flow over Oscillating Airfoils

**Carsten Helverschou**

Spring, 2013

EPT-M-2013-20

**MASTER THESIS**

for

Stud.techn. Carsten Berge Helverschou

Spring 2013

**Cartesian grid method for compressible flow over oscillating airfoils**  
*Kartesiske grid metode for kompressibel strømming over oscillerende vingeprofil***Background and objective**

The Cartesian grid method has recently become a popular method in computational fluid dynamics (CFD) to compute flows over complex geometries in aerodynamics. The reason lies in its simplification of grid generation. Instead of generating a body-fitted structured or unstructured grid, the geometry of an airfoil for example is embedded into a simple Cartesian grid by taking the effect of the airfoil into account at grid points near the surface of the airfoil.

The objective of the project is to investigate the Cartesian grid method for two-dimensional (2D) compressible flow over oscillating airfoils. The accuracy and efficiency of the method will be checked by comparison with body-fitted grid methods. The 2D Euler equations for compressible flow will be solved by an explicit finite volume method on a Cartesian grid. An existing Cartesian grid code for compressible flow over stationary cylinders and airfoils may be used as an example. The choice of the programming language will depend on the student's interest and programming skills.

**The following tasks are to be considered:**

- 1 Review the literature on Cartesian grid methods for flow over oscillating airfoils.
- 2 Develop and implement a Cartesian grid method to numerically solve the 2D Euler equations for compressible flow over oscillating airfoils.
- 3 Verify the Cartesian grid method for suitable test cases of 2D inviscid compressible flow over oscillating airfoils.
- 4 Assess the accuracy and efficiency of the Cartesian grid method for compressible flow over oscillating airfoils by comparison with body-fitted grid methods.

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.

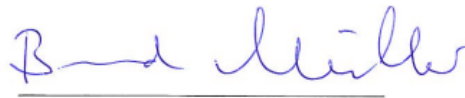
Pursuant to "Regulations concerning the supplementary provisions to the technology study program/Master of Science" at NTNU §20, the Department reserves the permission to utilize all the results and data for teaching and research purposes as well as in future publications.

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, 14. January 2013

  
Olav Bolland  
Department Head

  
Bernhard Müller  
Academic Supervisor

# Sammendrag

Et program har blitt utviklet for å løse de todimensjonale kompressible Eulerligningene for strømninger over et oscillerende flyprofil. En Kartesisk gittermetode er blitt benyttet, som gjør det nødvendig å behandle spøkelsespunkter i strukturen. Her den nye forenklete spøkelsespunktbehandlingsmetoden har blitt benyttet. De numeriske fluksene blir approksimert ved hjelp av en lokal Lax-Friedrich metode med MUSCL skjemaet, og den tredje ordens total variasjonsminkende Runge-Kutta-metoden har blitt benyttet for tidsdiskritisering. Bevegelse har blitt modelert ved å markere gitterpunktene utifra deres posisjon som enten fluid, spøkelsespunkt eller fast stoff. Den normale hastighetskomponenten til et spøkelsespunkt er valgt slik at dens gjennomsnitt med normalhastigheten i det nærmeste fluidpunktet er lik den normale hastighetskomponenten til den oscillerende flyvingen i det punktet på overflaten som ligger i en rett linje mellom spøkelsespunktet og fluidpunktet. Når man benytter seg av spøkelsespunktene, blir kanten på strukturen sett på som en bevegelig symmetrilinje som strømningshastighetene er speilet rundt. Metoden har i hovedsak blitt utviklet for bruk på profilet NACA0012, men kan med enkelhet benyttes på andre geometrier.

Programmet som henter ut resultatene som vises er fra et annet program enn det som opprinnelig ble utviklet for prosjektet. En undervurdering av hvor mange problemer som kunne dukke opp i slutfasen av utviklingen førte til en mangel på tid til å produsere resultater. Problemene som eksisterte da programmet ble forlatt var relatert til ustabilitet. Mest sannsynlig eksisterte ustabilitetsproblemene på grunn av feiltrinn under koding. Koden som benyttes til å skape resultatene er likevel veldig lik programmet som beskrives, for eksempel benytter de begge den nye forenklete spøkelsespunktbehandlingsmetoden.

Testtilfeller har blitt kjørt for å kunne sammenlignes med noen resultater fra en artikkel av Schneiders et. al.[9]. En NACA0012 flyvinge oscillerer mellom innfallsvinkel på 2.5 grader og -2.5 grader. Strømningen i denne simuleringen var transonisk, som det var mistenkt at kunne være problematisk for koden. Konvergens krevde en betydelig økt mengde iterasjoner å nå sammenlignet med kjøring med høyere mach-tall. Likevel visualiserer resultatene noen av problemene man kan støte på når man benytter disse metodene på transonisk strømning.

## Abstract

A program has been developed to solve the 2D compressible Euler equations for flows over oscillating airfoils. A Cartesian grid method has been used, making it necessary to handle ghost points in the structure. Here the new simplified ghost point treatment has been used. The numerical fluxes are approximated by the local Lax-Friedrich method with MUSCL, and the third order total variation diminish-

ing Runge-Kutta method is employed for time discretization. The movement of the airfoil has been modeled by flagging the grid points according to their current position as fluid, ghost or solid points. The normal velocity component at a ghost point is chosen such that its average with the normal velocity at the closest fluid point is equal to the normal velocity of the oscillating airfoil at the intersection with the straight line between ghost and fluid points. In using the ghost points the wall of the airfoil is considered as a moving symmetry boundary around which the flow velocities are mirrored. Thus the density, pressure, and tangential velocity component at a ghost point are set equal to their values at the closest grid point. The method has mainly been tailored for use on the NACA0012 airfoil, but other airfoil shapes can be easily implemented.

The program acquiring the results shown are from a different program than the one originally developed for this project. An underestimation of how many problems that would turn up during the final stages of development led to a lack of time to produce results. The problems existing when the code was abandoned were related to instability. Most likely the instability problems were due to programming errors. The code used to provide the results shown is very similar to the one explained in this article, for instance both utilize the new simplified ghost point treatment.

Test cases have been run to compare with some of the results from an article by Schneiders et al.[9]. A NACA00012 airfoil oscillating between angle of attack of 2.5 degrees and -2.5 degrees. The flow in this test case was transonic, which was suspected to possibly be problematic for the code. Convergence required a substantially increased amount of iterations to be had compared to simulations with higher mach numbers. Still, the results visualizes some of the problems one might encounter when applying these methods to transonic flow.

# Contents

<b>Legend</b>	<b>7</b>
<b>1 Introduction</b>	<b>9</b>
<b>2 Governing Equations</b>	<b>12</b>
2D Euler Equations . . . . .	12
Description of the geometry . . . . .	14
Boundary conditions . . . . .	15
<b>3 Numerical Discretization</b>	<b>16</b>
Spatial Discretization . . . . .	16
Numerical methods in time . . . . .	18
Stability analysis . . . . .	18
<b>4 Ghost point treatment</b>	<b>19</b>
Ghost Point Identification . . . . .	19
Ghost Point Values . . . . .	21
Oscillation Handling . . . . .	23
<b>5 The Program</b>	<b>26</b>
Functions . . . . .	26
Program diagram . . . . .	29
Flowchart . . . . .	30
<b>6 Results</b>	<b>31</b>
<b>7 Conclusions and Outlook</b>	<b>38</b>
<b>8 Acknowledgements</b>	<b>38</b>



# Legend

$\gamma$  = ratio of specific heats [-]

$R$  = gas constant [ $\frac{J}{kg K}$ ]

$u$  = velocity in x-direction [ $\frac{m}{s}$ ]

$v$  = velocity in y-direction [ $\frac{m}{s}$ ]

$t$  = time [s], or in section 2.2 it is used as thickness to chord ratio [-]

$c$  = speed of sound [ $\frac{m}{s}$ ], or in section 2.2 it is used as chord length [m]

$\rho$  = density [ $\frac{kg}{m^3}$ ]

$c_p$  = specific heat at constant pressure [ $\frac{J}{kg K}$ ]

$c_v$  = specific heat at constant volume [ $\frac{J}{kg K}$ ]

$T$  = temperature[K]

$C$  = Courant-Friedrichs-Lewy number [-]

$\Delta t$  = time step [s]

$\Delta x$  = grid cell size in x-direction[m]

$\Delta y$  = grid cell size in y-direction[m]

$p$  = pressure [ $\frac{N}{m^2}$ ]

$E$  = specific total energy [ $\frac{J}{kg}$ ]

# List of Figures

1	The NACA0012 profile. . . . .	14
2	Rectangular domain for flow over an airfoil. . . . .	15
3	The four parts in which ghost points are handled differently. . . . .	19
4	Assignment of angles at different ghost points. . . . .	20
5	Separation of the tail matrix. . . . .	21
6	The rollback method of the ghost points. . . . .	24
7	Notation of the points used in the discontinuity updating. . . . .	25
8	Function overview. . . . .	29
9	Flowchart. . . . .	30
10	NACA0012 - at $M_\infty = 0.755$ , $\alpha = 0$ degrees. - 301 x 151 grid. . . . .	31
11	$C_L$ versus angle of attack $\alpha(t)$ - 301 x 151 grid for NACA0012 $M_\infty =$ 0.755. . . . .	32
12	$C_L$ from [9] for comparison. . . . .	33
13	NACA0012 - at $M_\infty = 0.755$ , $\alpha = 0$ degrees. - 201 x 101 grid. . . . .	34
14	$C_L$ versus angle of attack $\alpha(t)$ - 201 x 101 grid for NACA0012 $M_\infty =$ 0.755. . . . .	34
15	Pressure coefficient for NACA0012 at $M_\infty = 0.755$ and $\alpha(t) = 2.5$ degrees. . . . .	35
16	Pressure coefficient for NACA0012 at $M_\infty = 0.755$ and $\alpha(t) = 2.34$ degrees. From [9] for comparison. . . . .	36
17	Convergence history for NACA0012, $M_\infty = 0.755$ , $\alpha = 0.016$ de- grees - 201 x 101 grid. . . . .	37

# 1 Introduction

Computational Fluid Dynamics (CFD) has become a more and more popular alternative to experiments ever since the computer was first invented. CFD can give accurate results with a margin of the costs laboratory experiments have. Duplicating an experiment with small changes becomes easy, since only some parameters need to be altered instead of new models that need to be built. CFD can also simulate situations for which it may be impossible or very hard to acquire a model or system, as vacuum, rare materials, or extreme temperatures. Visualization of results also often becomes easier and more complete when using CFD, as all the flow variables of the system are available, and plots can quickly be created on which small differences can be made visible. CFD is used in simulations in many scientific fields, ranging from weather analysis to airfoil optimization. With the advancement of computer processing speeds and the improvement of numerical algorithms CFD will probably see more and more use in the coming years.

When using CFD, the grid becomes an important part of the method. A very usual grid type is the body-fitted one, where the grid is created according to the shape of the structure around which the flow is passing. In cases of complex geometries, the creation of the grid itself can prove difficult, and often separate programs are used for this. The Cartesian Grid method (CGM) however, is based on a grid that does not change through the domain, because all blocks have equal size and shape. The advantages of the CGM are fast grid generation, ability to handle complex geometries, the values in each grid point is easily accessible and the amount of stored data required totally is reduced. With the CGM changing the geometry by adding bodies, structures, walls, channels or similar, can be done quickly and easily. The CGM is also compatible with several higher order methods in time and space, some of which are used in this project.

The main disadvantage of the Cartesian grid method is the handling of curved surfaces. Since a point only can be either a solid or a fluid point, the airfoil surface will follow a "stairway pattern" instead of a smooth curve. There are three ways of handling this: cut-cells, grid refinement, or usage of a very fine grid. In any case a constant referencing to the formula that defines the structure is important, to know the actual angle and curve of the face helps with modeling something that is closer to reality.

Another major problem with a Cartesian grid is movement. When the grid is completely fixed, a moving boundary will shift the attributes of a grid point or cell, which will cause discontinuities without proper handling. Without the use of cut-cells the state to which a point can be flagged can only be fully solid or fully ghost point. This leads to not so smooth movement; which again may lead to the airfoil movement to be possibly distorted and it can seem to move in a more staggered pattern than what is the case. An obvious solution to some of these problems is a finer grid, however that will increase computation times. Another solution to

the problem is presented in two articles, one from 2006 by Krishmann and Liu [4] and the other from 2010 by Liao et al. [6]. The solutions presented here does not change the flagging of points in any way, only the surface normals of the structure are changed in the ghost point treatment. This only works for small deviations of angle, flutters, pertubations of the structure. Still they obtained good results even with a large angle of attack as long as it does not deviate too much from this angle throughout the simulation.

Some history about the Cartesian grid method: Before the 1990's-2000's the Cartesian grid method was not as commercially popular as body-fitted grid methods. This may cause some people to view it as a new method, since it has been becoming increasingly popular lately. However, the Cartesian grid method was used as early as in 1972 by Peskin [8] to simulate flow in hearts. Here it was referred to as an immersed boundary method as it often is. In 1999 Udaykumara et al. [12] investigated the use of a Cartesian grid method on a moving interface. Here they discussed several of the issues that may arise when combining moving boundaries with a Cartesian grid. One of the problems mentioned here is that it can be hard keeping the conservation properties and the accuracy of the numerical solver when applying boundary treatment to a moving boundary. They handle this by the use of an interpolation scheme near the immersed boundary. Later in 2005 Mittal and Iaccarino [7] reviewed different aspects to the immersed boundary methods. Among these methods were ways of handling moving boundaries, where the problems associated with "freshly cleared cells" are discussed. Their article can almost be viewed as a manual or textbook for use of immersed boundary methods. For they address many different problems, solutions and alternatives that could be useful when working with immersed boundaries.

In 2007 Sjögreen and Petersson [10] developed a Cartesian grid method to solve the compressible Euler equations based on a ghost point treatment involving interpolation. This gave very accurate results, but the computation involved advanced interpolation schemes. In 2011 Farooq and Müller [1, 2] devised a Cartesian grid method for the 2D compressible Euler equations using a simplified ghost point treatment method by considering only points on the x- and y-grid lines adjacent to the body surface as possible mirror points for the ghost point . Compared to the method used by Sjögreen and Petersson this simplified ghost point treatment method drastically reduced the amount of work needed for the identification of the ghost points, which again allowed for faster simulations.

Skøien [11] later extended the simplified ghost point treatment method to the new simplified ghost point treatment method by considering grid points on the diagonal as well as possible mirror points, applying it to solving the 2D Euler and Navier-Stokes equations. In this paper a variation of that method will be used as it has been shown to give better results than just identifying ghost points on the x- and y-grid lines, while keeping tolerable computation times [11].

In 2013 Schneiders et al. [9] outlined cut-cell methods for the use on a moving boundary . Their main focus was the issue of temporal discontinuities in emerging cells, an issue explained earlier in the introduction. Several different approaches to the issue are discussed in their article, like the method that is used here, and they end up with a flux-redistribution related technique. With this , and several interpolations, they are able to obtain an overall conservative and stable scheme.

In this project the problems associated with the movement of the structure; i.e. the oscillation of airfoils, are solved by separate checks and updates on the parts of the airfoil where fluid points may emerge from the solid, as well as limiting the program to use on airfoils or airfoil-like shapes exploiting knowledge of the structure to improve accuracy. With the ghost point treatment used in this project, the normal at the solid boundary between the ghost point and its closest fluid point is used to create symmetry boundary conditions at the ghost point mirroring the flow variables at the closest fluid point. This keeps the effect of the accurate normal at the structure surface applied to the system.

In the article, first governing equations are gone through, then the geometry of the airfoil is explained. The boundary conditions, and how they are treated is shown, before the numerical discretization is gone through, first spatial, then time discretization. Then the ghost point treatment is explained in greater detail, visualizing the solutions used in this project, for stationary and oscillating cases. Some of the key characteristics of the program are explained, including class diagram and flow chart. Then the results from testing are shown, verifying and validating the methods, as well as showing some visuals. Finally one can read the conclusions, acknowledgements and reference list.

## 2 Governing Equations

In this section the governing expressions will be discussed. First there are the 2D Euler equations and the derivation of some required formulas. Then the equations of the geometry is described, and finally boundary conditions.

### 2D Euler Equations

As mentioned in the introduction, the equations to be solved are the 2D Euler equations for compressible flow. They read as follows:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0, \quad (2.1)$$

where  $U$  is the vector of the conservative variables, and  $F$  and  $G$  are the flux vectors in the x- and y-directions respectively:

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ u(\rho E + p) \end{bmatrix}, G = \begin{bmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ v(\rho E + p) \end{bmatrix} \quad (2.2)$$

The flow variables are:

$\rho$ : density,  $u$ : velocity in x-direction,  $v$ : velocity in y-direction,  $p$ : the pressure, and  $E$ : the specific total energy.

The conservative variables can be expressed by the primitive variables:  $\rho$ ,  $u$ ,  $v$  and  $p$ . To solve the equations, an expression for specific total energy using only those flow variables is required. The fluid in focus is air which is considered as a perfect gas. This allows for use of the following equations of state:

$$e = c_v T \quad (2.3)$$

$$p = \rho R T \quad (2.4)$$

where  $e$  is the specific internal energy and  $T$  is the temperature.

A few other definitions are required as well:

$$\gamma = \frac{c_p}{c_v} \quad (2.5)$$

$$R = c_p - c_v \quad (2.6)$$

where  $\gamma$  is the ratio of specific heats at constant pressure and volume respectively, and  $R$  is the gas constant.

From the expression for the pressure (2.4) and the one for internal energy (2.3) it is possible to derive:

$$p = \rho RT = \rho R \frac{(c_v T)}{c_v} = \frac{R}{c_v} \rho e \quad (2.7)$$

Now internal energy = total energy - kinetic energy:

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

By combining (2.7) and (2.8) a new expression for p can be acquired:

$$p = (\gamma - 1) \left[ \rho E - \frac{1}{2} \rho (u^2 + v^2) \right] \quad (2.9)$$

From this an expression for E can be derived:

$$E = \frac{p}{\rho(\gamma - 1)} + \frac{1}{2}(u^2 + v^2) \quad (2.10)$$

One of the main disadvantages to using the Euler equation is the loss of the viscous effects. Drag forces on the structure will not be as large as if the Navier Stokes equations had been used. However, with air as the fluid and the flow velocities in the ranges of Mach 2-3, the viscous effects will be much smaller than the effects of shock waves which are properly described by the compressible Euler equations.

## Description of the geometry

To describe the geometry of the structure to the program, all the user needs is a formula describing the surface of the structure. For airfoils the width of the structure can be given as a function of position  $x$ . One drawback is that the function is required to be symmetrical around the  $x$ -direction. If the formula of the wanted structure contains discontinuities, some slight alterations must be done to the code, however, not very severe. Here we consider NACA0012 airfoils, for which the formula used is:

$$y(x) = \frac{t}{0.2} c \left[ 0.2969 \sqrt{\frac{x}{c}} - 0.1260 \left(\frac{x}{c}\right) - 0.3516 \left(\frac{x}{c}\right)^2 + 0.2843 \left(\frac{x}{c}\right)^3 - 0.1015 \left(\frac{x}{c}\right)^4 \right] \quad (2.11)$$

where  $t=0.12$  is the the thickness chord  $c$  ratio for NACA0012.

For ghost point treatment purposes, the differentiated formula is also required for the angle of the profile wall. In case of discontinuities, a differentiation of the parts of the equation is also possible. The NACA0012 profile was chosen because of the many other works done on this profile.

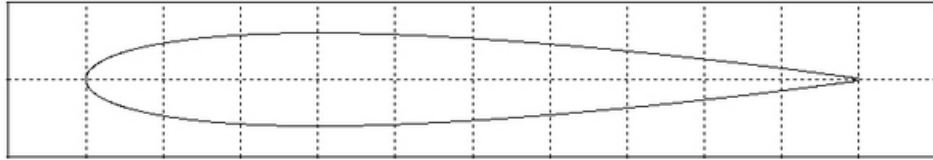


Figure 1: The NACA0012 profile.



## Boundary conditions

The calculations will be done for an external flow over a structure located in the center of the domain. The boundaries are all four sides of the area as well as the solid surface of the structure. At the left boundary values are set for  $\rho, u$  and  $v$  based on the mach number. (2.12) There is no development of the flow prior to the domain so the inlet velocity profile is constant.

$$U(x_0, y, t) = U_\infty \quad (2.12)$$

The upper and lower boundaries have zero gradient conditions (eq.2.13) to allow any shocks to pass unhindered. At the right boundary the flow is supposed to be supersonic. Therefore, a zero gradient is also used there (eq. 2.14), because that boundary condition leads to using the upwind method for the flux adjacent to the supersonic boundary there.

$$\frac{\partial U(x, y, t)}{\partial y} = 0 \quad (2.13)$$

for  $y = 0$  and  $y = y_{max}$ .

$$\frac{\partial U(x, y, t)}{\partial x} = 0 \quad (2.14)$$

for  $x = x_{max}$ .

Figure 2 shows the domain with the defined boundary conditions and lengths:

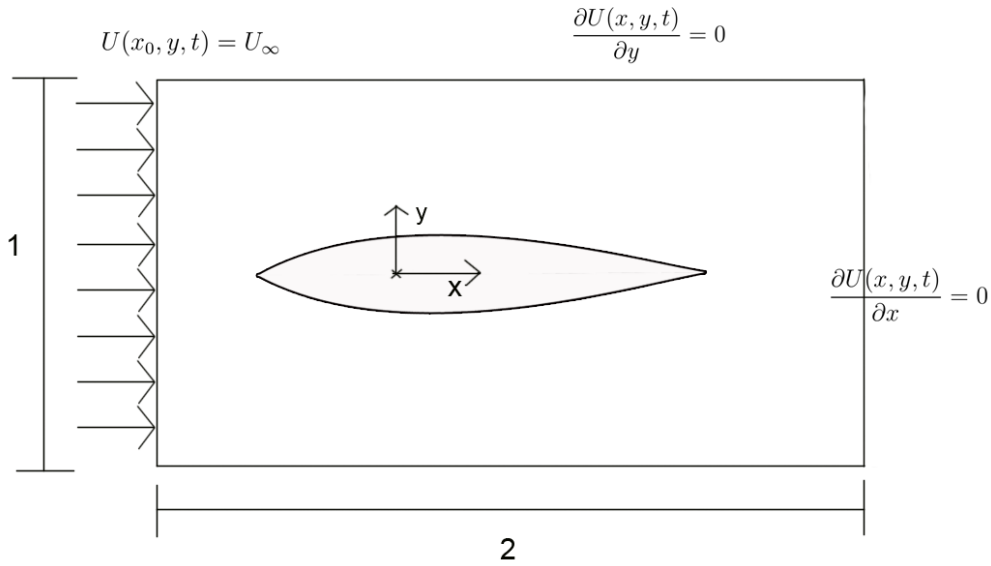


Figure 2: Rectangular domain for flow over an airfoil.

# 3 Numerical Discretization

## Spacial Discretization

For this project the finite volume method will be used for spacial discretization. This method has the ability to handle any shocks in the domain as well as to give fairly good stability properties combined with the third order TDV (total variation diminishing), Runge-Kutta 3 [3], which will be used in time.

The method is based on the calculation of fluxes in x- and y-direction. To make sure the method runs quickly in MATLAB, all the fluxes in each direction are calculated separately, before they are combined, defining the residual that is used at each stage of the Runge Kutta method.

To get the fluxes more accurate, the MUSCL scheme is used, which gives a higher order in space, but also adds the requirement of double ghost points. Here are the formulas:

**MUSCL scheme:** [13]

$$U_{j+\frac{1}{2}}^L = U_j + \frac{1}{2} \minmod(U_{j+1} - U_j, U_j - U_{j-1}) \quad (3.1)$$

$$U_{j+\frac{1}{2}}^R = U_{j+1} - \frac{1}{2} \minmod(U_{j+1} - U_j, U_{j+2} - U_j) \quad (3.2)$$

where minmod limiter performs this comparison:

$$\minmod(a, b) = \begin{cases} a & \text{if } ab > 0 \text{ and } |a| \geq |b| \\ b & \text{if } ab > 0 \text{ and } |b| \geq |a| \\ 0 & \text{if } ab \leq 0 \end{cases} \quad (3.3)$$

**Local Lax-Friedrich method:** [5]

Here matrices with the flux values in x- and y-direction are created from the U matrix. The left and right values are already calculated from it's own four neighboring points, as is seen in the MUSCL scheme above, and is then used to calculate the flux in the current face.

$$F_{i+\frac{1}{2},j} = \frac{1}{2}[F(U_{i+\frac{1}{2},j}^L) + F(U_{i+\frac{1}{2},j}^R) - |a_{i+\frac{1}{2},j}|(U_{i+\frac{1}{2},j}^R - U_{i+\frac{1}{2},j}^L)] \quad (3.4)$$

$$G_{i,j+\frac{1}{2}} = \frac{1}{2}[G(U_{i,j+\frac{1}{2}}^L) + G(U_{i,j+\frac{1}{2}}^R) - |a_{i,j+\frac{1}{2}}|(U_{i,j+\frac{1}{2}}^R - U_{i,j+\frac{1}{2}}^L)] \quad (3.5)$$

where  $|a_{i,j}|$  is the maximum local wave speed that is defined as follows:

$$|a_{i+\frac{1}{2},j}| = \max((|u| + c)(U_{i+\frac{1}{2},j}^L), (|u| + c)(U_{i+\frac{1}{2},j}^R)) \quad (3.6)$$

$$|a_{i,j+\frac{1}{2}}| = \max((|v| + c)(U_{i,j+\frac{1}{2}}^L), (|v| + c)(U_{i,j+\frac{1}{2}}^R)) \quad (3.7)$$

And c is here the speed of sound:

$$c = \sqrt{\frac{\gamma p}{\rho}}, \quad \gamma = \frac{c_p}{c_v} = 1.4 \text{ (for air)} \quad (3.8)$$

The local Lax-Friedrich method is originally a one dimensional solver, but here it is used in two dimensions. Each of the fluxes are calculated separately, and are then combined to create the two dimensional residual. The formula for the combination of the one-dimensional parts reads as follows:

$$\frac{dU_{i,j}}{dt} = -\frac{1}{\Delta x \Delta y} [(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j})\Delta y + (G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}})\Delta x] \quad (3.9)$$

where the right hand side defines the residual matrix  $R_{i,j}$  that is being used in the time discretization.

The finite volume method (3.9) for all nodes in the domain defines the system of ordinary differential equations:

$$\frac{dU}{dt} = R(U) \quad (3.10)$$

## Numerical methods in time

The numerical method used to calculate the ODE system (3.10), is the third order TVD method (RK3)[7] due to its high order accuracy in time, and the stability domain achieved with this method. The numerical formulas are as follows:

$$U^{(1)} = U^n + \Delta t R(U^n) \quad (3.11)$$

$$U^{(2)} = \frac{3}{4}U^n + \frac{1}{4}U^{(1)} + \frac{1}{4}\Delta t R(U^{(1)}) \quad (3.12)$$

$$U^{n+1} = \frac{1}{3}U^n + \frac{2}{3}U^{(2)} + \frac{2}{3}\Delta t R(U^{(2)}) \quad (3.13)$$

Of course between each of these iterations, the residual needs to be recalculated, boundary values updated and ghost point values set.

### Stability analysis

The von Neumann stability analysis shows that the Courant-Friedrichs-Lewy condition needs to be fulfilled for the explicit Euler method without MUSCL. For the RK3 method with MUSCL it has been safe to use the stability condition:

$$\Delta t = \min_{(i,j)} \left( \frac{C}{\frac{|u_{i,j}|+c_{i,j}}{\Delta x} + \frac{|v_{i,j}|+c_{i,j}}{\Delta y}} \right) \quad (3.14)$$

with  $C = \frac{1}{2}$ , and where  $u$  and  $v$  are the local velocities in  $x$ - and  $y$ -direction respectively, and  $c$  is the speed of sound.

One can clearly see from (3.14) that small grid spacings  $\Delta x$  and  $\Delta y$  imply a small time step. That is possibly one of the biggest problems with this explicit method. However, a small time step is also required to handle the oscillation of the structure.

## 4 Ghost point treatment

One of the common problems with the Cartesian grid method is how to handle points that are located just inside the solid structure. For this a ghost point treatment is required. Ordinary ghost point treatment creates a lot of overhead due to interpolations. Therefore, for this project a simplified ghost point treatment will be used to allow the system to run faster. This ghost point treatment method is based on the simplified ghost point treatment methods used by Farooq, Müller and Skøien. [2, 1, 11] Using an explicit numerical scheme forces us to have small time steps according to the CFL condition cf. (3.14). This also correlates nicely with the small time steps required for the oscillations of the structure to avoid too large movements in one time step.

### Ghost Point Identification

Due to the oscillations of the structure a reconstruction of the structure is required for every time step. This makes the natural place to handle the ghost point treatment the same place as the structure is reconstructed. After construction and rotation of the airfoil, the ghost points are identified. The airfoil is split into four parts where the flagging is handled differently in each.

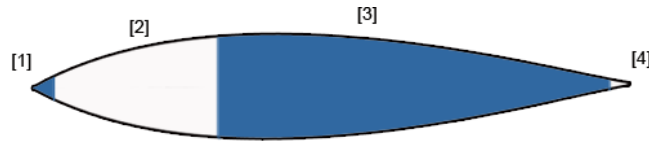


Figure 3: The four parts in which ghost points are handled differently.

**Part 1:** Due to the MUSCL scheme, two ghost points are needed in each coordinate direction. The first part that is handled, consists of the two first columns of the airfoil. Where columns here denote grid points in the y-direction. This is because no checks are needed to identify if the solid points should be set as ghost points or not, and the normal to the airfoil surface is close to vertical. This makes it possible to know that the best direction in which to find the closest airfoil surface point is directly to the left. The coupling with fluid points is based on the tangent of the airfoil surface in the y-coordinate of the ghost point being treated. If the angle between the surface tangent and the x-axis is close to vertical, a horizontal point is chosen, and if it's closer to 45 degrees a diagonal point is chosen. Each of the columns is gone through from top to center on the upper half of the airfoil and from bottom to center on the lower half of they-grid lines.

**Part 2:** The next part of the airfoil contains all grid points from the third grid point to the grid point around which the airfoil is rotating (0.25 chord length). Here most of the points are identified by directly setting the two first solid points in each column from the top or bottom as ghost points. However, due to the

steepness of the beginning of the airfoil more than two ghost points from the top and bottom may be needed. A separate check is here used to make sure all these cases are noticed and handled accordingly. The fluid point linking procedure in this part of the airfoil is as follows: For each of the first two ghost points in each column from top and bottom a check is performed, following a line in y-direction at the current x-coordinate until the airfoil surface is reached, where the normal for the surface is acquired. For the points located closer to the center of the airfoil, where the surface is steeper, a line in the x-direction is followed on the current y-coordinate until the airfoil surface is reached, where a similar check is performed to acquire the surface normal. When the surface normal of the surface point closest to the current ghost point is acquired, the fluid point is chosen based on this angle whether there already is a ghost point flagged between this ghost point and the fluid in this direction or not. The directional alternatives for this part are horizontal, diagonal or vertical faced in the direction from which the flow is coming.

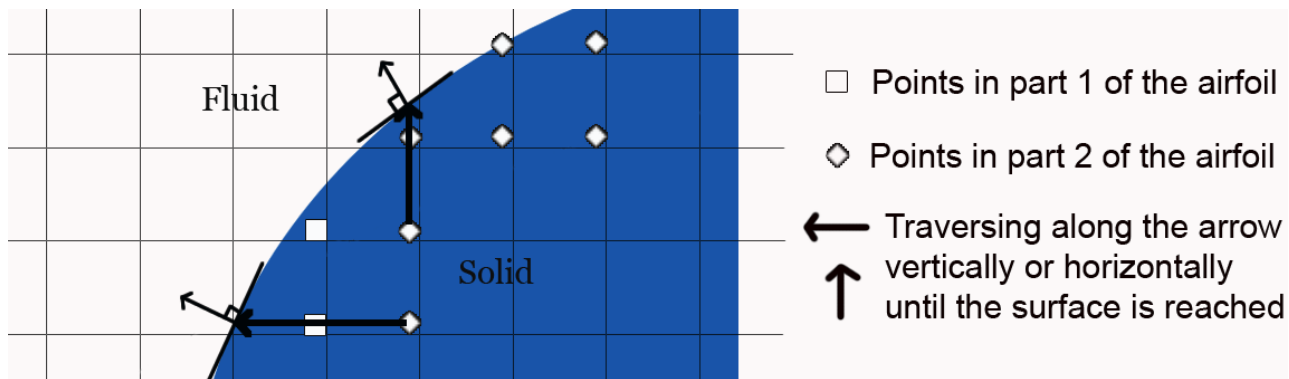


Figure 4: Assignment of angles at different ghost points.

In figure 4 one can see the assigning of the surface normals closest to each ghost point. The arrows indicate how the grid lines are followed until the airfoil surface is reached. The square points are located in part one, and get their surface normal by following the x-grid line. The diamond marked points are in part two, and follow the y-grid line instead. However, some points in part two follow the x-axis as on the square points. These are located as the third or lower point from the top, and as they will reach the airfoil surface sooner if moving on the x-axis, they should also use this method. A point like this can be seen in the figure; the diamond marked point which still moves to the left to find its closest airfoil surface point. (The diamond marked point where the lowest arrow starts.)

**Part 3:** The main part of the airfoil is from the point of rotation at a quarter chord almost until the very end. In this part one can assume that the airfoil surface is not very steep. This allows for direct flagging as ghost points the two first solid points encountered on each column when moving from top to center, and from bottom to center. Here the angle between the surface and the x-axis is also close

to zero, which makes all ghost points' closest airfoil surface angles dependent on the x-coordinates at which the points are located. The fluid point link is done in a very similar way as the previous part, but the possible directions in which the suitable fluid points for each ghost point may be located is limited to 45 degrees in each direction as well as vertically.

**Part 4:** The final part is the most advanced ghost point identification. Since the width of the airfoil approaches zero at the tail the airfoil geometry portrays an area where the width is less than four cells. The place where the width is only one cell, the MUSCL scheme cannot be utilized because the fluid points above the airfoil would get values from the points below and vice versa. Therefore the tail is cut in such a way that the tail ends with a width of two cells. To obtain an accurate calculation on this part of the airfoil the calculation of the residual needs to be based on the ghost points being linked to the fluid points above the airfoil when calculating the upper part, and the fluid points below the airfoil when calculating the lower part. This leads to the use of two matrices with ghost point values for this part of the airfoil. When the area below the tail of the airfoil is being calculated, the ghost points from the separate matrix are used, while the rest of the system is being calculated from the main matrix. Except for the storage in two separate matrices this part of the ghost point treatment behaves exactly like part 3.

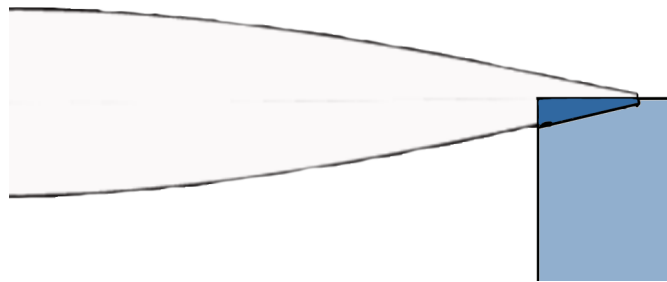


Figure 5: Separation of the tail matrix.

In figure 5 the domain which is treated separately is shown in blue, where the part of the structure is dark blue. When the ghost points and the linked fluid points have been identified, the values in the ghost points can be set.

## Ghost Point Values

For stationary structures the mirroring of the fluid point values onto the ghost point follows a symmetrical boundary condition which makes it possible to embed a solid boundary into a Cartesian grid. The values are updated according to the

following conditions:

$$\rho_G = \rho_F \quad (4.1)$$

$$u_{G,tang} = u_{F,tang} \quad (4.2)$$

$$u_{G,norm} = -u_{F,norm} \quad (4.3)$$

$$p_G = p_F \quad (4.4)$$

where G and F denote ghost and fluid points respectively, and tang and norm are the tangential and normal velocity components respectively. The density, pressure and tangential velocity component are set equal at the ghost point and the fluid point, whereas the normal velocity component at the ghost point is set to the negative value at the fluid point.

To determine the normal and tangential components of the velocity, the tangent of the structure between the ghost point and the fluid point needs to be identified. This can be done by differentiating the formula used to define the airfoil (2.11) at the x-position of the airfoil surface closest to the ghost point, which will yield the slopes of the tangent at this point. The identification of the surface point closest to the ghost point, is done similarly to what was done in the ghost point identification. The direction in which the fluid point is located compared to the ghost point is followed until the airfoil surface is reached, and the surface normal tangent is assigned from the differentiated airfoil surface formula. The angle between the tangent and the x-axis can then be calculated directly.

However, at the very leading edge of the profile, the face is almost completely vertical, and another approach must be used. Here the angle is set by moving on the current y-coordinate towards the front of the airfoil until the surface of the airfoil is reached. At that point the x-coordinate is registered. This x-coordinate is then inserted into the differentiated formula of the profile and an accurate angle will be acquired. After the angle has been found, the tangential and normal vectors for the face can be calculated, and then the tangential and normal velocity components set.

This procedure continues through all the ghost points in the structure, and ghost point coordinates and the coordinates of the fluid point each ghost point is linked with is stored in matrices to use for the updates. Between each stage of the Runge Kutta 3 method, the ghost point values need to be updated but the structure is not rebuilt. Therefore, the ghost point and ghost point link lists are gone through and their values are updated.



## Oscillation Handling

When the structure is oscillating, the way the values in the ghost points are updated needs to be slightly changed. When the structure moves, the fluid is also "pushed" in the direction that the airfoil is moving. To account for this the displacement angle is needed, as well as the distance from the rotation point along the airfoil central axis. Then, the velocity of the structure can be determined. That velocity vector is calculated, to determine its normal component. Then the following conditions are obtained:

$$\rho_G = \rho_F \quad (4.5)$$

$$p_G = p_F \quad (4.6)$$

$$u_{G,tang} = u_{F,tang} \quad (4.7)$$

$$u_{G,norm} = -u_{F,norm} + 2u_{S,norm} \quad (4.8)$$

where  $u_{S,norm}$  the normal velocity component of the point on the structure edge between the ghost point that is getting updated and the corresponding fluid point. S here stands for the solid surcafe of the airfoil.

Symmetry conditions with respect to the solid boundary of the structure are used as for a stationary structure, cf. (4.5-4.8). However, the normal velocity component at the ghost point is determined such that the average of its value and the normal velocity component at the closest fluid point corresponds to the normal velocity component of the structure boundary at the intersection between ghost and fluid point.

From the equation of state for perfect gas (2.9), we get:

$$\rho E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) \quad (4.9)$$

which results in the expression for  $(\rho E)_G$  being:

$$(\rho E)_G = \frac{p_F}{\gamma - 1} + \frac{1}{2}\rho_F(u_{F,tang}^2 + (-u_{F,norm} + 2u_{E,norm})^2) \quad (4.10)$$

When using a Cartesian grid method for an oscillating airfoil a temporal discontinuity will occur for fluid points located close to the boundary that were solid points in the previous time step [6]. These points most likely were ghost points before they were moved out of the structure completely, therefore lacking relevant values. In any case, the release of each of these points creates problems for the numerical method because they do not have proper values and should be handled. The method to solved this here is as follows: First the fluid points where this is a problem must be identified. This is done by going through the lists of ghost points on the sides of the airfoil where release of points may happen based on the current rotational direction of the airfoil. These ghost points are then calculated back to their position in the previous time step by use of a reversed rotational scheme. If

this location is now flagged as a fluid point, it means that the values at this point are those it was given when being a ghost point, and needs to be updated to fit in with the other fluid points in the area.

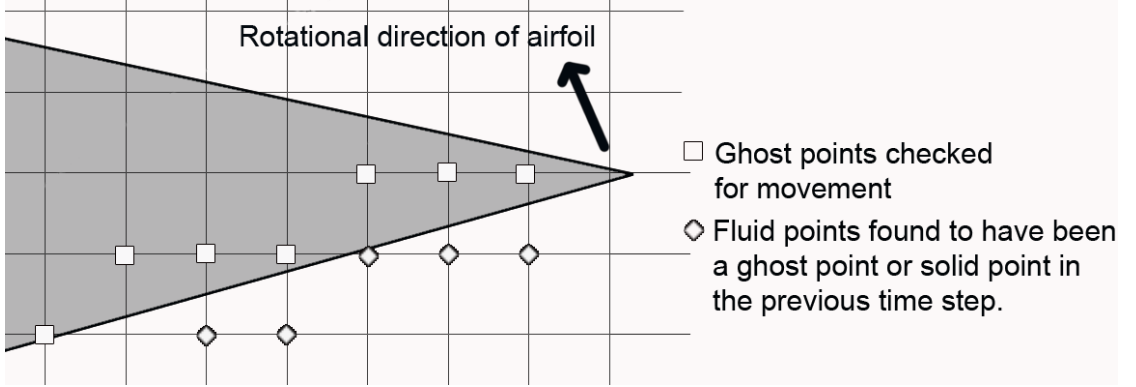


Figure 6: The rollback method of the ghost points.

In figure 6 the rollback method of the ghost points is shown. Not all of the points have moved a complete cell since the last time the airfoil was moved, resulting in not all of the points requiring to have their values updated. Figure 7 shows an enlargement near the trailing edge. The update itself is done similarly to the ghost point value updates:

$$\rho_{new} = \rho_N \quad (4.11)$$

$$p_{new} = p_N \quad (4.12)$$

$$u_{new,tang} = u_{N,tang} \quad (4.13)$$

$$u_{new,norm} = \frac{1}{2}(u_{surface,norm} + u_{N,norm}) \quad (4.14)$$

where  $u_{new,tang}$  and  $u_{new,norm}$  are the tangential and normal velocity components of the fluid point being updated.  $u_{surface,norm}$  and  $u_{N,norm}$  are the normal velocities of the surface of the airfoil and the point. Here the expression for  $(\rho E)_{new}$  is also a bit more advanced. We have the expression for  $(\rho E)$ , using (4.11-14) in (4.5):

$$(\rho E)_{new} = \frac{p_N}{\gamma - 1} + \frac{1}{2}\rho_N(u_{N,tang}^2 + (\frac{1}{2}u_{surface,norm} + u_{N,norm})^2) \quad (4.15)$$

The reason why this is an important step is because to perform the numerical calculation and get the correct values at the ghost point, the new fluid point values denoted by subscript new, need to have been set. However, the ghost point values will not be mirroring the perfect symmetrical boundary conditions if the new fluid point has not yet had its values calculated.

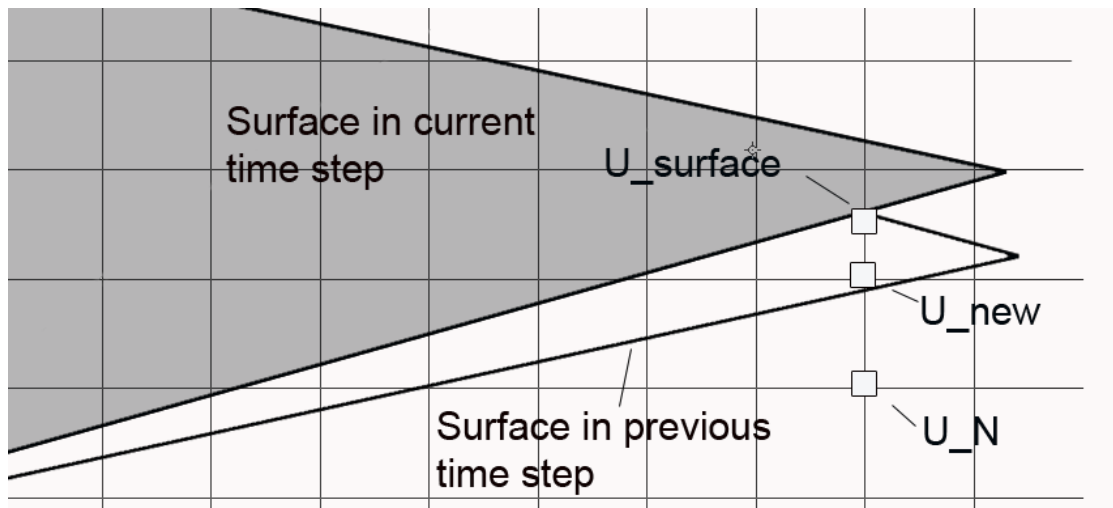


Figure 7: Notation of the points used in the discontinuity updating.

In Figure 7, the different points used in the calculations of the emerging points are shown. The dark structure is the airfoil at this time step, while the white structure is where the airfoil was in the previous time step.  $U_N$  here denote  $U_{\text{neighbor}}$ , and will be either above of below the point being calculated based on if the airfoil is moving upwards or downwards.

## 5 The Program

The program has been implemented in MATLAB, with a procedure oriented approach. Different parts of the program are divided into functions, making them possible to be called when needed. The main program controls the flow in the program and initiates the other methods as it progresses through the code. Finally the results are displayed in a separate function.

### Functions

#### Main

The main program keeps the control in the program flow. It is where system values as grid spacing and max time are set, it also runs the setup method. The main program contains the loop which runs through all the time steps, and calling the required methods as they are needed. The main also performs checks to make sure stability is kept with the CFL check method, calculating the time step for each time step.

#### Set Up

The setup method creates most of the matrices needed in the flow of the program. It also sets up constants and calculates values that are only required to be calculated once at the startup of the program.

#### Draw Airfoil

The flag map, containing flags indicating what type of material or state every block/point of the system contains, is created by this method. The structure is then sent to the rotate Airfoil method which rotates the airfoil to the current position the structure has using the oscillation formula set.  $(\sin(\omega))$  The flag map is then returned to the Move Structure method.

#### Rotate Airfoil

This method acquires an unrotated airfoil and the angle to which it should be rotated. The positions are updated for each point of the structure according to the new angle and the flag map is returned to the draw airfoil method.

#### CFL test

This method performs a test giving the smallest possible time step that still fulfills the Courant-Friedrichs-Lewy condition.

#### Move Structure

This method calls the Draw Airfoil method, before identifying the ghost points in the structure it is given. For each ghost point identified the set GhostValues method is run, updating the values in these points according to the parameters set by the Move Structure method.

### **One StepRK3**

This is the method handling the time discretization. It acts in some way as its own main program, because it has few calculations of its own. The function basically calls the other methods based on the RK3 scheme.

### **set GhostValues**

Initially, for every time step and for every stage in the RK3 method, the values at the ghost points needs to be updated. That means that there are several different methods that use the set GhostValues method to get their ghost point updated with its correct value. This method performs that action.

### **Get Residual**

For the RK3 method to work, a residual is needed. This method handles the acquiring of that residual. It has other methods calculating the fluxes, this method only gathers them into the residual ready for use by the RK3.

### **Boundary Treatment**

Just like with the set GhostValues method, the boundary points' values need to be frequently updated. This method contains the conditions for all the boundary types and applies the new values at the boundary according to them.

### **Discontinuity Check**

This method goes through the ghost points in the parts of the airfoil that may release points with the current rotational direction. The details of this method is explained in part 4: Oscillation Handling, and makes sure points with potential discontinuities are updated to correct values.

### **Flux Calculations (X/Y)**

This program is written for a 2D problem, with some numerical methods originally intended for 1D problems. Therefore some methods require separate methods for x- and y-direction. Especially the fluxes needs to be calculated in one direction at the time, to make the program run smoother. The flux calculations methods use methods like MUSCL and getInternal to make a matrix with the calculated fluxes in all the grid points of the system.

### **MUSCL (X/Y)**

The MUSCL method is another method that requires both an x- and y-version. It follows the formula of the MUSCL scheme, and calculates the left and right flux values required by the flux calculation methods. The MUSCL methods call the minMod method for a part of their calculations.

### **minMod**

The minMod method has a very small task; performing the minMod comparison explained in the spacial discretization part earlier,(cf. equation 3.3).

### **get Internal(X/Y)**

The get internal method is a method compiling and isolating all the parts of the flux calculations that require handling the primitive variables.

### **TailCopy**

This method overwrites values back and forth between the main system matrix and the lower tail matrix so that the numerical methods may function as if the system is not split at all. This method is run between each step of the RK3 method to make sure the values are as they should at all times.

### **Display Values**

The display values method receives the finished system matrices, derives primitive variables and displays these in different types of plots, making it easier for the user to analyse.

## Program diagram

Figure 4 shows the functions used in the program and what other functions they interact with:

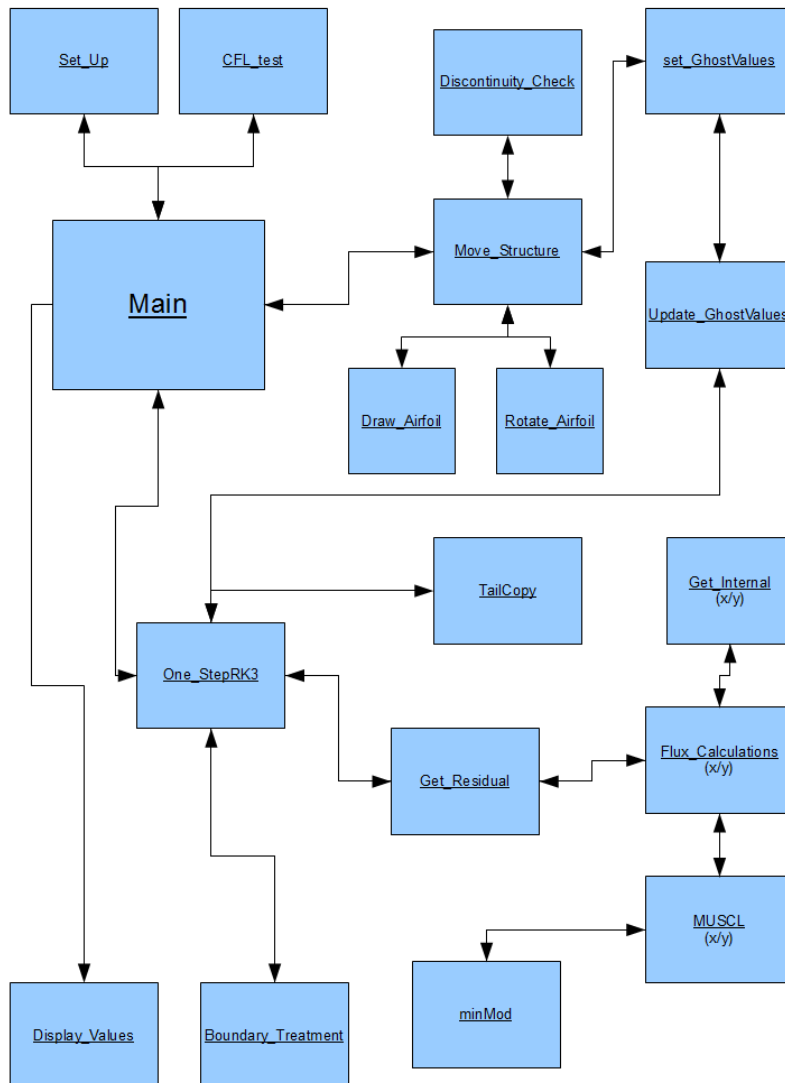


Figure 8: Function overview.

## Flowchart

Figure 5 shows the flow of the program, what is done when and where:

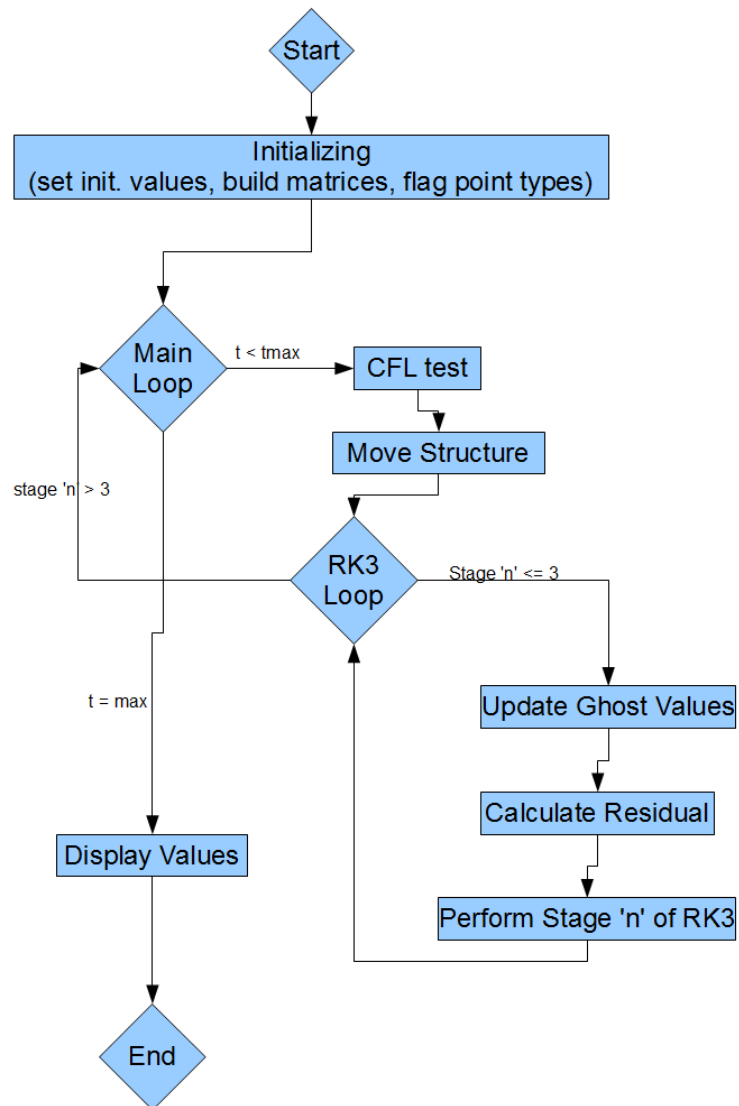


Figure 9: Flowchart.



## 6 Results

The code used for the results is a modified version of the code used in the master thesis work of Are Skjøien [9]. The ghost point treatment is fairly similar to the one described in this article, as well as the numerical methods used. The

Simulations have been done for verification, in addition to some general plots of solutions. Comparisons are done with results from the work by Schneiders et al. [9], a transonic case for the NACA0012 with small oscillations at angle of attack  $\alpha(t) = 0.016 + 2.51\sin(\omega t)$  degrees, where the reduced frequency is  $\frac{\omega c}{(2u_\infty)} = 0.0814$  with  $c$  being the chord length.

The freestream Mach number, density and pressure are chosen as  $M_\infty = 0.755$ ,  $\rho_\infty = 1.2070$ ,  $p_\infty = 100100$  respectively. The domain spans 4 meters in x-direction and 2 meters in the y-direction, and is discretized by 301 and 151 points in the x- and y-directions respectively. The point around which the rotation is based, is located at a quarter chord from the leading edge. I.e. at  $x = \frac{c}{4}$  with the leading edge at the origin. The simulation was run for 75000 time steps.

First a quick visualization of the system:

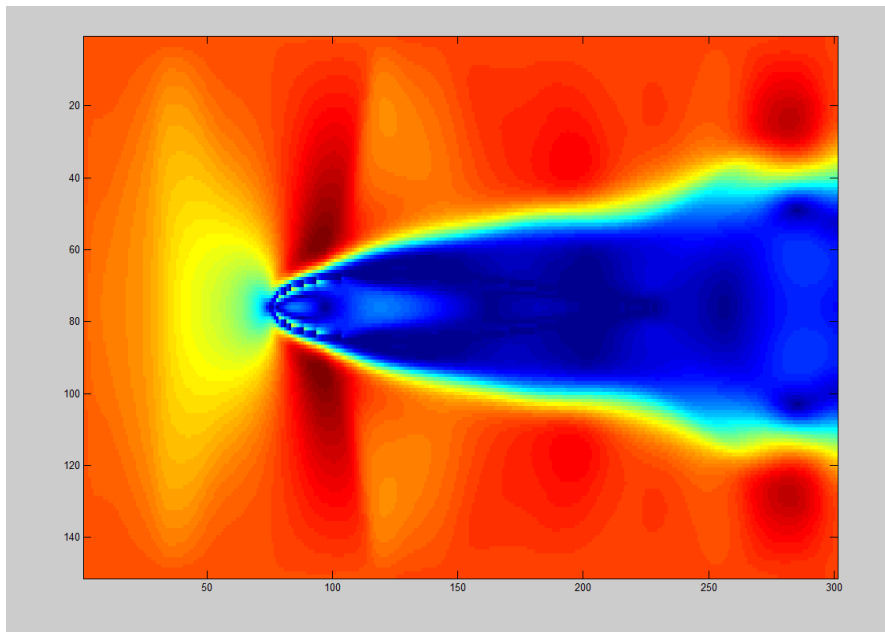


Figure 10: NACA0012 - at  $M_\infty = 0.755$ ,  $\alpha = 0$  degrees. - 301 x 151 grid.

Figure 10 shows Mach contour lines of the system after a simulation has been run. It may be a bit difficult seeing the airfoil, it is only outlined by a slight shadow. The angle of attack here is zero degrees, and this is after the simulation has run through several oscillations for a prolonged time.

In figure 11 the lift coefficient is plotted against the angle of attack  $\alpha(t)$ . The data for the  $C_L$  values was too much to store everything of, so only the part with a positive angle of attack is shown, but it gives an impression in any case.

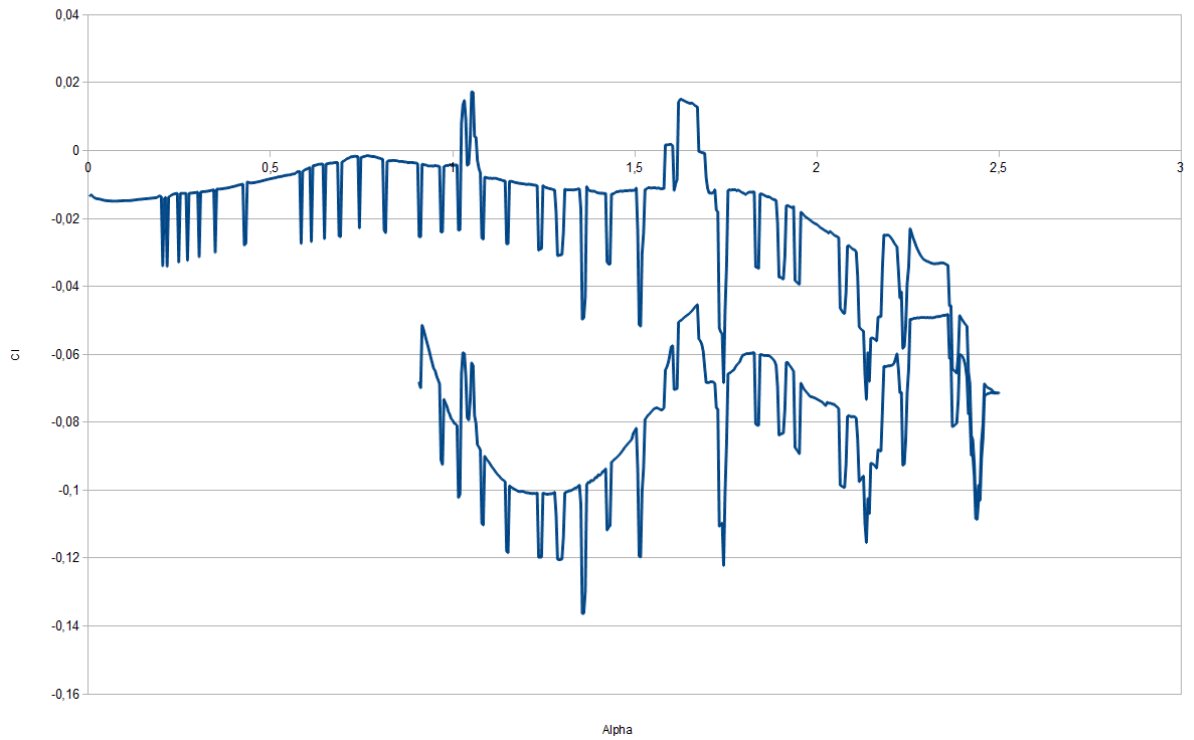


Figure 11:  $C_L$  versus angle of attack  $\alpha(t)$  - 301 x 151 grid for NACA0012  $M_\infty = 0.755$ .

To compare with results from the literature, a similar type of plot is shown. This is taken from the article of Schneiders et. al. [9]:

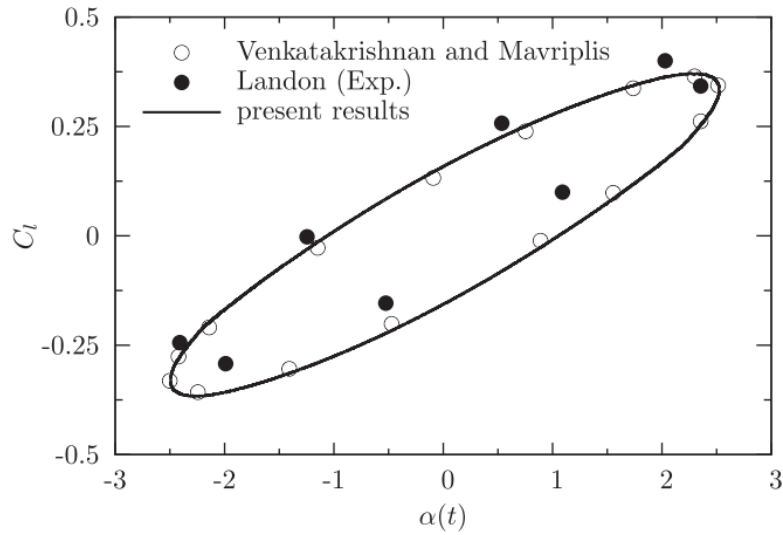


Figure 12:  $C_L$  from [9] for comparison.

As one can see from this comparison, the  $C_L$  values are not quite as expected. The values are alternating massively, and are too low in magnitude. Still if the alternations of the values is neglected a curve can be seen, which shows a kind of pattern.

To see if the solution would improve with more grid points without having to run longer simulations, a test run was done for fewer grid points, to see if an even worse solution was obtained then. Figure 12 shows the domain, where a different color scheme has been used to allow for the airfoil to be seen easier. The jaggedness of the surface due to a lower number of grid points can be seen very clearly here.

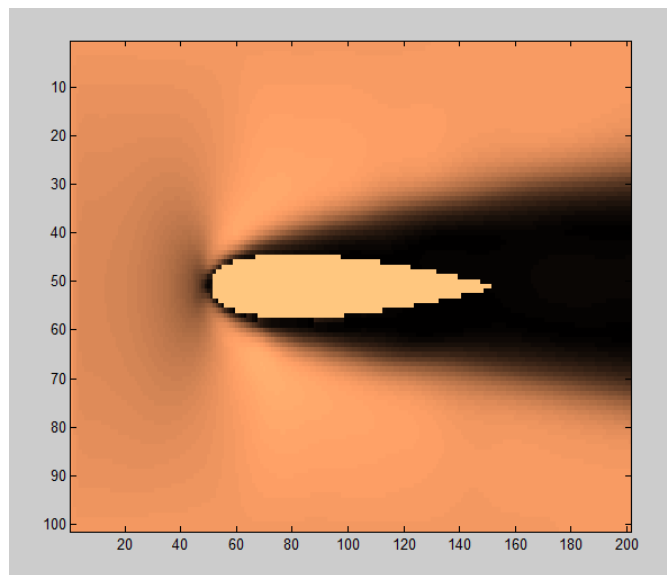


Figure 13: NACA0012 - at  $M_\infty = 0.755$ ,  $\alpha = 0$  degrees. - 201 x 101 grid.

Figure 13 shows a plot for the lift coefficient  $C_L$  against the angle of attack  $\alpha(t)$ . Mach number, density and pressure is equal to that of figure 10, but the number of grid points is reduced.

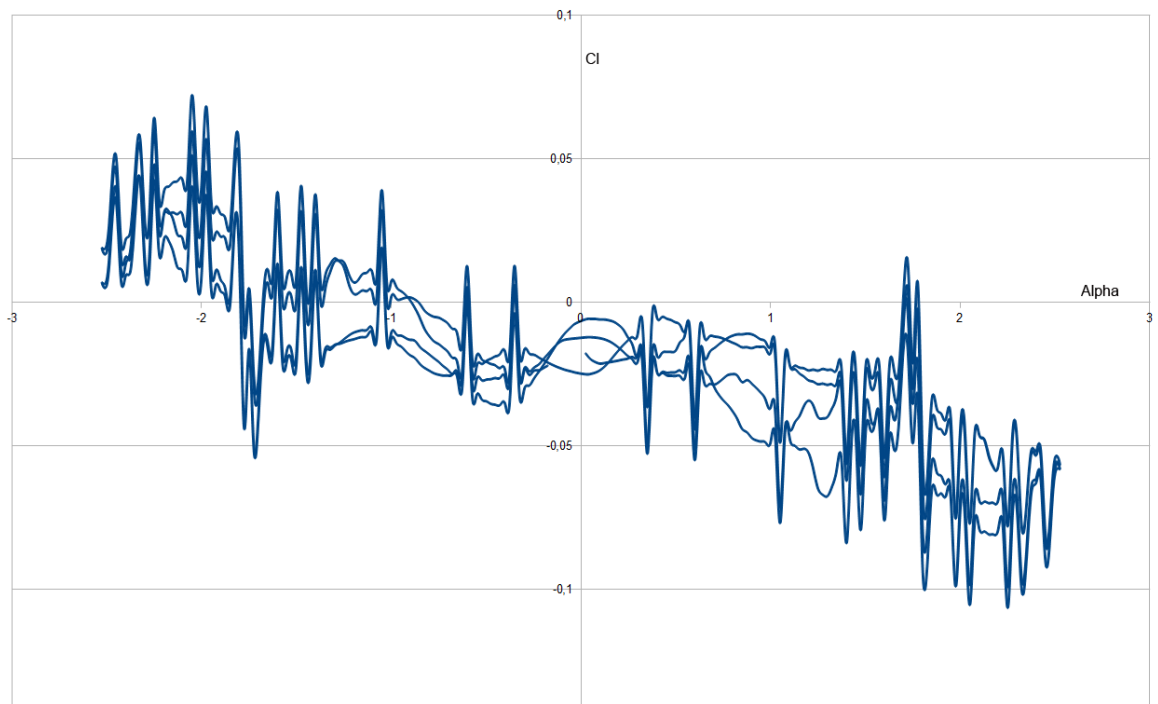


Figure 14:  $C_L$  versus angle of attack  $\alpha(t)$  - 201 x 101 grid for NACA0012  $M_\infty = 0.755$ .

By comparing figure 11 and figure 13 it becomes clear that the small increase in grid points from 201 x 101 to 301 x151 improved the solution very much. What can be seen in figure 14 is a complete mess with few patterns to be seen at all. It is not possible to conclude that increasing the amount of grid points absolutely will give perfect results eventually, but it is possible to assume that a better solution can be acquired by the use of more grid points.

The pressure coefficient at for a pitching airfoil. The angle of attack  $\alpha(t) = 2.5$  degrees. This is for a stationary case where the simulation was run for 20000 time steps before the results were obtained. were also acquired for  $C_p$  in the 201 x101 test case:

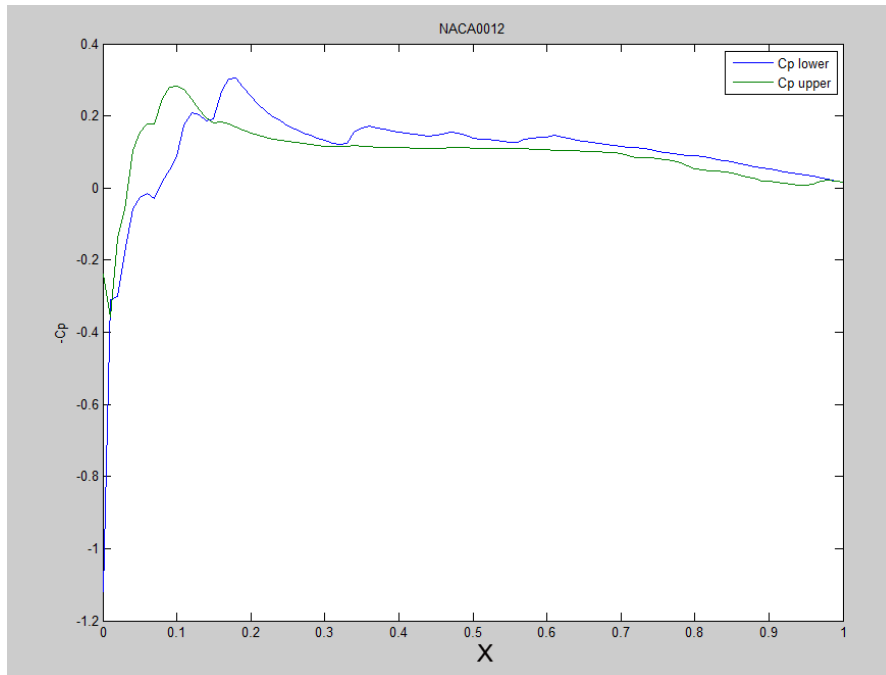


Figure 15: Pressure coefficient for NACA0012 at  $M_\infty = 0.755$  and  $\alpha(t) = 2.5$  degrees.

For comparison with the results from the work of Schneiders et. al [9], a  $C_p$  plot is provided. Here the results from the work of Kirshman and Liu [4] is also visible on the plot.

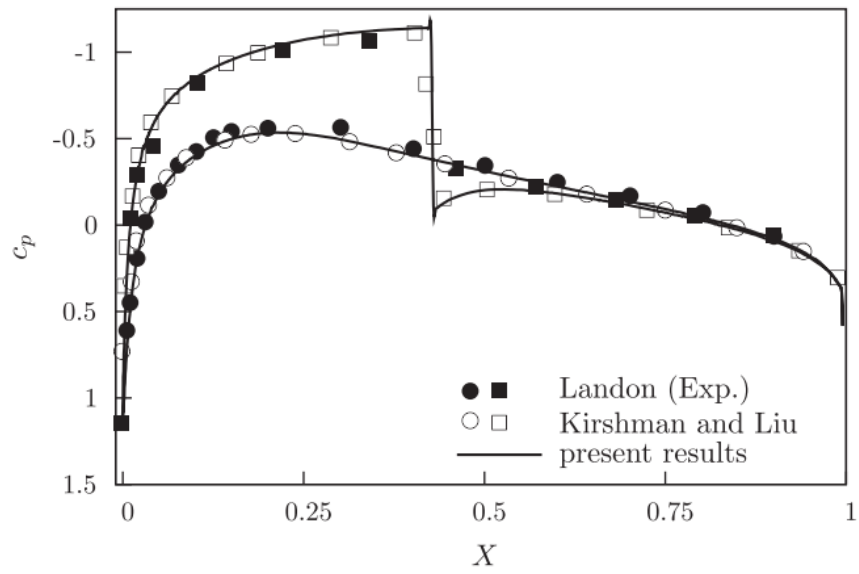


Figure 16: Pressure coefficient for NACA0012 at  $M_\infty = 0.755$  and  $\alpha(t) = 2.34$  degrees. From [9] for comparison.

When comparing the results in the figures 15 and 16, one can see that the pressure is predicted too high (i.e.  $-C_p$  too low) near  $x = 0.15$ .

Figure 17 shows the convergence history for the steady state solution at  $\alpha = 0.016$ . The value on the vertical axis is  $\|\rho^{n+1} - \rho^n\|_2$

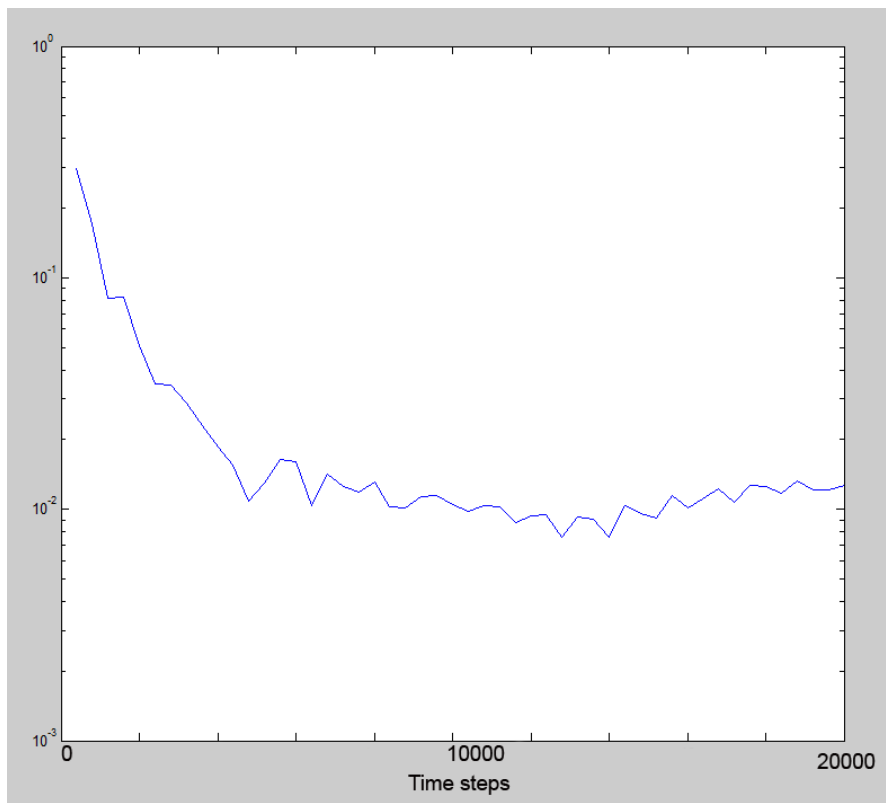


Figure 17: Convergence history for NACA0012,  $M_\infty = 0.755$ ,  $\alpha = 0.016$  degrees - 201 x 101 grid.

As one can see, some convergence is gotten after only a part of the total time steps. However, complete convergence is not obtained even after 20000 iterations.

## 7 Conclusions and Outlook

The future looks bright for the Cartesian grid method. Versatility is one of its main features, which is exactly what is needed in the time to come. Being able to solve for complex geometries, and make quick alterations to these is what makes it so interesting. The method has seen a tremendous development the last ten years as problems with run times and movement are being solved or provided alternative solutions to frequently. It is a challenging method to develop, but scientists have shown that it is indeed a method worth working on.

The methods described for ghost point treatment for oscillating airfoils was not completely verified, since not all of them were implemented in the program used for results in same way as is described in this article. It would be very interesting seeing a later work implementing some of these methods in a functioning program. The results showed that the methods used may not be the best suited for transonic flow, however through grid refinement better solutions and results could be acquired even for transonic cases.

## 8 Acknowledgements

The attempt of combining many of the different recently developed solutions in one program has for me not went as well as I had originally thought. Still, a lot was learned, and with some help, results were had. Last year's master student Are Skøien's program was helpful, and the excellent supervision from my advisor Bernhard Müller helped tremendously. I would like to thank them both greatly for what they have contributed with to this project.



# References

- [1] M.A.Farooq,  
*Cartesian Grid Method for Compressible Flow Simulation.*  
Doctoral Thesis: Norwegian University of Science and Technology, Trondheim, 2012
- [2] M. A. Farooq and B. Müller  
*Accuracy of the Cartesian Grid Method for Hyperbolic Conservation Laws Using Standard and Simplified Ghost Point Treatments at the Immersed Boundary.*  
Journal of Structural Mechanics, Volume 44, No. 3, Pages 279-291, 2011
- [3] S. Gottlieb and C-W.Shu,  
*Total variation diminishing Runge-Kutta schemes.*  
Mathematics of Computation, 67, Pages 73-85 1998
- [4] D.J. Kirshman and F.Liu  
*Flutter prediction by an Euler method on non-moving Cartesian grids with gridless boundary conditions.*  
Computers and Fluids, Volume 35, Issue 6, Pages 571-586, July 2006
- [5] R.J. LeVeque  
*Finite Volume Methods for Hyperbolic Problems.*  
Cambridge University Press 2002
- [6] W. Liao, E.P.C. Koh, H.M. Tsai, F. Liu  
*Euler calculations with embedded Cartesian grids and small-perturbation boundary conditions.*  
Journal of Computational Physics, Volume 229, Issue 9, Pages 3523-3542, May 2010
- [7] R. Mittal and G. Iaccarino,  
*Immersed Boundary Methods.*  
Annual Review of Fluid Mechanics, 37, Pages 239-261, 2005
- [8] C. S. Peskin,  
*Flow pattern around heart valves: A numerical method.*  
Journal of Computational Physics, 10, Pages 252-271, 1972
- [9] L.Schneiders, D.Hartmann, M.Meinke, W. Schröder  
*An accurate moving boundary formulation in cut-cell methods.*  
Journal of Computational Physics, Volume 235, Pages 786-809, February 2013
- [10] B. Sjögreen and N.A.Petersson  
*A Cartesian embedded boundary method for hyperbolic conservation laws.*  
Communications in Computational Physics, 2, Pages 1199-1219, 2007

- [11] A. Skøien,  
*Cartesian grid methods for the compressible Navier-Stokes equations.*  
Master Thesis: Norwegian University of Science and Technology, Trondheim,  
June 2012
- [12] H.S. Udaykumar, R. Mittal, W. Shyy.  
*Computation of solid-liquid phase fronts in the sharp interface limit on fixed grids.*  
Journal of Computational Physics, 153, Pages 534-74 1999
- [13] B. van Leer  
*Towards the Ultimate Conservative Difference Scheme, V. A Second Order Sequel to Godunov's Method.*  
Journal of Computational Physics, 32, Pages 101-136, 1979