



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Adaption of a two phase solver for axisymmetric problems

**Lars Liestøl**

Master of Science in Product Design and Manufacturing

Submission date: June 2012

Supervisor: Bernhard Müller, EPT

Co-supervisor: Claudio Walker, EPT

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



EPT-M-2012-58

**MASTER THESIS**

for

student Lars Liestøl

Spring 2012

Adaption of a two phase solver for axisymmetric problems

*Adapsjon av en tophase løser for aksesymmetriske problemer***Background and objective**

Applications where the behaviour of the impacting drops plays an important role include ink-jet printing, spray cooling, pesticide spraying. One of the difficulties arising in modelling droplet impacts is the moving contact line problem. We work on a multiscale model of the contact line. Our solver is currently implemented for 2D Cartesian problems. We would like to extend the capabilities of the solver to axisymmetric problems. The Master thesis involves the implementation of the necessary changes in the code.

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

1. Familiarisation with the solver code
2. Adaption for axisymmetric case for single phase problems
3. Adjustment of the jump conditions for axisymmetric case
4. Validation of the code with suitable test cases

-- " --

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.

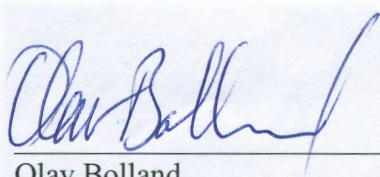
The candidate is requested to initiate and keep close contact with his/her academic supervisor(s) throughout the working period. The candidate must follow the rules and regulations of NTNU as well as passive directions given by the Department of Energy and Process Engineering.

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.

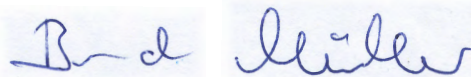
Pursuant to "Regulations concerning the supplementary provisions to the technology study program/Master of Science" at NTNU §20, the Department reserves the permission to utilize all the results and data for teaching and research purposes as well as in future publications.

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.

Department of Energy and Process Engineering, 16. January 2012



Olav Bolland  
Department Head



Bernhard Müller  
Academic Supervisor

Research Advisors:  
Claudio Walker

## Abstract

# Abstract

This report documents the adaptation of a two dimensional two phase Navier-Stokes solver to axisymmetric problems. The changes from Cartesian to cylindrical coordinates are thoroughly described with finite difference methods for the heat equation, Poisson equation, single and two phase Navier-Stokes equations. The jump conditions at interfaces are modified to accommodate these changes for the two phase Navier-Stokes equations.

The changes to the solver are done step by step, and every change is verified through intermediate test cases with analytical solutions to limit the possible sources of errors.

Finally all stepwise changes are joined together to form an axisymmetric two phase Navier-Stokes solver. Results are presented for a resting bubble, and for both viscous and inviscid oscillating elliptic bubbles.

# Sammendrag

Denne rapporten dokumenterer endringen av en todimensjonal to-fase Navier-Stokes-løser til aksesymmetriske tredimensjonale problemer. Endringene fra et kartesisk til et sylindrisk koordinatsystem er grundig beskrevet med endelige differanse-metoder for varmeligningen, Poisson-ligningen, enfase og tofase Navier-stokes ligninger. Hopp er modifisert til å ta høyde for disse endringene i tofase Navier-Stokes ligninger.

Endringene til løseren er gjort stegvis, og hvert steg er validert ved hjelp av egne problemer med analytiske løsninger for å begrense antall mulige feilkilder

Til slutt er alle delvise endringer satt sammen for å danne en tredimensjonal aksesymmetrisk Navier-Stokes løser. Resultater for en boble i ro, samt både viskøse og ikke-viskøse oscillerende elliptisk bobler.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Comparing studies . . . . .	2
<b>2</b>	<b>Governing equations</b>	<b>2</b>
<b>3</b>	<b>Numerical method for the two-dimensional problem</b>	<b>4</b>
3.1	Program structure . . . . .	4
3.2	Interface tracking . . . . .	5
3.3	Grid . . . . .	6
3.4	Discretization . . . . .	7
3.5	Ghost Fluid Method . . . . .	9
<b>4</b>	<b>Extention of numerical methods for axisymmetric problems</b>	<b>12</b>
4.1	Viscous terms . . . . .	13
4.2	Poisson equation . . . . .	13
4.3	Ghost fluid method . . . . .	14
4.4	Curvature . . . . .	16
<b>5</b>	<b>Verification of sub-tasks</b>	<b>16</b>
5.1	Two-dimensional Cartesian heat conduction problem . . . . .	17
5.2	Verification of the two-dimensional heat conduction problem . . . . .	17
5.3	Axisymmetric heat conduction problem . . . . .	19
5.4	Verification of the axisymmetric heat conduction problem . . . . .	19
5.5	Verification of the Poisson equation . . . . .	21
5.6	Verification of an axisymmetric single phase Navier-Stokes solver . . . . .	22
5.7	Verification of the Ghost Fluid Method . . . . .	23
<b>6</b>	<b>Results</b>	<b>24</b>
6.1	Spherical bubble at rest . . . . .	24
6.2	Oscillating elliptic bubble . . . . .	26
<b>7</b>	<b>Conclusions and outlook</b>	<b>29</b>
	<b>References</b>	<b>32</b>

## List of Figures

1	Illustration of coordinate system . . . . .	3
2	Illustration of an interface and normal vector $\mathbf{n}$ . . . . .	4
3	Illustration of the staggered grid layout . . . . .	7
4	$u(x)$ and $\phi(x)$ . . . . .	9
5	Subcell fractions. . . . .	10
6	Stencil for the pressure Laplacian. . . . .	14
7	Grid convergence for two-dimensional heat equation. . . . .	19
8	Grid convergence for axisymmetric heat equation . . . . .	20
9	Deviation from analytical solution on 128x128. . . . .	21
10	Grid convergence for axisymmetric Poisson equation . . . . .	22
11	Flow distribution from simulation and deviation from the analytical solution. . . . .	23
12	Convergence rate of the GFM in the viscous case. . . . .	24
13	Stable spherical bubble. . . . .	25
14	Maximum velocity in the domain. . . . .	26
15	Elliptic bubble with viscous dampening. . . . .	27
16	Inviscid oscillating bubble. . . . .	29





# 1 Introduction

This master thesis is the result of my work on my master's project in computational fluid dynamics. The thesis is a continuation of my project thesis from the fall of 2011. The goal of this work is to extend the capabilities of an existing two-dimensional Navier-Stokes solver for multiphase flow simulation to being able to handle axisymmetric problems. This means that the solver will have to be able to compute problems in x- and y-directions as well as in axial and radial directions.

The biggest difference between the two problems is how the Laplace operator is evaluated. In the axisymmetric problem, the circumference of the cylinder increases as the radius increases. This has to be accounted for when solving these problems.

To make this task a manageable one, it was divided into smaller parts:

- Solution of the two-dimensional heat equation using the existing solver.
- Extension of the existing solver to the axisymmetric heat equation.
- Extension of the existing two-dimensional Poisson solver for the pressure to the axisymmetric case.
- Extension of the existing two-dimensional single phase solver to an axisymmetric single phase solver.
- Extension of the jump conditions for the jump at interfaces between two phases from two dimensions in the existing solver to develop an axisymmetric two phase Navier-Stokes solver.

This report will explain the basis of the existing solver to give a basic understanding of what has been done as well as how challenges have been dealt with. Following a short literature review of axisymmetric multiphase flow simulation, the axisymmetric Navier-Stokes equations and jump conditions at interfaces are presented in section 2. Section 3 describes the two dimensional solver which is to be adapted. A thorough description of the extension of the equations from two dimensional to axisymmetric problems is given in section 4. Verification of the stepwise adaptation of the solver is given in section 5. Section 6 provide results from test cases of the axisymmetric multiphase Navier-Stokes equation. Finally, 7 concludes the thesis with thoughts on the results and what may be done in the future.

## 1.1 Comparing studies

As a subject, multiphase flow adds notable complexion to computational fluid dynamics with the introduction of surface tension and discontinuities between phases [12]. Multiphase flow plays an important role in several industries such as the petroleum and process industry [1].

Axisymmetric fluid mechanics include a wide array of interesting cases and can be used to understand different phenomena in pipe flow, jets, and wakes among others.

The solver by Claudio Walker [13] that is worked on in this thesis focuses on the moving contact line problem. Applications of this problem include ink-jet printing, spray cooling and pesticide spraying [13]. Modeling these applications axisymmetrically can add a lot of interesting possibilities to solve problems that are closer to a three dimensional physical case, e.g. a droplet.

Jack Chessa and Ted Belytschko [2] describe a solver which simulates the two phase axisymmetric flow problem. They are using a level set method for interface tracking, but a quite different method of discretization than what has been done in this thesis work, namely the enriched finite element method.

Mark Sussman and Elbridge Gerry Puckett [11] use a similar discretization to what has been done in this thesis, but couple the level-set method with the volume-of-fluid method to achieve a conservative method for two phase flows. They highlight the biggest issue with the level-set approach as the loss of mass which can occur in certain situations.

## 2 Governing equations

We consider two phase fluid flow in axisymmetric cylindrical coordinates. The equations shown in this section are written such that all azimuthal derivatives and the velocity component in the azimuthal direction are neglected. This means that although all calculations are done in two dimensions,  $r$  and  $y$ , there is a third dimension represented by the azimuthal angle in which all  $r$ - $y$  planes are identical. Figure 1 shows the calculation plane oriented in the three dimensional coordinate system.

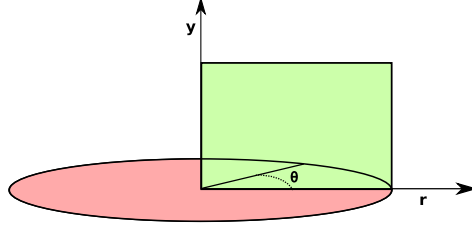


Figure 1: Illustration of coordinate system, calculations are done in the green  $r - y$  plane, and changes in the azimuthal angle  $\theta$  are neglected.

The radial and axial components of the momentum equations governing incompressible flow with constant viscosity and density reads:

$$\rho \left( \frac{\partial u_r}{\partial t} + (\mathbf{u} \cdot \nabla) u_r \right) = -\frac{\partial p}{\partial r} + \mu \left( \nabla^2 u_r - \frac{u_r}{r^2} \right) + \rho f_r \quad (1)$$

$$\rho \left( \frac{\partial u_y}{\partial t} + (\mathbf{u} \cdot \nabla) u_y \right) = -\frac{\partial p}{\partial y} + \mu (\nabla^2 u_y) + \rho f_y \quad (2)$$

The axisymmetric continuity equation for incompressible flow read:

$$\nabla \cdot \mathbf{u} = 0. \quad (3)$$

Here  $\mathbf{u} = [u_r, u_y]^T$  is the velocity vector with the radial and axial components  $u_r$  and  $u_y$ , respectively. The advection operator is  $\mathbf{u} \cdot \nabla = u_r \frac{\partial}{\partial r} + u_y \frac{\partial}{\partial y}$  and the Laplacian operator is  $\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{\partial^2}{\partial y^2}$ .  $\rho$ ,  $p$ ,  $\mu$ ,  $f_r$  and  $f_y$  are the density, pressure, viscosity, radial body force and axial body force respectively.

The jump at interfaces between two phases is governed by equations (4) and (5), where  $\mathbf{n}$  is the normal unit vector to the interface,  $\mathbf{t}$  is the tangent unit vector,  $\tau$  is the viscous stress tensor,  $\sigma$  is the surface tension, and  $\kappa$  is the local curvature of the interface:

$$\begin{bmatrix} \mathbf{n}^T \\ \mathbf{t}^T \end{bmatrix} (p\mathbf{I} - \tau)\mathbf{n} = \begin{bmatrix} \sigma\kappa \\ 0 \end{bmatrix}, \quad (4)$$

$$[u] = 0, \quad (5)$$

where  $[u] = u^+ - u^-$  is the difference in velocities in the two phases at the interface. These jump conditions must be taken into account when calculating the viscous terms as well as when solving the Poisson equation for the pressure. This is done by using the Ghost Fluid Method [6].

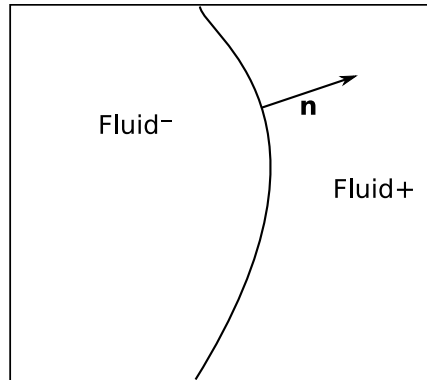


Figure 2: Illustration of an interface and normal vector  $\mathbf{n}$

### 3 Numerical method for the two-dimensional problem

The existing solver is a two-dimensional two phase Navier-Stokes solver written by my co-advisor Claudio Walker [13]. It is written in the programming language Fortran 90, which is a language widely used in engineering and numerical research due to its capabilities in computational speed.

#### 3.1 Program structure

The program itself is made up by several modules containing subroutines and functions called by the main program to perform all computations. The main program calls subroutines to initialize the simulation, to perform each time step and to write data. To start the simulations there is a separate program designed to determine the initial condition. This program basically writes starting values for velocity, pressure and the  $\phi$  function, which is the level set function. The structure of the main program is given in the list below.

- Declare variables needed in the main program.

- Read input from the input file to initialize parameters and set boundary conditions.
- Construct the computation grid.
- Initialize the differential operations.
- Read the initial conditions created by the initial condition program.
- Open files for writing results.
- Start the time step loop in which all calculations are done. This step also includes the time discretization, writing results and some progress verification.

As an example, the module `poisson.f90` contains several subroutines and functions. In this module there is a main subroutine called `"solve_poisson.f90"` which is used to solve the Poisson equation for the pressure. The equation is solved with a tri-diagonal matrix algorithm. The coefficients for the band matrix and the right hand side of this algorithm are calculated with a function and a subroutine, while another subroutine is performing the algorithm calculations itself. The last subroutine in this module is one used to calculate the jump in pressure over the fluid-fluid interface.

### 3.2 Interface tracking

In a two phase flow, one of the most important tasks is to know where one fluid phase ends and another starts, this is the fluid-fluid interface [hereby, only called interface]. While there are other ways of doing this such as the volume of fluid method [3], this solver uses the level-set method proposed by Sethian and Osher [10]. For more details on applications of the level-set can be found in the book by Osher & Fedkiw [7]. With this method every point in the domain is given a signed distance value which describes which phase is present at the point and how far from the interface the point is. The sign determines the phase and the distance value is the orthogonal distance to the interface. In the solver this value is denoted by  $\phi$ , and it is advected with the local fluid velocity

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0. \quad (6)$$

Since the discretization of the advection equation (6) will not be exact,  $\phi$  loses its signed distance property over time. To account for this inaccuracy

the level-set function is reinitialized every few time steps. This is done by solving equation (7) until it reaches steady state.

$$\frac{\partial\phi}{\partial\tau} + \text{sign}(\phi)(|\nabla\phi| - 1) = 0 \quad (7)$$

The level-set function also allows the solver to directly calculate the normal vector  $\mathbf{n}$ , and the curvature  $\kappa$  of the interface, which are quantities used to calculate the jump conditions.

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \quad (8)$$

$$\kappa = -\nabla \cdot \mathbf{n} \quad (9)$$

### 3.3 Grid

There are several different ways of arranging points in which you calculate your results. The grid used in this solver is a uniform structured staggered grid. This means that all cells are either square or rectangular and of equal size. A staggered grid in two dimensions is in fact three separate grids: one for velocity vectors in the x-direction, one for velocity vectors in the y-direction and one for all scalar values. This is done to ease the velocity-pressure coupling. Since finite difference stencils require a certain number of nodes in each direction relative to the node considered, the grid is expanded beyond the intended domain using ghost points which are not taken into account when visualizing simulation results. A simple sample grid is shown in figure 3. These ghost points are used to enforce boundary condition. If one for example would like to use a Neumann boundary condition saying that  $\frac{\partial u}{\partial x} = 0$ , the first ghost node value will equal the first node value in the calculation domain counting from the boundary, the second ghost node value will equal the second calculation node value and so forth.

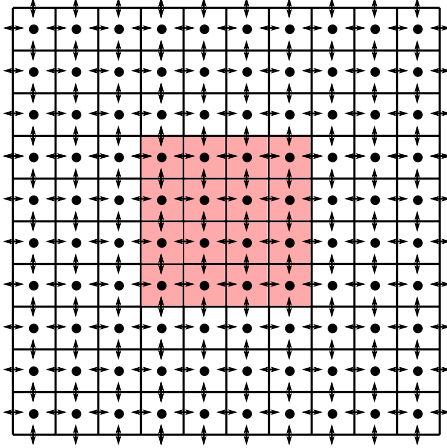


Figure 3: Illustration of the staggered grid layout. Red cells cover the calculation domain, and the white cells are the ghost cells.

### 3.4 Discretization

The viscous terms are discretized using central differences schemes in two steps. After the first derivation the velocity components are displaced by  $\frac{\Delta x}{2}$  relative to the spatial direction they have been differentiated in. As the first derivatives are used to calculate jump coefficients, they are interpolated to values in the cell center before using central differences a second time.

Time discretization is done using a 3rd order Runge-Kutta discretization. This is a weighted average of three incremental forward Euler time steps. The maximum stable time step size is calculated using the interface tension, curvature, viscosity, density, grid size and velocities. The expression for calculating the time step size will be explained in section 6. To eliminate error sources during validation of the sub tasks, the time discretization is temporarily replaced by a single forward Euler time step. The 3rd total variation diminishing (TVD) order Runge-Kutta method for  $\frac{\partial \mathbf{U}}{\partial t} = R(\mathbf{U})$  reads

$$\begin{aligned}
\mathbf{U}^{(1)} &= \mathbf{U}^n + \Delta t R(\mathbf{U}^n) \\
\mathbf{U}^{(2)} &= \frac{3}{4}\mathbf{U}^n + \frac{1}{4}\mathbf{U}^{(1)} + \frac{1}{4}\Delta t R(\mathbf{U}^{(1)}) \\
\mathbf{U}^{n+1} &= \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\mathbf{U}^{(2)} + \frac{2}{3}\Delta t R(\mathbf{U}^{(2)})
\end{aligned} \tag{10}$$

Where parenthesis note the stages.

To ensure that the solution is incompressible, the pressure is calculated using a direct projection method. An intermediate velocity field is calculated from the old velocity field using the viscous, advective and external force terms

$$\mathbf{u}^* = \mathbf{u} + \Delta t \left( -(\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{\mu}{\rho}\nabla^2\mathbf{u} + f \right) \tag{11}$$

The divergence of this intermediate velocity field is then used to compute the right hand side in the Poisson equation for the pressure such that the velocity at the new time level  $\mathbf{u}^{n+1}$  becomes divergence free.

As mentioned, the Poisson equation (12) is discretized using a tridiagonal matrix algorithm. A stencil of three points in both directions is used to organize the coefficients for this algorithm in a matrix with dimensions  $mn \times mn$ , where  $m$  is the number of cells in the x-direction, and  $n$  is the number of cells in the y-direction. There is one value for each pressure node along the diagonal, and in the diagonals to the left and right, coefficients for the stencils in both directions are stored. On the right hand side of the equation, is a vector containing the divergence of the intermediate velocities at the evaluated nodes. Solving the system  $Ax = b$ , where  $A$  is the coefficient matrix,  $x$  is the scaled pressure  $p^* = \frac{\rho}{\mu}\Delta t$  we are after, and  $b$  is the divergence of  $\mathbf{u}^*$ , is a task which requires a relatively large amount of computational time. To make this task more efficient, the matrix  $A$  is actually stored as a band matrix of dimension  $mn \times 5$ .

$$\nabla^2 p^* = \nabla \cdot \mathbf{u}^* \tag{12}$$



### 3.5 Ghost Fluid Method

The Ghost Fluid Method is a way of sharply treating discontinuities across the interface. To explain how this method works in practice we will consider a one dimensional problem in the interval  $0 \leq x \leq 1$ , with an interface at  $x = 0.5$ . The jump value at the interface is known as  $a$ , i.e.  $[u] = u^+ - u^- = a$ . The boundary value problem is defined by

$$\nabla(u(x)) = 0 \quad u(0) = 0 \quad u(1) = a \quad (13)$$

If we say that the interface is between nodes  $k$  and  $k + 1$  the differentiation over these nodes will give an artificially increased value. Defining function  $u(x) = 0$  from  $x = 0$  to  $x = 0.5$  and  $u(x) = 2$  from  $x = 0.5$  to  $x = 1$  the numerical differentiation would suggest

$$\left(\frac{\partial u}{\partial x}\right)_{k+\frac{1}{2}} = \frac{u_{k+1} - u_k}{\Delta x} = \frac{2 - 0}{\Delta x} \neq 0. \quad (14)$$

This is obviously not a good approach since the jump is instantaneous and therefore not differentiable. We separate the function calling the function to the left of the interface  $u^-$  and the function to the right of the domain  $u^+$ , according to the sign of the level-set function. We can then define  $u^+ - u^- = a$  at the interface. As seen in equation (14) the two domains are clearly mixed.

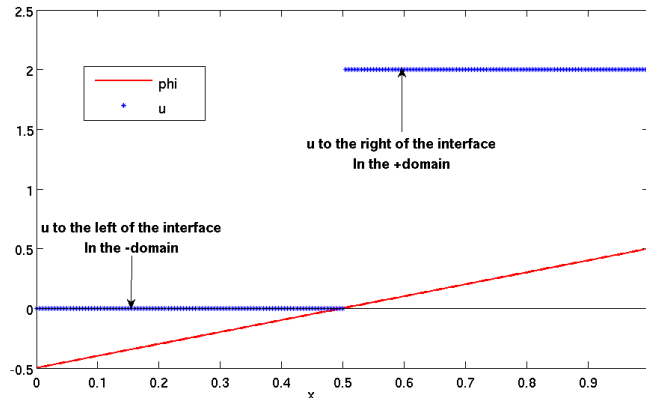


Figure 4:  $u(x)$  and  $\phi(x)$ .

We would like to differentiate over the interface using only values from one domain. If we rewrite equation (14) in the right domain, we get

$$\left(\frac{\partial u}{\partial x}\right)_{k+\frac{1}{2}} = \frac{u_{k+1}^+ - u_k^+}{\Delta x} = 0. \quad (15)$$

The result would be somewhat as we wanted, but  $u^+$  is not defined to the left of the interface. Using  $u^+ = u^- + a$  we can rewrite the latter equation as

$$\left(\frac{\partial u}{\partial x}\right)_{k+\frac{1}{2}} = \frac{u_{k+1}^+ - (u_k^- + a)}{\Delta x} = 0. \quad (16)$$

To the left of the interface, we get

$$\left(\frac{\partial u}{\partial x}\right)_{k+\frac{1}{2}} = \frac{u_{k+1}^- - u_k^-}{\Delta x} = 0 \quad (17)$$

or

$$\left(\frac{\partial u}{\partial x}\right)_{k+\frac{1}{2}} = \frac{(u_{k+1}^+ - a) - u_k^-}{\Delta x} = 0. \quad (18)$$

We now have equations for differentiating over the interface, but until now we have not introduced the exact location of the interface. Taking the subcell resolution into account is crucial to attain the results wanted for these types of problems. Using the level-set function, we can split the cell containing the interface into two pieces. The fraction of the cell to the left of the interface is determined by:

$$\theta = \frac{|\phi_k|}{|\phi_k| + |\phi_{k+1}|} \quad (19)$$

We have separated the cell, so that  $\theta\Delta x$  is the fraction to the left of the interface and  $(1 - \theta)\Delta x$  is the fraction to the right of the interface. We expand the problem given in equation (13) to the Poisson equation

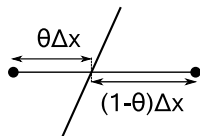


Figure 5: Subcell fractions.

$$\nabla^2 u(x) = f(x) \quad u(0) = 0 \quad u(1) = 2 \quad (20)$$

Defining the jump of  $u$  at the interface as  $u_I$  and the jump in the first derivative of  $u$  at the interface as  $\left[\frac{\partial u_I}{\partial x}\right] = b$ , we can say

$$\left(\frac{u_{k+1} - u_I}{(1 - \theta)\Delta x}\right) - \left(\frac{u_I - u_k}{\theta\Delta x}\right) = b, \quad (21)$$

which then allows us to solve for  $u_I$

$$u_I = u_{k+1}\theta + u_k(1 - \theta) - b\theta(1 - \theta)\Delta x. \quad (22)$$

On the left side of the interface we can approximate the derivative as

$$\frac{u_I - u_k}{\theta\Delta x} = \frac{u_{k+1} - u_k}{\Delta x} - b(1 - \theta), \quad (23)$$

which allows us to write the Laplacian as

$$\left[\left(\frac{(u_{k+1} - a) - u_k}{\Delta x} - b(1 - \theta)\right) - \left(\frac{u_k - u_{k-1}}{\Delta x}\right)\right] / \Delta x = f(x) \quad (24)$$

Where the relation  $u^+ = u^- + a$  was used where applicable, so that  $u$  has the value from its respective domain. Finally, we rewrite equation (24) to

$$\left[\left(\frac{u_{k+1} - u_k}{\Delta x}\right) - \left(\frac{u_k - u_{k-1}}{\Delta x}\right)\right] / \Delta x = f(x) + \frac{a}{(\Delta x)^2} + \frac{b(1 - \theta)}{\Delta x} \quad (25)$$

to the left of the interface. The right side of the interface is written:

$$\left[\left(\frac{u_{k+2} - u_{k+1}}{\Delta x}\right) - \left(\frac{u_{k+1} - u_k}{\Delta x}\right)\right] / \Delta x = f(x) - \frac{a}{(\Delta x)^2} + \frac{b\theta}{\Delta x} \quad (26)$$

This formulation allows us to calculate the derivative values of a problem across an interface with discontinuities. Another valuable aspect of this approach is that since the jump can be adjusted for at the right side of the equation, the band matrix in the Poisson equation is symmetric positive definite and the conjugate gradient method can be used.

This approach can also be expanded to work with equations which have varying coefficients in the domain. Considering a variable coefficient Poisson equation  $\nabla(\beta\nabla u) = f(x)$ , with the jumps  $[u] = a$  and  $[\frac{\partial u}{\partial x}] = b$  we can add coefficients according to its respective domain to equation (21), solve for  $u_I$  and get

$$u_I = \frac{\beta^+ u_{k+1} \theta + \beta^- u_k (1 - \theta) - b \theta (1 - \theta) \Delta x}{\beta^+ \theta + \beta^- (1 - \theta)}. \quad (27)$$

This allows us to define valid equations for the problem dealing with discontinuities in both the solution as well as its derivative at an interface between  $k$  and  $k + 1$ .

$$\left[ \hat{\beta} \left( \frac{u_{k+1} - u_k}{\Delta x} \right) - \beta^- \left( \frac{u_k - u_{k-1}}{\Delta x} \right) \right] / \Delta x = f(x) + \frac{\hat{\beta} a}{(\Delta x)^2} + \frac{\hat{\beta} b (1 - \theta)}{\beta^+ \Delta x}. \quad (28)$$

$$\left[ \beta^+ \left( \frac{u_{k+2} - u_{k+1}}{\Delta x} \right) - \hat{\beta} \left( \frac{u_{k+1} - u_k}{\Delta x} \right) \right] / \Delta x = f(x) - \frac{\hat{\beta} a}{(\Delta x)^2} + \frac{\hat{\beta} b \theta}{\beta^- \Delta x}, \quad (29)$$

where  $\hat{\beta}$  is defined by

$$\hat{\beta} = \frac{\beta^+ \beta^-}{(1 - \theta) \beta^- + \theta \beta^+}. \quad (30)$$

## 4 Extention of numerical methods for axisymmetric problems

As the task was to implement axisymmetric capabilities to the solver without losing the Cartesian capabilities, all changes to the solver were made to take effect only if the user specifies that the problem is an axisymmetric problem. To accomplish that, a global logic variable is allocated when the program starts, and if this variable is ".true." the changes come in to effect.

## 4.1 Viscous terms

In two-dimensional Cartesian coordinates the Laplace operator is evaluated as equation (31), while using cylindrical polar coordinates, it is evaluated as in (32)

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (31)$$

$$\nabla^2 = \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial}{\partial r} \right) + \frac{\partial^2}{\partial y^2} \quad (32)$$

As mentioned, the viscous terms are differentiated twice to evaluate this operation, which gives a fairly straightforward way of changing the coordinate systems. But as the velocity components do not share grids, the radial coordinate factor used in the Laplacian has to correspond with the location of the variables. The velocity is also displaced by differentiating, so the factor  $r$  and  $\frac{1}{r}$  have to be separated by  $\frac{dx}{2}$ , to accommodate this displacement. As this issue was a reoccurring one, we defined two global variables when the grid is constructed at the beginning of the program sequence. One vector for staggered grid  $r_s$ , and one for the center of the cells  $r_c$ .

## 4.2 Poisson equation

As part of the direct projection method, the Poisson equation (12) has to be evaluated. As mentioned this is done by using a tridiagonal matrix algorithm. For Cartesian coordinates the Laplace operator for the pressure is directly computed using a five node central differences scheme. This means that the implementation of an axisymmetric alternative could not be solved the same way as with the viscous terms in section 4.1, with sequentially including radial coordinate factors. By using the chain rule we can rewrite equation (32) to:

$$\nabla^2 = \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial}{\partial r} \right) + \frac{\partial^2}{\partial y^2} = \frac{1}{r} \left( \frac{\partial}{\partial r} + r \frac{\partial^2}{\partial r^2} \right) + \frac{\partial^2}{\partial y^2} = \frac{1}{r} \frac{\partial}{\partial r} + \frac{\partial^2}{\partial r^2} + \frac{\partial^2}{\partial y^2}. \quad (33)$$

By doing this, the matrix algorithm did not have to be rewritten. The modification was only adding coefficients to calculate  $\frac{1}{r} \frac{\partial}{\partial r}$  in addition to the

second derivative which was already being calculated in the two-dimensional Laplacian. In order to use the direct projection method for pressure-velocity coupling, the Poisson equation (12) has to be evaluated. Equations (34) and (35) show the difference between the two-dimensional and axisymmetric Laplacian discretization, and figure 6 shows a generic stencil used in these equations.

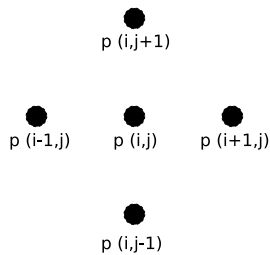


Figure 6: Stencil for the pressure Laplacian.

$$\nabla^2 p = \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\Delta x^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\Delta y^2} + O(\Delta x^2, \Delta y^2) \quad (34)$$

$$\begin{aligned} \nabla^2 p = & \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\Delta r^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\Delta y^2} \\ & + \frac{1}{r_{i,j}} \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta r} + O(\Delta r^2, \Delta y^2). \end{aligned} \quad (35)$$

### 4.3 Ghost fluid method

The ghost fluid method was written as a Cartesian method. On Cartesian coordinates the method works very well because it leads to a symmetric negative definite matrix. However, developing and implementing it for the axisymmetric case is complicated as the first derivative and the radius is not symmetric. The development itself was done by using symbolic objects in MATLAB. The final stencils for the cylindrical coordinate system were

$$\left[ \hat{\beta} \left( \frac{u_{i+1} - u_i}{\Delta x} \right) - \beta^- \left( \frac{u_i - u_{i-1}}{\Delta x} \right) \right] / \Delta x + \frac{1}{r_i} \frac{\tilde{\beta}(u_{i+1} - u_{i-1})}{2\Delta x} =$$

$$f(x)_i + \frac{\hat{\beta}a}{\Delta x^2} + \frac{\hat{\beta}b(1-\theta)}{\beta^+\Delta x} + \frac{1}{r_i} \frac{\tilde{\beta}a}{2\Delta x} + \frac{1}{r_i} \frac{\tilde{\beta}b(1-\theta)}{2\beta^+} \quad (36)$$

$$\left[ \beta^+ \left( \frac{u_{i+2} - u_{i+1}}{\Delta x} \right) - \hat{\beta} \left( \frac{u_{i+1} - u_i}{\Delta x} \right) \right] / \Delta x + \frac{1}{r_{i+1}} \frac{\bar{\beta}(u_{i+2} - u_i)}{2\Delta x} =$$

$$f(x)_{i+1} - \frac{\hat{\beta}a}{\Delta x^2} + \frac{\hat{\beta}b\theta}{\beta^-\Delta x} + \frac{1}{r_{i+1}} \frac{\bar{\beta}a}{2\Delta x} - \frac{1}{r_{i+1}} \frac{\bar{\beta}b\theta}{2\beta^-}. \quad (37)$$

Which are valid at the node before and after the interface, respectively, and where

$$\tilde{\beta} = \frac{2\beta^+\beta^-}{\beta^-(1-\theta) + \beta^+(1+\theta)} \quad \text{and} \quad \bar{\beta} = \frac{2\beta^+\beta^-}{\beta^-(2-\theta) + \beta^+\theta}. \quad (38)$$

These equations are needed to calculate the jump corrections in both the viscous forces as well as in the Poisson equation. The difference between the two is that in the Poisson equation the derivative of the solution has no jump, so  $b = 0$  and the only right hand side correction needed is the term which involves the jump in the solution, as  $a \neq 0$ . In the viscous terms, however, the case is the opposite. There are no jumps in the solution, meaning  $a = 0$ . The possible change in viscosity gives a jump in the first derivative, so  $b \neq 0$ .

In the Poisson equation we would like to keep the direct solver unchanged. In the Cartesian case this is not a problem since  $\hat{\beta}$  is the same in equations (36) and (37). When declaring the coefficients for the direct solver in the axisymmetric case, we needed to assign values according to where the interface was.

In the two-dimensional case,  $\hat{\beta}$  is stored in the same variable as  $\beta^+$  and  $\beta^-$ . They are stored in an array with one  $\beta$  (being  $\hat{\beta}$ ,  $\beta^-$  or  $\beta^+$ ) for each node. In the axisymmetric case this approach cannot be used. As seen in equations

(36) and (37),  $\tilde{\beta}$  and  $\bar{\beta}$  is used as coefficients for the same node values as  $\hat{\beta}$ ,  $\beta^+$  and  $\beta^-$ . This was solved by using variables for  $\beta$  designated to the Cartesian case when evaluating nodes away from the interface.

In the viscous terms we calculate the Laplacian as mentioned in a stepwise manner. Due to this approach, the stencil can be used directly as it stands, since  $u$  is known and the right hand side of the equation is the unknown. By using the stencil as shown, any changes done to  $u$  in the one phase solver must be taken into account. So where the jump correction is calculated we undid changes previously made to the solver, but only in the nodes directly in front or after the interface.

## 4.4 Curvature

A small, but important change is also the change from a line curvature in the two-dimensional case to a surface curvature in the axisymmetric case. A surface curvature is given as

$$\kappa = \kappa_1 + \kappa_2 = \frac{1}{r_1} + \frac{1}{r_2}. \quad (39)$$

Where  $r_1$  and  $r_2$  are the principal radii.  $\kappa_1$  is already calculated using the level set function. Finding  $\kappa_2$  is done by simply using trigonometry.

$$\kappa_2 = \frac{n_r}{r} \quad (40)$$

where  $n_r$  is the radial component of the normal vector to the interface in the calculation plane.

## 5 Verification of sub-tasks

As mentioned in the introduction, the thesis work was broken down to specific sub-tasks. This section will describe how the tasks were done, what challenges occurred and how the tasks were verified. This way of solving the task as a whole gave a lot of insight into how the program code worked.



## 5.1 Two-dimensional Cartesian heat conduction problem

The first sub-task was making the solver only to compute the viscous terms of the Navier-Stokes equation. This task was done solely by modifying the main program. Evaluating the viscous terms in any component of the momentum equation with a simple forward Euler scheme, replacing the kinematic viscosity by the thermal diffusivity, and neglecting advection is the same as evaluating the heat equation (41)

$$\frac{\partial T}{\partial t} = \alpha \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) - Q. \quad (41)$$

Where  $T$  is the temperature  $\alpha$  is the thermal diffusivity,  $t$  is the time and  $Q$  is a heat source.

## 5.2 Verification of the two-dimensional heat conduction problem

To verify the two-dimensional Cartesian heat conduction problem we implemented a manufactured solution [9]. This was done by defining a function which had suitable boundary conditions and was analytically solvable. Then by adding the analytical solution as the heat source in the solver, the steady state of the solver should be the function we suggested.

The function used in this case was

$$F(x, y) = \left( \sin(\pi x) + \frac{1}{a} \sin(b\pi x) \right) \left( \cos(\pi y) + \frac{1}{a} \cos(b\pi y) \right) \quad (42)$$

Where the heat source of the problem was given as  $\alpha$  times the Laplacian of the function

$$\begin{aligned} \nabla^2 F = & \left( -\pi^2 \sin(\pi x) - \frac{(b\pi)^2}{a} \sin(b\pi x) \right) \left( \cos(\pi y) + \frac{1}{a} \cos(b\pi y) \right) \\ & + \left( -\pi^2 \cos(\pi y) - \frac{(b\pi)^2}{a} \cos(b\pi y) \right) \left( \sin(\pi x) + \frac{1}{a} \sin(b\pi x) \right) \end{aligned} \quad (43)$$

This gives us, for integer values of  $b$ , homogeneous Dirichlet boundary conditions for  $F$  at  $x = 0$  and  $x = 1$  and homogeneous Neumann conditions at  $y = 0$  and  $y = 1$  i.e.  $F(0, y) = F(1, y) = 0$  and  $\frac{\partial F}{\partial y}(x, 0) = \frac{\partial F}{\partial y}(x, 1) = 0$ . These boundary conditions were in this intermediate stage of the work hard-coded into the program by overwriting the ghost points in each time step. All ghost points in the x-direction were given the value zero, and the ghost points in the y-direction mirrored the first points used in the computational domain. When the solver reaches steady state the equation given will be

$$\frac{\partial T}{\partial t} = k \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) - \nabla^2 F = 0 \implies \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = \nabla^2 F. \quad (44)$$

The  $T$  computed by the main program should equal the manufactured solution  $F$ . As mentioned, a forward Euler time discretization scheme was used. Even though there are a built in function to stop the simulation when it reaches steady state, the simulation was run in excess of this, due to the fast convergence of the problem. To quantify the results, the simulations have been run on varying grid resolutions. By doing so we can see that the convergence rate is what was expected using the discretization schemes that were used. Figure 7 shows convergence rates for the heat conduction problem solved on both vector component grids  $u$  and  $v$ , where the convergence rate is consistent with the theoretical order of the used discretization stencils.

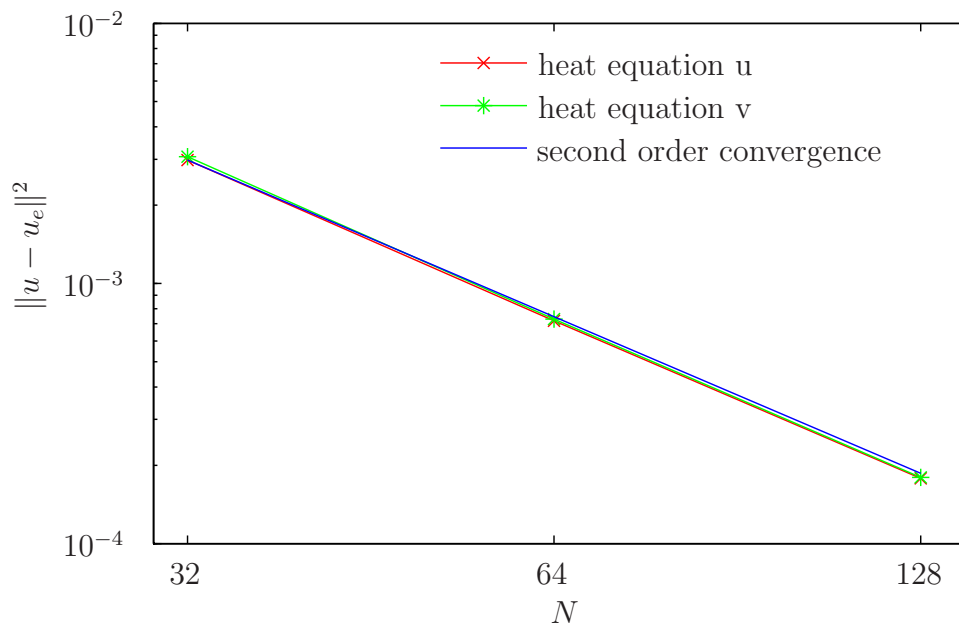


Figure 7: Grid convergence for two-dimensional heat equation.

### 5.3 Axisymmetric heat conduction problem

The next step was to adapt the solver to handle axisymmetric problems. That meant changing the coordinate system from Cartesian coordinates to cylindrical coordinates. The heat equation remains the same  $\frac{\partial T}{\partial t} = \alpha \nabla^2 T - Q$ , but the Laplace operator  $\nabla^2$  differs according to equations (31) and (32). In the axisymmetric solver, we have also added a parameter for the distance from  $r = 0$  to the beginning of the domain to be able to solve problems where the left boundary of the domain is located at  $r > 0$ .

### 5.4 Verification of the axisymmetric heat conduction problem

Verifying the axisymmetric problem was done very similar to the Cartesian heat equation, although the manufactured solution was "rotated" by replacing  $x$  with  $y$  and  $y$  with  $r$  in the original function. Another difference between the two manufactured solutions is that the simulation was done from  $r = 1$  to  $r = 2$ .

$$F = \left( \cos(\pi r) + \frac{1}{a} \cos(b\pi r) \right) \left( \sin(\pi y) + \frac{1}{a} \sin(b\pi y) \right) \quad (45)$$

The Laplacian also differs from the two-dimensional one

$$\begin{aligned} \nabla^2 F = & \left( -(\pi)^2 \cos(\pi r) - \frac{1}{a} (b\pi)^2 \cos(b\pi r) \right) \left( \sin(\pi y) + \frac{1}{a} \sin(b\pi y) \right) \\ & + \left( -\pi^2 \sin(\pi y) - \frac{(b\pi)^2}{a} \sin(b\pi y) \right) \left( \cos(\pi r) + \frac{1}{a} \cos(b\pi r) \right) \\ & + \frac{1}{r} \left( -\pi \sin(\pi r) - \frac{\pi b}{a} \sin(b\pi r) \right) \left( \sin(\pi y) + \frac{1}{a} \sin(b\pi y) \right) \quad (46) \end{aligned}$$

In the same way as with Cartesian coordinates the simulation was run on three different grid resolutions, to see that the convergence rate was as expected. It was run on the sizes 32x32, 64x64 and 128x128. The figure below shows the grid convergence, which was indeed second order as expected for the second order finite difference method. As for time discretization and convergence criterion, the same approach as is in section 5.2 was applied.

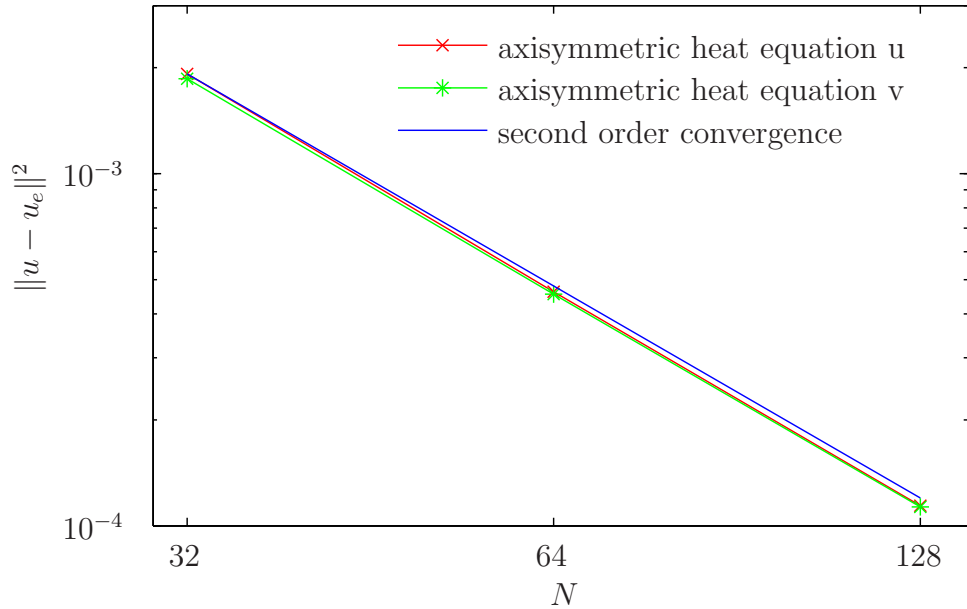


Figure 8: Grid convergence for axisymmetric heat equation at steady state for grid sizes 32x32, 64x64 and 128x128.

Figure 9 shows the deviation of the numerical from the analytical solution for 128x128 grid points. As we see the error magnitude is proportional to the terms  $\frac{1}{a} \cos(b\pi r)$  and  $\frac{1}{a} \sin(b\pi y)$ .

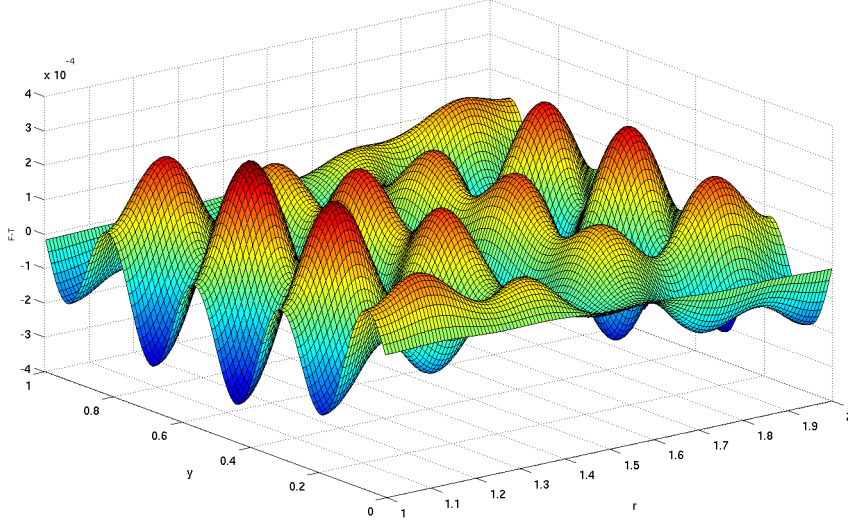


Figure 9: Deviation from analytical solution on 128x128.

## 5.5 Verification of the Poisson equation

As the Poisson equation is solved directly, the verification can be done by solving the matrix equation once. To verify the solution we generated an artificial intermediate velocity field  $\mathbf{u}$  and calculated  $\nabla^2 p^* = \nabla \cdot \mathbf{u}^*$  as described in section 3.4. The velocity field is described by equations (47) and (48)

$$u^* = -(\sin(r) + ba \sin(rb))(\sin(y) + a \sin(yb)) \quad (47)$$

$$v^* = (\cos(y) + ba \cos(yb))(\cos(r)) + a \cos(rb) \quad (48)$$

Which will yield the exact scaled pressure of

$$p^* = (\cos(r) + a \cos(rb))(\sin(y) + a \sin(yb)) \quad (49)$$

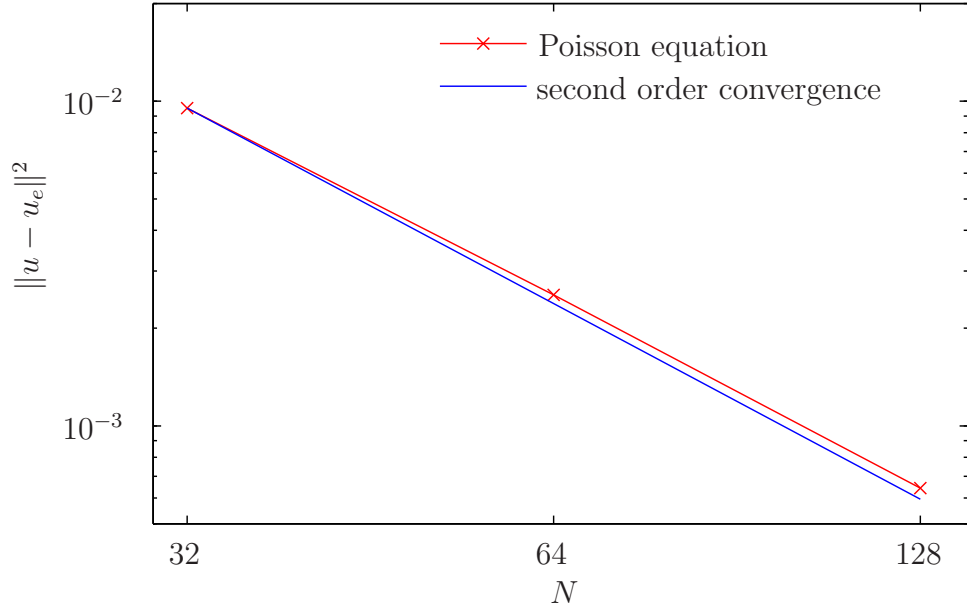


Figure 10: Grid convergence for axisymmetric Poisson equation for grid sizes 32x32, 64x64 and 128x128.

## 5.6 Verification of an axisymmetric single phase Navier-Stokes solver

Adding together the changes made in the previous sub tasks, we have a working axisymmetric single phase solver. To verify that the changes made to the solver work as a whole, we solved an axisymmetric simple Couette flow. This problem is analytically solvable and thereby easy to verify. The domain is restricted by walls at  $r = 1$  and  $r = 6$  and uses periodic boundary conditions at  $y = 0$  and  $y = 2\pi$ . The outer wall is moving at a constant velocity of 1, and the pressure is constant over the whole domain. Steady state conditions of this problem are described by equation (51)

$$\frac{\partial}{\partial r} \left( r \frac{\partial \mathbf{u}}{\partial r} \right) = 0, \quad \mathbf{u}(1, y) = 0, \quad \mathbf{u}(6, y) = 1\mathbf{j} \quad (50)$$

where  $\mathbf{j} = (0, 1)^T$ . This gives the analytical solution of the velocity distribution

$$\mathbf{u} = \frac{1}{\log(6)} \log(r) \mathbf{j} \quad (51)$$

Figure 11 shows the flow distribution of the simple Couette flow and the deviation from the analytical solution.

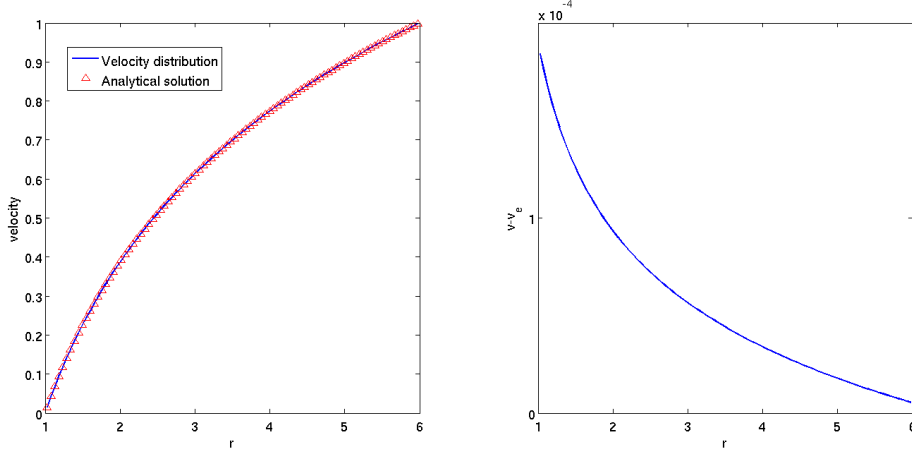


Figure 11: Flow distribution from simulation and deviation from the analytical solution.

## 5.7 Verification of the Ghost Fluid Method

Verification that the Ghost Fluid Method in the viscous terms was done by solving  $\nabla^2 \mathbf{u}$  where  $\mathbf{u} = \mathbf{u}(r)$ , and the solution in cylindrical coordinates is defined as follows

$$\begin{aligned} \mathbf{u} &= \log(r + 1) & \nabla^2 \mathbf{u} &= \frac{1}{r(r + 1)^2} & r &\in [0, 0.5) \\ \mathbf{u} &= \log(0.5 + 1) & \nabla^2 \mathbf{u} &= 0 & r &\in (0.5, 1] \end{aligned} \quad (52)$$

The interface is located at  $r = 0.5$  and the jump in the first derivative is  $b = \frac{-\beta^-}{0.5+1}$ .  $\beta^- = 100$  and  $\beta^+ = 1$ .

A part of the discretization error in solving this equation comes from  $r$  which differentiates the cylindrical coordinates from the Cartesian coordinates. When using the values from the different grid sizes, the error norm

convergences at a lower order than the solution. This is due to the fact that as  $\Delta r$  gets smaller, the first node comes closer to the boundary (consider the situation if the left boundary is at  $r = 0$ , where  $\frac{1}{r}$  diverges as  $r$  closes in on the boundary). As we can see in figure 12 the error converges with an order of 1.5

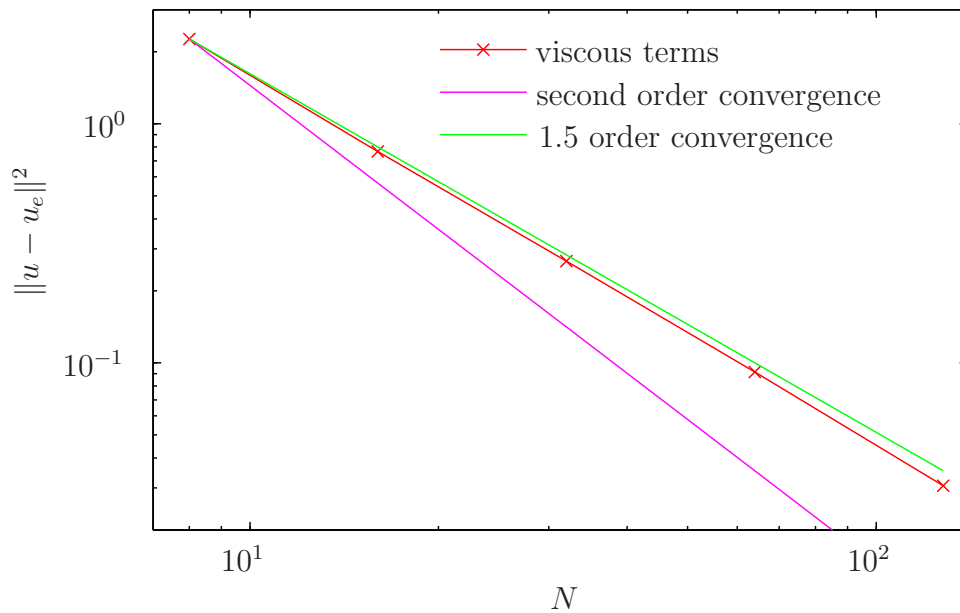


Figure 12: Convergence rate of the GFM in the viscous case.

## 6 Results

Combining the single phase Navier-Stokes solver with the modified two phase models for curvature jump conditions, we had an axisymmetric two phase Navier stokes solver. For simplicity, all variables are done non-dimensional throughout this section.

### 6.1 Spherical bubble at rest

The first test case of the two phase axisymmetric Navier-Stokes solver was the simple case of a spherical bubble at rest. The test was defined by



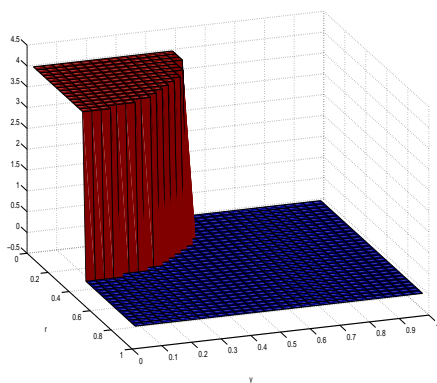
- The domain was constructed from  $r = 0$  to  $r = 1$  and  $y = 0$  to  $y = 1$  with symmetric boundary conditions at all four boundaries of the domain.
- The interface was defined in the initial conditions with  $\phi = \sqrt{r^2 + y^2} - 0.5$ , simulating only one quarter of the symmetrical bubble.
- Inside the bubble the non-dimensional density was 2 and the non-dimensional viscosity  $1/10$ , and outside the bubble the non-dimensional density was 1 and the non-dimensional viscosity 1. The non-dimensional surface tension between the two phases was 1.

The time discretization in the axis symmetric multiphase solver is a third order TVD Runge-Kutta scheme. The maximum time step is calculated in the same way as in the two-dimensional case, using an expression provided by Kang et al. [4].

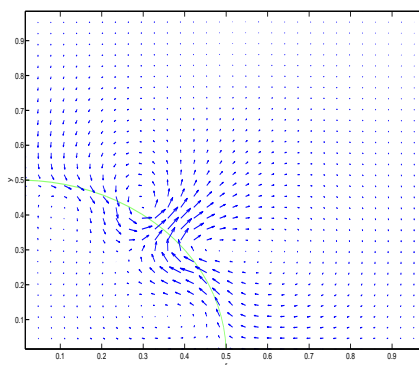
$$\frac{\Delta t}{2} \left( (C_{\text{CFL}} + V_{\text{CFL}}) + \sqrt{(C_{\text{CFL}} + V_{\text{CFL}})^2 + 4S_{\text{CFL}}^2} \right) \leq 1 \quad (53)$$

where  $C_{\text{CFL}} = \frac{|u|_{\text{max}}}{\Delta r} + \frac{|v|_{\text{max}}}{\Delta r}$ ,  $V_{\text{CFL}} = \max\{\frac{\mu^-}{\rho^-}, \frac{\mu^+}{\rho^+}\} \frac{4}{\Delta r^2}$  and  $S_{\text{CFL}} = \sqrt{\frac{\sigma|\kappa|}{\min\{\rho^-, \rho^+\} \Delta r^2}}$

To illustrate the stable bubble, figure 13 shows the sharp pressure jump across the interface and the velocity distribution. Note that the highest velocity in the figure is  $6.318e - 5$ .



(a) Pressure



(b) Velocity vectors. Highest velocity corresponds to  $6.318e - 5$

Figure 13: Stable spherical bubble.

The results indicate that the bubble does not change in any way. This simple simulation shows that the height of the pressure jump is equal to the interface tension multiplied by the local three dimensional curvature, which in this case is  $\kappa = 1/0.5 + 1/0.5 = 4$ . To show that the bubble remains stable, figure 14 shows the maximum absolute velocity in the domain for each time step.

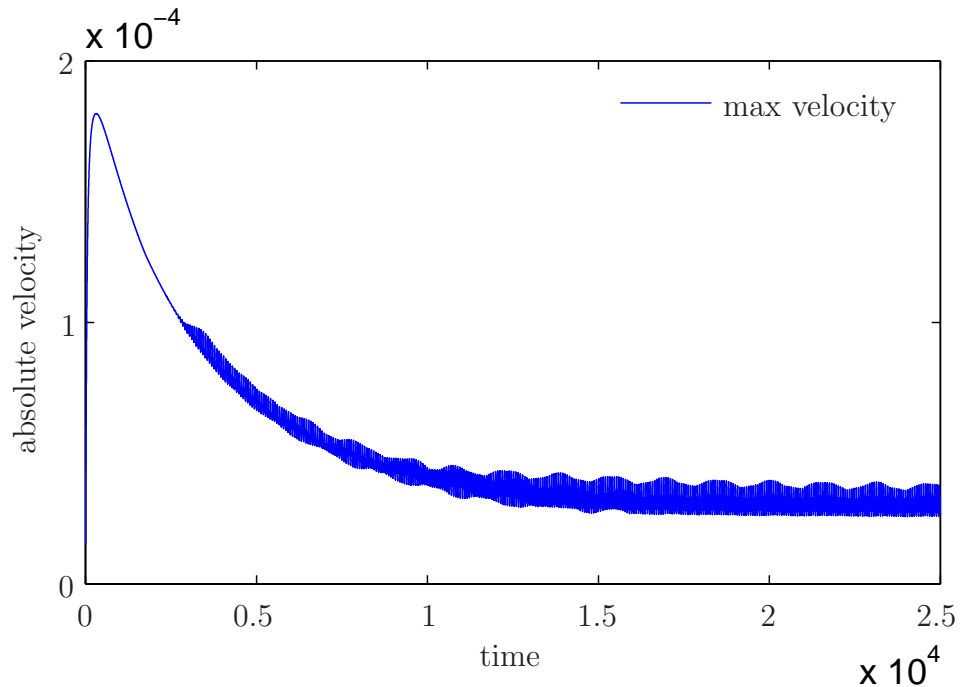


Figure 14: Maximum velocity in the domain.

## 6.2 Oscillating elliptic bubble

A popular test in two-phase simulation is the oscillating bubble. In this test we define a bubble with an elliptic interface, which due to differences in the radius will have a non-constant jump across the interface as per equation (4). This section will present two cases. The inviscid case with no body forces, will be a perpetual undamped oscillation. For the viscous case the oscillations will be damped and the interface will return to a circular shape

For both cases, the the ellipse was defined by an ellipse function with the same area as a circle with radius  $r_0 = 1/3$ , semi-major axis  $a = 1.1r_0$ , and semi-minor axis  $b = r_0/1.1$ . Also in both cases, the densities were  $\rho^- = 2$  and  $\rho^+ = 1.5$ .

For the viscous damped case the viscosities were  $\mu^- = 1$  and  $\mu^+ = 0.1$ . The CFL-number was 0.5. The simulation was over dimensioned time wise, since the test was only done to ensure that viscosity would dampen the oscillation. Figure 15 shows that an elliptic bubble with no body forces and no initial velocity will find an equilibrium state where the bubble is spherical.

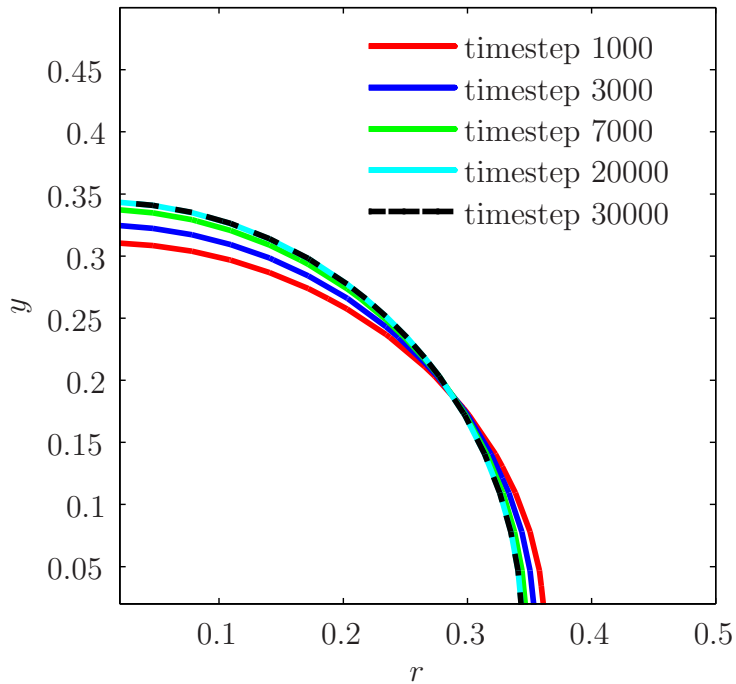


Figure 15: Elliptic bubble with viscous dampening.

### The inviscid oscillating bubble

The inviscid bubble will, as stated above, oscillate without dampening. In this test of the solver we are considering an inviscid bubble surrounded by an inviscid fluid. The size of the initial bubble was unchanged, but the CFL number was reduced to 0.05 both to ensure a stable solution and to get accurate values for the oscillation time. The oscillation should follow equation (54) as stated by Lamb [5]

$$T_0 = 2\pi \sqrt{\frac{(3\rho^- + 2\rho^+)r_0^3}{24\sigma}} \quad (54)$$

Which for our case would give an oscillation time of  $T_0 = 0.74$  time units. Table 1 shows the oscillation time for the inviscid oscillating bubble in two different domain sizes.

$m \times n$	$x, y \in \langle 0, 1 \rangle$	$x, y \in \langle 0, 1.5 \rangle$
32x32	1.0317	1.278
64x64	1.07	1.0584
128x128	1.0766	1.0389

Table 1: Oscillation time for the inviscid oscillating bubble.

The oscillation time is not following equation (54) in this table. However, the analytical solution implies that the bubble is surrounded by an infinite fluid body, and in our case the domain is constrained at  $x = y = 1$  and  $x = y = 1.5$  respectively. We can see that by increasing the size of the domain, the solution is converging in the right direction, although not towards the analytical solution by Lamb. Increasing the domain size further could help the solution converge to the analytical solution.

Expanding the calculation domain to  $x, y \in \langle 0, 2.5 \rangle$  and the densities to  $\rho^- = 1$   $\rho^+ = 0.5$  gave the result presented in table 2. The analytical solution for these densities corresponds to a non-dimensional oscillation time of 0.4937.

$m \times n$	Oscillation time
50x50	0.4723
64x64	0.4930
96x96	0.4915

Table 2: Oscillation time for the inviscid oscillating bubble.

The method for calculating the oscillation time is unfortunately not very accurate, as it is a trade-off between saving as many time steps as possible when still wanting to be able to run multiple tests. This can be the reason that the results from table 2 do not converge exactly to the analytical solution.

Figure 16 shows the velocity field of an inviscid oscillating bubble, note the difference from the bubble at rest.

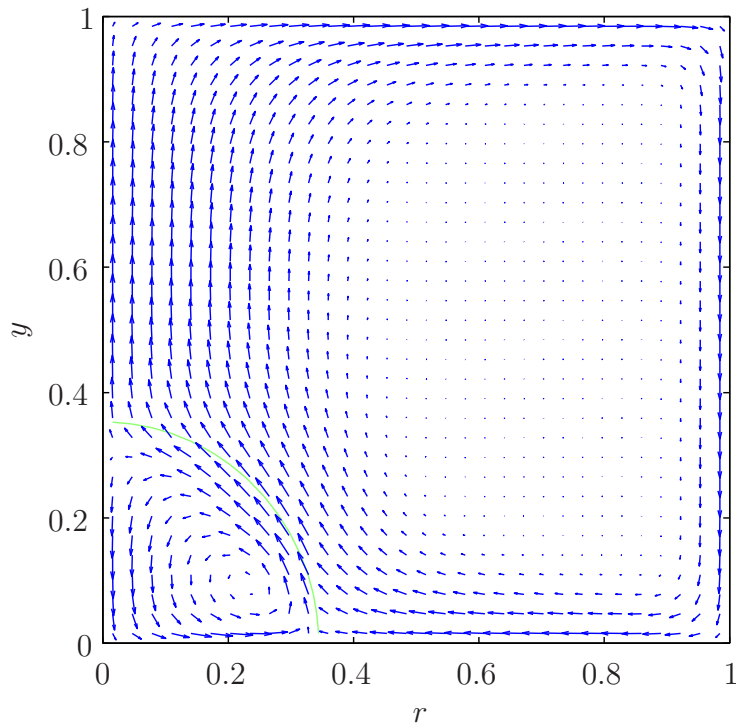


Figure 16: Inviscid oscillating bubble.

## 7 Conclusions and outlook

The two-dimensional two phase Navier-Stokes solver was adapted to handle axisymmetric problems in a thorough in a stepwise verified manner. Results indicate that each intermediate modification to the solver was done satisfyingly. Results from the axisymmetric two phase solver illustrates that the jumps are correctly handled in a sharp manner as could be expected.

As this project was to work on an existing solver in a relatively unfamiliar programming language and programming structure, a lot of the problems I encountered were related to difficulties of identifying how the program handled every variable that was modified. As a personal effort I think this work has given me a good basic understanding of how computational fluid dynamics programs work, and what challenges may occur.

To work further on this project would involve mostly testing the solvers capabilities further. After the work done to the solver and validating sub-tasks,

there was little time for running more test cases. Cases such as a two phase jets would be interesting, but relatively time consuming. If this adaptation is subject to be used in further work with the solver, some inefficient code should be rewritten.

As the solver's initial intention is to model the moving contact line, it would be interesting to add a model for the axisymmetric contact line movement. The capability for treating such a problem is already present in the solver, and there is a lot of interesting literature available on the subject of three dimensional contact line [8].

## Acknowledgements

I would like to take this opportunity to thank my advisor Bernhard Müller for making sure that there was a progress schedule I could follow, and for a critical review of this report. I would also like to thank my co-advisor Claudio Walker for defining such an interesting project, and for help in everything from basic Linux skills to the inner workings of his solver. Without his teachings I would still be trying to validate the first few steps of the adaption.

## References

- [1] J. Brill. Multiphase flow in wells. *Journal of Petroleum Technology*, 39(1):15–21, 1987.
- [2] J. Chessa and T. Belytschko. An enriched finite element method and level sets for axisymmetric two-phase flow with surface tension. *International Journal for Numerical Methods in Engineering*, 58(13):2041–2064, 2003.
- [3] C.W. Hirt and B.D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.
- [4] M. Kang, R.P. Fedkiw, and X.D. Liu. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, 15(3):323–360, 2000.
- [5] H. Lamb. *Hydrodynamics* Cambridge, 1932.
- [6] X.D. Liu, R.P. Fedkiw, and M. Kang. A boundary condition capturing method for Poisson’s equation on irregular domains. *Journal of Computational Physics*, 160(1):151–178, 2000.
- [7] S. Osher and R.P. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Verlag, 2003.
- [8] M. Pasandideh-Fard, S. Chandra, and J. Mostaghimi. A three-dimensional model of droplet impact and solidification. *International Journal of Heat and Mass Transfer*, 45(11):2229–2242, 2002.
- [9] P.J. Roache. Verification and validation in computational science and engineering. *Computing in Science Engineering*, pages 8–9, 1998.
- [10] J.A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Number 3. Cambridge Univ Pr, 1999.
- [11] M. Sussman and E.G. Puckett. A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162(2):301–337, 2000.
- [12] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct numerical simulations of gas-liquid multiphase flows*. Cambridge Univ Pr, 2011.

- [13] C. Walker and B. Müller. Contact line treatment with the sharp interface method. In *MekIT'11 Sixth National Conference on Computational Mechanics*, pages 451–462. Tapir Academic Press, 2011.