




Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: M.Sc Petroleum Engineering / Drilling	Spring, 2019 Open
Author: Keino Roxman	 (signature of author)
Supervisor: Kjell Kåre Fjelde	
Title of master's thesis: Boundary Condition Treatment in a Transient Flow Model	
Credits: 30	
Keywords: MATLAB AUSMV Scheme Zero-order extrapolation technique First-order extrapolation technique Compatibility relations method Drift flux model	Number of pages: 66 + supplemental material/other: 48 Stavanger, 14.06.2019

Acknowledgement

I would like to thank my fiancée and my family for their support throughout my studies.

I would also like to give a special thanks to my supervisor, Kjell Kåre Fjelde, for providing excellent feedback and being very positive and helpful.

Abstract

Transient flow modelling used for hydraulic calculations and well control evaluations is a vital part to the safety in the industry to properly prepare and build correct procedures when constructing wells. Modeling liquid and gas flow is complex and it's important to keep introducing new, efficient ideas to reduce model uncertainties and run time. This can help reduce numerical liabilities, risk and cost. The objective in this thesis has been to implement a new way of treating the boundary conditions in a transient flow model and compare it to more widely used methods. In addition, demanding benchmark cases have been developed to properly test the different boundary condition treatment methods against one another.

The AUSMV scheme is used for modeling the flow and pressures in the work in this thesis. The AUSMV scheme uses a simplified transient drift flux model for two-phase flow to predict flow and pressures in the system. Water and air are used instead of oil and gas to reduce the model complexity and instead put focus on the boundary treatment. The model makes a variety of assumptions and include conservation of mass and momentum and is supplied with closure laws.

Previous work with AUSMV scheme has mostly used a zero-order extrapolation technique to treat the boundary condition. The first-order extrapolation technique has been introduced recently to improve upon the zero-order. With the work in this thesis, it would be possible to avoid the use of extrapolating the mid-cell value to the boundary completely. The compatibility relations method uses characteristics that transport information from the interior computational domain toward the boundary. On the outgoing boundary, the characteristic compatibility relations together with the imposed physical conditions are used to determine the unknown variables at the boundary points.

For a horizontal pipeline with an open end, when considering pressure pulse propagation both the use of compatibility relations method and first-order extrapolation technique worked well. However, the compatibility relations method seems to work slightly better avoiding some unphysical oscillation that was seen initially in the simulation.

For a closed well with a migrating kick, the compatibility relations method had no clear benefit over first-order extrapolation technique. The compatibility relations method even had some stability issue for a finer grid.

For the open hole cases (case 3 and case 4), a kick is migrating on its own or being circulated out. This leads to an unloading situation where the liquid in front of the kick is forced out of the well violently before the gas breaches through. For the rough grid of 25 cells, there were large differences in results when comparing zero-order and first order extrapolation techniques and compatibility relations method. However, for a fine grid of 100 cells, all models seem to give the same results regarding pressure development. However, there was a difference in maximum liquid and gas flowrates predicted between the boundary condition treatment methods. Compatibility relations method tended to predict a lower maximum liquid flowrate and higher maximum gas flowrate.

The input of using a sufficiently refined grid was demonstrated. This reduces numerical diffusion and the difference between the different boundary condition treatment methods are reduced. One can also note that for rough grids, the zero-order extrapolation technique tended to be more unstable.

In general, increasing the number of cells in the well will have a greater impact than which boundary condition treatment is chosen. The reason for this is reduced numerical diffusion which is an important factor in simulation results accuracy. In addition, with an increased number of cells, the numerical errors that were introduced with a zero-order extrapolation technique (when considering large gas expansion effect) is reduced. There was a tendency that first-order extrapolation technique in combination with a rough grid produced more numerical problems.

There was an issue with extra gas being added in the system for all simulations (except case 1). It was always a similar amount gas being added and it happened early in the simulation. However, this issue is unlikely to have anything to do with the newly implemented boundary condition treatment as it was present whether the compatibility relations method was used or

not. There was also a compatibility relations method stability issues for case 2 when using a refined grid. These issues were not solved and needs further investigation.

Table of Contents

Acknowledgement	i
Abstract	iii
Table of Contents	vi
List of Figures:	viii
List of Symbols:	x
1. Introduction	1
1.1 Objective.....	2
2. Transient Drift Flux Model	4
2.1 Conservation Laws.....	4
2.2 Closure Laws	5
2.3 Eigenvalues	7
2.4 Characteristics	9
2.5 Discretization	12
2.6 Primitive Variables.....	14
2.7 Well Status.....	15
2.8 Water Hammer Effect.....	16
3. AUSMV Scheme	17
3.1 Numerical Flux Formulas for AUSMV	17
3.2 Boundary Treatment	19
3.3 Extrapolation	20
3.3.1 Practical Implementation of Extrapolation Techniques	21
3.4 Compatibility relations	23
3.5 Slope Limiters	28
4. Simulations & Discussion	30
4.1 Case 1 – Horizontal Pipe Flow	32
4.2 Case 2 – Vertical Well Kick, Closed Well.....	38
4.3 Case 3 – Vertical Well Kick, Open Well.....	43
4.4 Case 4 – Vertical Well Kick, Open Well, Circulate Kick Out	51
5. Conclusion and Future Work	61

Future Work.....	63
6. References	65
7. Appendices	67
Appendix 1.....	67
Appendix 2.....	89

List of Figures:

Figure 2.1: One-phase inflow boundary [3].	10
Figure 2.2: One-phase outflow boundary [3].	10
Figure 2.3: Two-phase inflow boundary [3].	11
Figure 2.4: Two-phase outflow boundary [3].	12
Figure 2.5: Discretization in an explicit AUSMV scheme [4].	13
Figure 3.1: Difference between boundary cell value and outlet boundary flux value.	19
Figure 3.2: Illustration of different forms of variable extrapolation. Space extrapolation of variable X at fixed time [7].	20
Figure 3.3: One-phase inlet boundary [3].	24
Figure 3.4: One-phase outlet boundary [3].	25
Figure 3.5: Two-phase inlet boundary [3].	26
Figure 3.6: Two-phase outlet boundary [3].	27
Figure 3.7: Slope limiter concept [6].	29
Figure 4.1: Graph showcasing the vacuum issue at inlet due to low atmospheric pressure.	32
Figure 4.2: Inlet pressure vs. time with atmospheric pressure set to 10 bar.	33
Figure 4.3: Inlet pressure vs. time of first-order extrapolation technique.	34
Figure 4.4: Inlet pressure vs. time of compatibility relations method.	35
Figure 4.5: Inlet pressure vs. time for both boundary condition treatment methods.	36
Figure 4.6: Pressure vs. distance for first-order extrapolation technique.	36
Figure 4.7: Inlet pressure vs. time comparing first-order extrapolation technique and compatibility relations method.	37
Figure 4.8: Bottomhole pressure vs. time of both first-order extrapolation technique and compatibility relation method and refinement.	39
Figure 4.9: Bottomhole pressure vs. time for compatibility relations method boundary condition treatment.	40
Figure 4.10: Bottomhole pressure vs. time comparing first-order extrapolation technique and compatibility relations method.	41
Figure 4.11: Gas mass vs. time for first-order extrapolation technique.	41
Figure 4.12: Gas volume vs. time grid refinement for first-order extrapolation technique and compatibility relation method.	42
Figure 4.13: Bottomhole pressure vs time for zero-order and first-order extrapolation techniques with grid refinement.	44
Figure 4.14: Kick volume vs. time for zero-order and first-order extrapolation techniques with grid refinement.	45
Figure 4.15: Bottomhole pressure vs. time for first-order extrapolation technique and compatibility relations method with grid refinement.	46
Figure 4.16: Kick volume vs. time for first-order extrapolation technique and compatibility relations method with grid refinement.	46
Figure 4.17: Depth vs. gas fraction for 25 cells (top row) and 100 cells (bottom row).	48

Figure 4.18: Liquid mass rate out vs. time for all methods over grid refinement.	49
Figure 4.19: Depth vs. liquid velocity for 25 cells (top row) and 100 cells (bottom row) with grid refinement.	50
Figure 4.20: Bottomhole pressure vs. time for zero-order and first-order extrapolation technique with grid refinement.	52
Figure 4.21: Gas volume vs. time for zero-order and first-order extrapolation techniques.	53
Figure 4.22: Bottomhole pressure vs time for first-order extrapolation technique and compatibility relations method.	54
Figure 4.23: Gas volume vs time for first-order extrapolation technique and compatibility relations method.	55
Figure 4.24: Depth vs. gas fraction for 25 cells (top row) and 100 cells (bottom row).	56
Figure 4.25: Liquid mass rate out vs. time for all boundary condition treatment methods for different grid refinement.	57
Figure 4.26: Gas mass rate out vs. time for all boundary condition treatment methods over grid refinement.	59
Figure 4.27: Depth vs. liquid velocity for 25 cells (top row) and 100 cells (bottom row) with grid refinement.	60
Figure 5.1: Gas mass vs. time showing gas being added to the system unintentional.	64

List of Symbols:

- α – Volume fraction
- ρ – Density, kg/m³
- \mathbf{v} – Velocity, m/s
- Γ – Mass exchange, m³
- t – Time, s
- g – Acceleration due to gravity, m/s²
- q – Flow rate, m³/s
- p – Pressure, Pa
- \mathbf{F} – Numerical Fluxes
- θ – Angle of inclination, ° relative to vertical
- f – Frictional coefficient
- d – Diameter, m
- Re – Reynold's number
- μ – Viscosity, Pa*s
- a – Speed of sound in liquid/gas, m/s
- K – Gas distribution coefficient
- S – Slip velocity, m/s
- λ – Gas fraction
- ω – Approximation for sound velocity, m/s
- M – Maximum number of cell segments
- Subscripts**
- l – Liquid
- g – Gas
- mix – Mixture
- I – Inner
- o – Outer
- t – Time
- z – Spatial variable
- I – Conservative variables
- j – Cell number
- Superscript**
- n – Time level

1. Introduction

Running kick simulations is a vital part of the planning phase of drilling oil and gas wells. There are countless risks associated with the drilling and completion operations and being able to predict what will occur if a kick were to take place with some degree of accuracy helps minimize the risk and build the correct procedures. To simulate a kick, we use a transient flow model which can show how different parameters vary in time. Two-phase models are popular within the oil and gas industry to describe the flow and pressures of liquid and gas. In this thesis, a two-phase model is used to describe the flow of liquid through a long horizontal pipe and simulating a kick through a vertical well. Due to the complex nature of flow of liquid and gas, a simpler drift flux model is commonly used. The drift flux model is here used to describe the transient flow that will predict well pressures and flowrates.

There are different kick models in the industry, and they all contain uncertainties that need to be accounted for. That's why it's important to keep introducing new, efficient ideas to reduce the numerical liabilities in the models to avoid the wrong decisions being made because of them. One interesting kick model has been presented in [9] where they look at a Managed Pressure Drilling (MPD) system with the focus of reducing the numerical diffusion in the system by implementing slope limiters or front tracking in combination with grid refinement. This is particularly important in an MPD system as there are small pressure margins for bottomhole pressure [9].

As for transient models, there are only a few of them in the industry as they are very complex and require a vast number of well specific information to give any sort of accurate result. In addition, they require years of testing and calibration but can provide critical information to understand how different scenarios may develop throughout the operations. In [6] and [15], a glimpse is given of how many factors and mathematical models that are required in a rigorous simulation model. These models can have a wide scope of use which include hydraulic calculations, kick simulations, underbalanced drilling and many more. They are used during the

planning phase, training and are also important when monitoring the operations to detect any abnormalities that may occur.

In this thesis, the upwind scheme, the advection-upstream-splitting-model hybrid scheme (AUSMV), is used to solve the drift flux model. The AUSMV scheme is well described in [5]. The drift flux model is a non-linear hyperbolic system of partial differential equations which describes propagation of sonic waves and mass transport waves. As AUSMV is a first-order scheme, numerical diffusion will smear out sharp discontinuities. There are some methods to counteract this which will be looked at later in the thesis.

For the different type of cases that will be studied in this thesis, the numerical fluxes of the boundary cells will be different and will require different methods to find their values. At a boundary, certain values will be given by the physics of the system (physical conditions). An example of this is atmospheric pressure at the outlet of a well. The other variables have to be found by different numerical approaches using interior information from the computational domain. The total number of physical and numerical boundary condition we can set at a boundary equals the number of conservation laws that is defined by the drift flux model (3) and the three waves this system describes (pressure pulses propagating upstream and downstream and mass transport of gas.)

In this thesis, we will explore extrapolation techniques and compatibility relations. Extrapolation method is simpler method, which according to [7], "*is generally sufficient for second-order schemes*" (Hirsch, 353). Compatibility relations is a more advanced method and involves more complex mathematics to calculate the fluxes at the boundaries. Compatibility relations equations will be implemented and compared to simulations that have been run using extrapolation method.

1.1 Objective

The objective of this thesis is to implement compatibility relations into the AUSMV scheme for the boundary treatment and compare the results with the AUSMV scheme using extrapolation technique. Then one will draw conclusion whether it is better to invest the time to implement

the compatibility relation or stick with the simpler method of using extrapolation methods to find the boundary flux values.

To get a good understanding of the accuracy of the results, a set of different robust benchmark cases have been developed to compare the results using different approaches for handling the boundary conditions.

- A horizontal pipe with generation of a pressure pulse propagating in the system.
- Closed well with a migrating kick.
- Open well with a migrating kick.
- Circulate kick out with open well.

A part of the work in this thesis has been to build robust benchmark test cases for the models to be compared against. As the hope with this study is to find what sort of strengths and weaknesses each method will have. One method may be better at predicting certain aspects of a model compared to the other, e.g. one method could be more reliable at predicting pressures pulses early in the simulation or the other could be more reliable at predicting the outlet gas and liquid flowrates. By comparing the results from the different methods, we can identify what could be the optimal numerical boundary treatment.

2. Transient Drift Flux Model

Two-phase modeling is widely used within the oil and gas industry as it can be used to predict the flow and pressure of liquids and gas. However, from a modeling and numerical perspective, two-phase flow is known for being quite complicated. As described in [2], *“two-phase flow can be described by a one-dimensional two-fluid model involving six first order non-linear partial differential equations describing mass, momentum and energy balances for each of the phases.”* In addition, it has to be supplemented with different closure laws including PVT models, friction, etc. Here, we will use a simpler two-phase flow model named the drift flux model which is obtained by adding the momentum equations. A gas slip relation has to be supplied to model that gas moves faster than liquid [5]. Although the fluids in the industry consists of oil and gas, this thesis uses water and air as this will reduce some of the complexity from the model. Within reasonable range of the parameters, the drift-flux model has been shown to be hyperbolic [16].

2.1 Conservation Laws

A description of the model is given in [8].

With the assumption that the flow is isothermal, the drift-flux model takes the form:

Mass conservation equation for liquid phase:

$$\partial_t[\alpha_l \rho_l] + \partial_z[\alpha_l \rho_l v_l] = \Gamma_l \quad 2.1$$

Mass conservation equation for gas phase:

$$\partial_t[\alpha_g \rho_g] + \partial_z[\alpha_g \rho_g v_g] = \Gamma_g \quad 2.2$$

Mixture momentum equation:

$$\partial_t[\alpha_l \rho_l v_l + \alpha_g \rho_g v_g] + \partial_z[\alpha_l \rho_l v_l^2 + \alpha_g \rho_g v_g^2 + p] + \partial_x P = -q \quad 2.3$$

In these equations t represents time and z is the spatial variable along the flow. The subscripts l and g represent liquid and gas, respectively. α , ρ , v are volume fraction, density, and velocity, respectively, for the different phases and P is the pressure.

2.2 Closure Laws

To be able to solve for the unknowns in the drift-flux model, a set of closure laws are used. The necessary amount of closure laws must be equal to the number of unknowns in the drift-flux model. Additional assumption must also be established [8]:

There is no mass transfer between the phases:

$$\Gamma_l = \Gamma_g = 0 \quad 2.4$$

Where Γ_l and Γ_g represent mass exchange between the two phases.

The mixture properties of the two phases:

$$\rho_{mix} = \alpha_l \rho_l + \alpha_g \rho_g \quad 2.5$$

$$\mu_{mix} = \alpha_l \mu_l + \alpha_g \mu_g \quad 2.6$$

$$v_{mix} = \alpha_l v_l + \alpha_g v_g \quad 2.7$$

The sum of the phase volume fraction will always be equal to one:

$$\alpha_l + \alpha_g = 1 \quad 2.8$$

Source term is expressed as:

$$q = F_w + F_g \quad 2.9$$

The effect of gravitational forces is expressed as:

$$F_g = g \rho_{mix} \cos \theta \quad 2.10$$

The effect of frictional forces has in our model been expressed as:

$$F_w = \frac{2fv_{mix}\rho_{mix}abs(v_{mix})}{(d_o - d_i)} \quad 2.11$$

The θ represents the inclination of the system and g is the acceleration due to gravity. d_o and d_i are the outer and inner diameter of the annular flow area. To find the friction factor, the Reynolds number must be calculated.

$$Re = \frac{\rho_{mix}(d_o - d_i)abs(v_{mix})}{\mu_{mix}} \quad 2.12$$

The Reynolds number tells us whether the flow is turbulent or laminar. If $Re < 2000$, the flow will be laminar and if $Re > 3000$, the flow is turbulent.

Friction factor, laminar:

$$f = \frac{24}{Re} \quad (\text{for annulus}) \quad 2.13$$

Friction factor, turbulent:

$$f = 0.052Re^{-0.19} \quad 2.14$$

Between Reynold's number 2000 and 3000, there is a transition zone. In this zone, it is hard to determine the friction factor precisely, and therefore there is a linear interpolation used between laminar friction factor at 2000 and turbulent friction factor at 3000. This ensures a smooth transition between the two different flow regimes.

There are some simple models to predict the density of liquid and gas in the system. The liquid density model:

$$\rho_l = \rho_{l,0} + \frac{(P - P_0)}{a_l^2} \quad 2.15$$

Gas density model:

$$\rho_g = \frac{P}{a_g^2} \quad 2.16$$

Where a_l represents the speed of sound in liquid: $a_l = 1500 \text{ m/s}$. a_g represents the speed of sound in gas: $a_g = 316 \text{ m/s}$. P_0 is the pressure at standard condition (1 bar).

The gas slip model:

$$v_g = K(\alpha_g v_g + \alpha_l v_l) + S \quad \text{or} \quad v_g = K v_{mix} + S \quad 2.17$$

v_{mix} is mixture velocity and is the sum of the superficial velocities of gas and liquid. K and S are flow dependent coefficients. K is a coefficient and S is the slip velocity. E.g. K = 1.0 and S = 0 the gas and liquid will move with the same velocity. In our model, K = 1.2 and S = 0.55.

It should be noted that the closure laws used here are simple and that for field conditions, more advanced models must be adapted. For instance, the gas slip relation will change when there are different flow patterns in the well. The closure models used here have been used in previous paper presenting the model and one example is given in [8].

2.3 Eigenvalues

As mentioned, the drift-flux model is a hyperbolic system and describes propagation of pressure pulses and mass waves. A kick is an example of a mass wave that would be migrating upwards. When changes happen to the system, opening of valves or starting a pump for example, sonic waves are generated throughout the system [1]. For a nonlinear system of partial differential equations, it is possible to analyze the system and find the so-called eigenvalues of the system. They represent the speed of the waves propagating in the system. If the values of these are real values, we have a so-called hyperbolic system. The sign of these waves (propagating upstream or downstream) will also determine how the flow conditions should be handled at the boundaries.

Under the condition that liquid is incompressible and that $\alpha_g \rho_g \ll \alpha_l \rho_l$, in the two-phase region where $\alpha_g \in (0,1)$ the following approximation for sound velocity can be derived [16].

$$\omega^2 = \frac{\rho_g a_g^2}{\alpha_g \rho_l (1 - K \alpha_g)} \quad 2.18$$

The eigenvalues are given as:

$$\lambda_1 = v_l - \omega \quad 2.19$$

$$\lambda_2 = v_g \quad 2.20$$

$$\lambda_3 = v_l + \omega \quad 2.21$$

(2.20) is the wave speed of the gas volume wave that travels downstream and (2.19) and (2.21) represent the pressure pulses propagating upstream and downstream. When gas fraction is zero ($\alpha_g = 0$) the eigenvalues are given as:

$$\lambda_1 = v_l - a_l \quad 2.22$$

$$\lambda_3 = v_l + a_l \quad 2.23$$

(2.22) and (2.23) correspond to pressure pulses propagating upstream and downstream, respectively. a_l is the sound velocity in the liquid phase ($a_l = 1500 \text{ m/s}$). Correspondingly, in pure gas regions ($\alpha_g = 1$):

$$\lambda_1 = v_g - a_g \quad 2.24$$

$$\lambda_3 = v_g + a_g \quad 2.25$$

Where a_g is the sound velocity in the gas phase ($a_g = 316 \text{ m/s}$).

2.4 Characteristics

One-phase Flow

The characteristics for one-phase flow in the (z,t) plane are defined by:

$$C1: \quad \frac{dz}{dt} = \lambda_1 = v_l - a_l \quad 2.26$$

$$C2: \quad \frac{dz}{dt} = \lambda_2 = v_l + a_l \quad 2.27$$

The characteristics show how information is propagating in the system. Along the linearized characteristics, it is possible to transform the partial differential equations into a system of ordinary differential equations named compatibility relations. They will represent the information that is carried along the characteristic. Whether a characteristic is leaving or entering the computational domain will have impact on how the boundary should be treated. Whether a characteristic is leaving or entering will be determined by the sign of the eigenvalue.

Numerical and physical boundary conditions

At the boundary of the computational domain, one of the characteristics leaves the domain and brings information from the computational domain toward the boundary. This is called a numerical boundary condition. Hence, a numerical technique is needed to bring information toward the boundary. In this thesis both extrapolation techniques and use of compatibility relations will be explored. Additionally, another characteristic enters the domain that brings information from the boundary toward the interior. This data must be given by the physics of the problem and is called the physical boundary condition. [7]

In Fig. (2.1), the characteristics can be seen entering and leaving the domain at the inflow boundary. The information that is transported from the inside of the domain ω_1 represents the numerical boundary condition. ω_2 is information entering the domain and must be given, which represents the physical boundary condition [7].

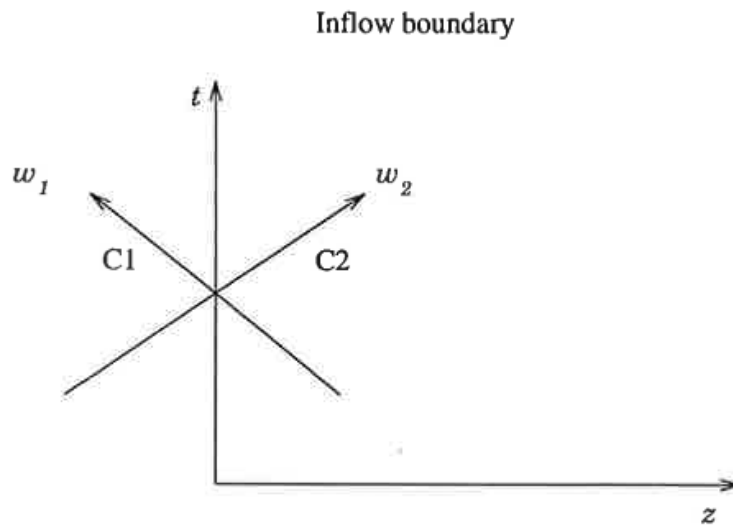


Figure 2.1: One-phase inflow boundary [3].

In Fig. (2.2), the characteristics entering and leaving the domain can be seen at an outflow boundary. Here, w_2 is the numerical boundary condition and must be calculated. w_1 is the information that enters the domain and must be given [3].

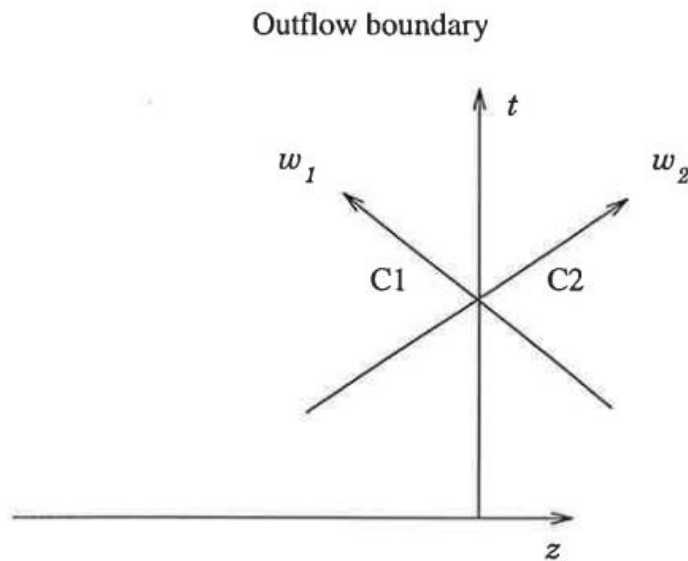


Figure 2.2: One-phase outflow boundary [3].

Two-phase flow

The characteristics for two-phase flow in the (z,t) plane:

$$C1: \frac{dz}{dt} = \lambda_1 = v_g \quad 2.28$$

$$C2: \frac{dz}{dt} = \lambda_2 = v_l + \omega \quad 2.29$$

$$C3: \frac{dz}{dt} = \lambda_3 = v_l - \omega \quad 2.30$$

Where ω is given by Eq. (2.18).

In two-phase flow, two of the characteristics will correspond to two physical boundary conditions and one will be the numerical condition if it's inflow. For outflow, there will be two numerical conditions and one physical condition.

Numerical and physical boundary conditions

In Fig. (2.3), the characteristics for two-phase flow at inlet is shown. w_1 and w_2 are entering the domain and must be given. While w_3 is the numerical condition and must be calculated.

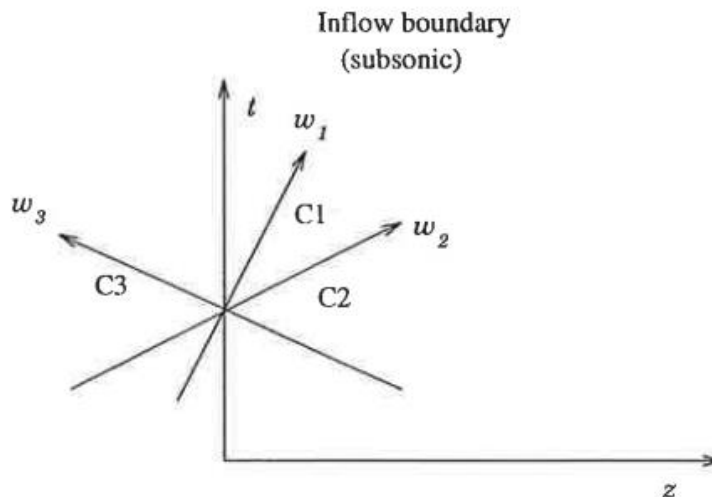


Figure 2.3: Two-phase inflow boundary [3].

In Fig. (2.4), the characteristics for two-phase flow at the outlet can be seen. ω_1 and ω_2 are leaving the domain and must be calculated, i.e. the numerical boundary condition. While ω_3 is entering the domain and must be given, i.e. the physical condition.

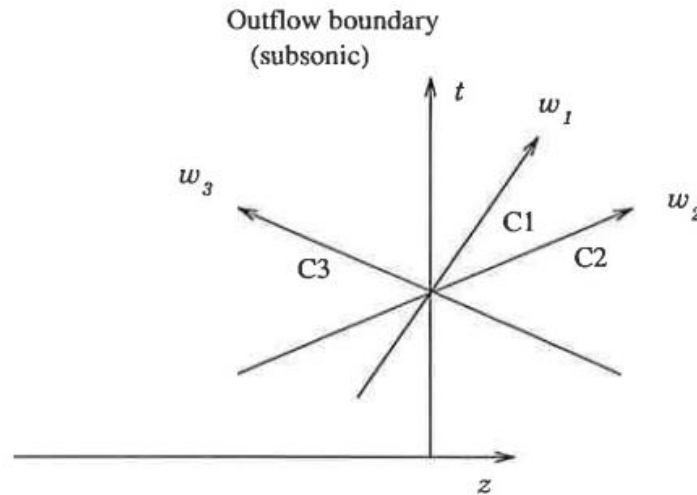


Figure 2.4: Two-phase outflow boundary [3].

2.5 Discretization

When applying conservation and closure laws in a simulation model, the well is divided into M number of segments where each cell has a length of Δz . The number one cell is at inlet (bottom) and M at the outlet (top). Increasing the number of cells in the model will increase the accuracy of the solution but will require more computational time to execute. The equations will be solved in each segment and flow variables will be considered constant for each cell. A discretization typically consists of 50 to 100 segments. This will for instance ensure that the local variations in temperature and pressure are reflected in the density calculations which will have an impact on the total hydrostatic pressure in the well [4].

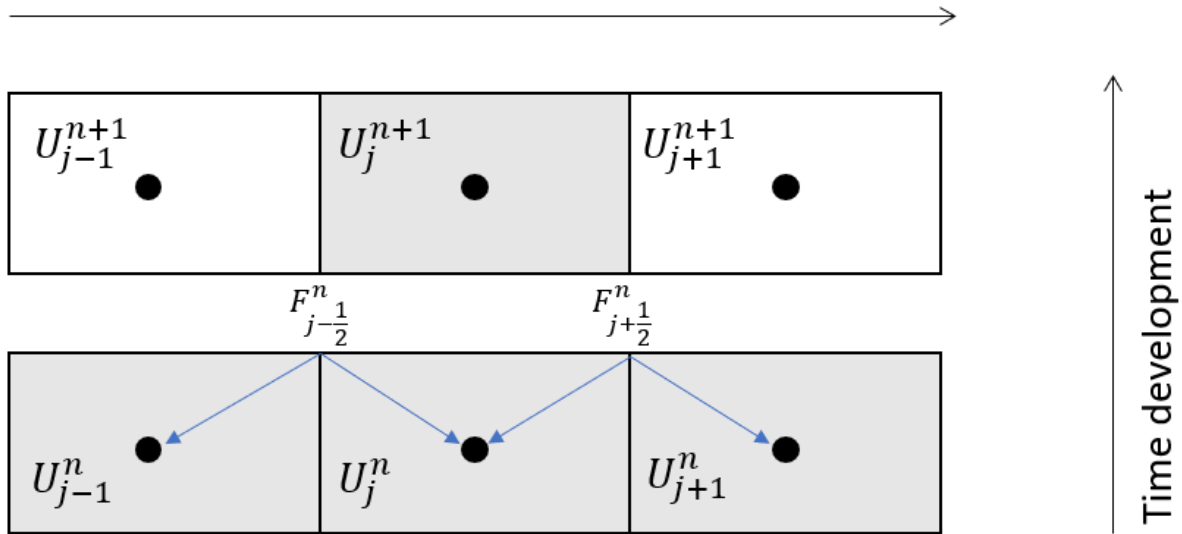


Figure 2.5: Discretization in an explicit AUSMV scheme [4].

Each cell is updated in time using the expression:

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta z} (F_{j+1/2}^n - F_{j-1/2}^n) - \Delta t Q_j^n \quad 2.31$$

Here U represents the conservative variables defined by the conservation laws while F represents the numerical fluxes across the boundaries. Δt represents the timestep chosen. When the flux F is calculated based on old values at time level n , the scheme is explicit. In this case, the scheme has to satisfy the CFL condition:

$$\Delta t = \frac{CFL * \Delta z}{\max(\lambda)} \quad 2.32$$

This will ensure that the characteristic cannot pass more than one cell for each time step. CFL is a number between 0 and 1. One must note that the conservative variables are defined in the midpoint at the cells. This is important to have in mind when treating the boundary.

The simulation starts the calculation at a known initial stage where all flow variables either are known and/or can be calculated. From there on, every cell is updated in time. The newly calculated variables of mass and momentum in the cells will depend on the fluxes of mass and momentum at cell boundaries. The cells are updated at the new time level for the current well situation at the new time level [4]. There are many different numerical methods available to base the simulation on. For the work in this thesis, the AUSMV scheme is used which is an explicit scheme [2]. In the explicit scheme, the variables are updated in time based on variables from the previous time level. In addition, for the AUSMV scheme, it's vital to have a solid understanding of sonic wave propagation speed in the fluid mixture to get accurate simulation results [5].

2.6 Primitive Variables

When calculating the pressure and phase-volumes for a new time level, $U_{1,j}$ and $U_{2,j}$ are used to determine the new values in each cell. Eq. (2.15) and (2.16) are used to calculate the liquid and gas densities for the pressure at new time level. The momentum-conservative variable $U_{3,j}$ is used to calculate the phase velocities in combination with Eq. (2.17). The next equations show how the phase-volume fractions and velocities are calculated [8]:

$$\alpha_{g,i}^{n+1} = \frac{U_{2,i}^{n+1}}{\rho_{g,i}^{n+1}} \quad 2.33$$

$$\alpha_{l,i}^{n+1} = 1 - \alpha_{g,i}^{n+1} \quad 2.34$$

$$U_{3,i}^{n+1} = (U_1 v_l)_j^{n+1} + (U_2 v_g)_j^{n+1} \quad 2.35$$

From Eq. (2.17)

$$v_g = K(\alpha_l v_l + \alpha_g v_g) + S \rightarrow v_g = \frac{K\alpha_l v_l + S}{1 - K\alpha_g} \quad 2.36$$

Substituting v_g from Eq. (2.36) into Eq. (2.35).

$$v_{l,j}^{n+1} = \frac{U_{3,j}^{n+1}(1 - K\alpha_{g,j}^{n+1}) - SU_{2,j}^{n+1}}{U_{1,l}^{n+1}(1 - k\alpha_{g,j}^{n+1}) + U_{2,j}^{n+1}K\alpha_{l,j}^{n+1}} \quad 2.37$$

2.7 Well Status

The following presents how the numerical boundary treatment has been presented in previous papers – [6], [9], and [10].

Open Well

At the inlet boundary of the well (bottom), the mass flowrates for the two different phases will be inputs and therefore known. With this information, both mass and convective momentum fluxes can be determined. However, the inlet pressure or pressure flux must be calculated, and the following expression can be used [6]:

$$P_{inlet} = P_1 + \frac{\Delta z}{2} \rho_{mix} g \cos \theta + \frac{\Delta z}{2} F_w \quad 2.38$$

While at the outlet boundary (top), both mass and convective momentum fluxes have been extrapolated using the middle value in the boundary cell [6]. One should keep in mind that only a zero-order extrapolation have been used here (will be explained later). However, if the values are changing much from cell to cell (a large gradient), a zero-order extrapolation may be too simple. For instance, at the top of the well, a gas kick migrating will experience large expansion. The outlet pressure flux has been set to atmospheric condition as it's open to the environment.

Closed Well

The same Eq (2.38) is also used for finding the inlet pressure flux and the values for mass and convective momentum fluxes are set to zero. The outlet pressure flux will be defined by the expression [6]:

$$P_{outlet} = P_M - \frac{\Delta z}{2} \rho_{mix} g \cos \theta - \frac{\Delta z}{2} F_w \quad 2.39$$

It must also be mentioned that in paper [8], one used first-order extrapolation techniques instead of Eq. (2.32) and Eq. (2.38) to define the pressures.

$$P_{inlet} = P_1 + \frac{1}{2}(P_1 - P_2) \quad 2.40$$

$$P_{outlet} = P_M + \frac{1}{2}(P_M - P_{M-1})$$

2.41

The extrapolation technique will be explained more in general later.

2.8 Water Hammer Effect

Water hammer effect is something that people can experience in everyday life without giving it a second thought. By closing the water tap quickly it's possible to hear a thud and this thud is a result of moving fluids coming to an immediate stop. By stopping moving fluids in an instant there is a large pressure spike in the system, and this is not dangerous in normal household pipes. However, in an industrial scale, it can very quickly destroy equipment and become deadly.

Generally, it is wise to slowly shut off pipes that have liquid flowing as the pressure spike can severely wear down equipment. In [14], Yuan et al. investigated water hammer effect when shutting in riser when rapid loading event was taking place and if the equipment could handle the pressure increase. Many factors were investigated in the paper, including oil-based mud, water-based mud, size of influx volume and shut-in time. The paper concludes that shutting in a riser in approximately 10 seconds the water hammer effect did not contribute too significantly high pressure when closing the riser during rapid unloading [14]. For this thesis, closing of the BOP will take place early in the simulation and the flow has not at this stage increased much so the water hammer effect is reduced.

3. AUSMV Scheme

The AUSMV scheme is a hybrid flux-vector splitting scheme (FVS) and is short for “Advection Upstream Splitting Method” and the V is for velocity splitting functions. The AUSMV scheme was specifically developed for use in petroleum engineering applications. Compared to other numerical methods, it does not rely on thorough mathematical analysis of the models and is simple to implement. However, it does require an approximate expression for the sound velocity [5].

3.1 Numerical Flux Formulas for AUSMV

The numerical FVS flux is presented in [5] and are as follows:

$$F_{j+1/2}^{FVS}(\mathbf{U}_L, \mathbf{U}_R) = (\alpha_l \rho_l)_L \Psi_{l,L}^+ + (\alpha_l \rho_l)_R \Psi_{l,R}^- + (\alpha_g \rho_g)_L \Psi_{g,L}^+ + (\alpha_g \rho_g)_R \Psi_{g,R}^- + (F_p)_{j+1/2} \quad 3.1$$

Where $F_p = (0, 0, p)^T$ and

$$\Psi_{l,L}^+ = \Psi_l^+(v_{l,L}, c_{j+1/2}), \quad \Psi_{l,R}^- = \Psi_l^-(v_{l,R}, c_{j+1/2}) \quad 3.2$$

Where

$$\Psi_l^+(v, c) = V^+(v, c) \begin{pmatrix} 1 \\ 0 \\ v \end{pmatrix}, \quad \Psi_l^-(v, c) = V^-(v, c) \begin{pmatrix} 1 \\ 0 \\ v \end{pmatrix} \quad 3.3$$

And in the same manner for gas phase:

$$\Psi_{g,L}^+ = \Psi_g^+(v_{g,L}, c_{j+1/2}), \quad \Psi_{g,R}^- = \Psi_g^-(v_{g,R}, c_{j+1/2}) \quad 3.4$$

And

$$\Psi_g^+(v, c) = V^+(v, c) \begin{pmatrix} 0 \\ 1 \\ v \end{pmatrix}, \quad \Psi_g^-(v, c) = V^-(v, c) \begin{pmatrix} 0 \\ 1 \\ v \end{pmatrix} \quad 3.5$$

The velocity splitting formulas V^+ are defined as:

$$V^\pm(v, c) = \begin{cases} \pm \frac{1}{4c} (v \pm c)^2 & \text{if } |v| \leq c \\ \frac{1}{2} (v \pm |v|) & \text{otherwise} \end{cases} \quad 3.6$$

The AUSMV replaces the velocity splitting functions V^+ by a more general pair \tilde{V}^+ . The new velocity splitting function is defined as:

$$\tilde{V}^\pm(v, c, \chi) = \begin{cases} \chi V^\pm(v, c) + (1 - \chi) \frac{v \pm |v|}{2}, & |v| \leq c \\ \frac{1}{2} (v \pm |v|) & \text{otherwise} \end{cases} \quad 3.7$$

The pressure term F_p is the same as in the FVS type discretization:

$$p_{j+1/2} = P^+(v_L, c_{j+1/2})p_L + P^-(v_R, c_{j+1/2})p_R \quad 3.8$$

The parameter χ must be specified to get “good” numerical fluxes and is the main challenge as χ defines a whole family of AUSMV schemes [5]. In [5], the following was proposed:

$$\chi_L = \alpha_R, \quad \chi_R = \alpha_L \quad 3.9$$

The parameter c is the approximation of sound velocity and is associated with a gas liquid mixture. c is defined as:

$$c(\alpha_g) = \begin{cases} \min(\alpha_l, \omega), & \text{if } \alpha_g < 0.5 \\ \min(\alpha_g, \omega), & \text{if } \alpha_g > 0.5 \end{cases} \quad 3.10$$

The AUSMV scheme has replaced the velocity splitting function V^+ with a more general \tilde{V}^+ in the convective flux part and with the weighting functions χ_L, χ_R . The conservative variables in the different cells at the new time level are found with the following expression:

$$U_{i,j}^{n+1} = U_{i,j}^n - \frac{\Delta t}{\Delta x} (F_{j+1/2}^{AUSMV} - F_{j-1/2}^{AUSMV}) - \Delta t q_i^n \quad 3.11$$

Where $U_{i,j}$ is the mass conservative variables and the mixture momentum conservative variable and q_i is the sum of external forces. F^{AUSMV} represents mass and momentum fluxes defined by the formulas above. i, j, n are conservative variables, cell number and time level, respectively. n is old time level and $n + 1$ will be the new time level.

3.2 Boundary Treatment

The AUSMV scheme has a hyperbolic nature and therefore one must be careful treating the boundaries in the model. It is vital that the information going in and out of the system to be precise. Before the work on this thesis, the simulation model has used an extrapolation method to find the boundary flux values (e.g. used in [6], [8] and [9]), as the AUSMV scheme itself does not have formulas to calculate them. The main principle has been to impose the variables that are physically determined by the system and extrapolate the variables that are unknown using the values defined by the mid values in the boundary cells. There are slightly different methods to handle the boundary depending on the status of the well [6]. However, the work in this thesis has been to implement the ideas from [3] to compare how and if compatibility relations improves the simulation model.

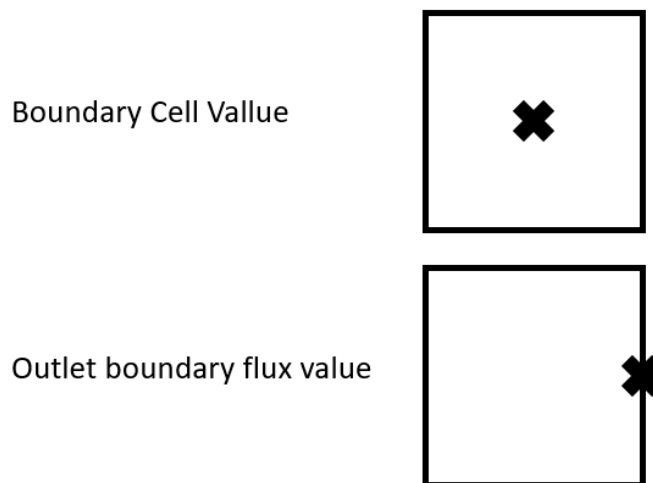


Figure 3.1: Difference between boundary cell value and outlet boundary flux value.

3.3 Extrapolation

The boundary cells in the AUSMV scheme have been estimated using extrapolation methods. There are many different forms of extrapolation methods can be applied to schemes but only a few different relevant ones will be listed. These methods can be applied to any set of variables (conservative, primitive or characteristic) and first order is generally sufficient for second-order schemes [7]. The illustration and formulas below are from [7] and give a quick overview of the difference between zero-order and first-order extrapolation.

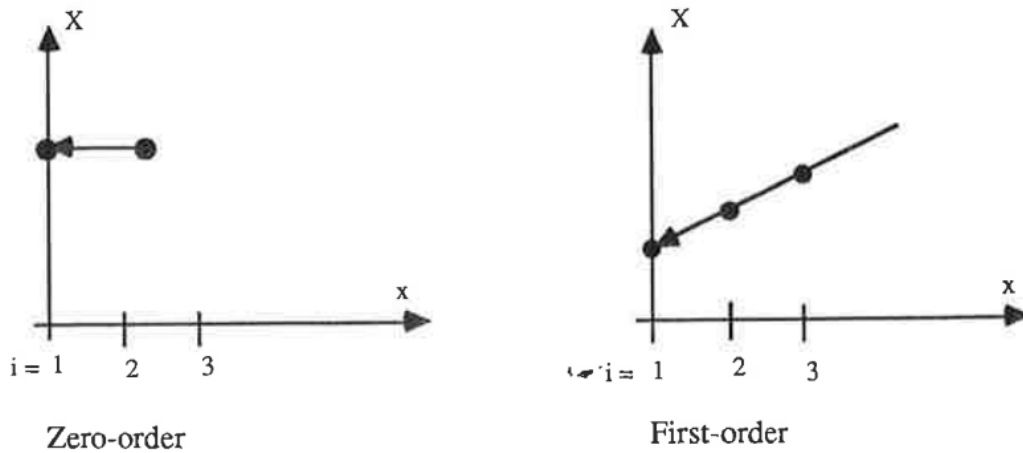


Figure 3.2: Illustration of different forms of variable extrapolation. Space extrapolation of variable X at fixed time [7].

A. For space extrapolation

Zero-order extrapolation

$$X_M^{n+1} = X_{M-1}^{n+1} \quad \text{or} \quad \Delta X_m = \Delta X_{m-1} \quad 3.12$$

Where

$$\Delta X = X^{n+1} - X^n \equiv \Delta X^n \quad 3.13$$

First-order extrapolation

$$X_M^{n+1} = 2X_{M-1}^{n+1} - X_{M-2}^{n+1} \quad \text{or} \quad \Delta X_M^n = 2\Delta X_{M-1}^n - \Delta X_{M-2}^n \quad 3.14$$

B. For space-time extrapolation

Zero-order extrapolation

$$X_M^{n+1} = X_{M-1}^{n+1} \quad \text{or} \quad \Delta X_M^n = \Delta X_{M-1}^{n-1} \quad 3.15$$

First order space/zero order in time

$$X_M^{n+1} = 2X_{M-1}^n - X_{M-2}^n \quad \text{or} \quad \Delta X_M^n = 2\Delta X_{M-1}^{n-1} - \Delta X_{M-2}^{n-1} \quad 3.16$$

First order in space and time

$$X_M^{n+1} = 2X_{M-1}^n - X_{M-2}^{n-1} \quad \text{or} \quad \Delta X_M^n = 2\Delta X_{M-1}^{n-1} - \Delta X_{M-2}^{n-2} \quad 3.17$$

C. Time extrapolation

Zero order

$$X_M^{n+1} = X_M^n \quad \text{or} \quad \Delta X_M = 0 \quad 3.18$$

First order

$$X_M^{n+1} = 2X_M^n - X_M^{n-1} \quad \text{or} \quad \Delta X_M^n = \Delta X_M^{n-1} \quad 3.19$$

The space-extrapolation method can either be used with an implicit or explicit treatment of the numerical boundary condition. The space-time extrapolation is, however, adapted for explicit schemes and the pure time extrapolations are better adapted for implicit schemes. As for accuracy, for linear equations, the boundary scheme may be one lower order than the interior schemes without sacrificing accuracy of the complete scheme [7].

3.3.1 Practical Implementation of Extrapolation Techniques

Inlet Boundary:

For the work in this thesis, it is necessary to use first-order extrapolation due to the large gradient involved in the calculations as a result of the big contributions from the source terms (hydrostatic pressure). For the inlet boundary it is possible to use Eq. (2.40) to calculate the inlet pressure. A zero-order extrapolation would be too simple and most likely give incorrect values. Instead, the Eq. (2.38) was chosen to find the inlet pressure. This will be more reliable at including the effect of the large pressure gradient – in the same way as first-order extrapolation.

Outlet Boundary:

For the outlet boundary, it is necessary to differentiate between open and closed well status. If one has a closed well, then the mass and momentum fluxes will naturally be zero throughout the simulation. The pressure has to be calculated, and similarly to inlet pressure, one has to decide whether to use first-order extrapolation for calculating pressure or use the momentum equation more directly. The first-order equation Eq. (2.41) can be used, but Eq. (2.39) was used for the work in this thesis.

For an open well status, the mass and momentum fluxes will not be zero and will have to be calculated. However, the pressure at outlet will be surface pressure (1 bar). In an open well, there will be a large gas expansion when a kick starts approaching the surface of the well. If a zero-order extrapolation is used, one must be able to tolerate a higher margin of error in the simulation results. The following equations show how the fluxes are defined in a zero-order extrapolation technique.

$$F_{1,Boundary,Mass}^{AUSMV} = \alpha_{l,M} \rho_{l,M} v_{l,M} \quad 3.20$$

$$F_{2,Boundary,Mass}^{AUSMV} = \alpha_{g,M} \rho_{g,M} v_{g,M} \quad 3.21$$

$$F_{3,Boundary,Momentum}^{AUSMV} = \alpha_{l,M} \rho_{l,M} v_{l,M}^2 + \alpha_{g,M} \rho_{g,M} v_{g,M}^2 + P_{outlet} \quad 3.22$$

Here we can see that the flux terms related to mass transport are using zero-order extrapolation technique. For the momentum flux, the part related to mass movement use also zero-order extrapolation while the pressure term is defined by the physical boundary condition (e.g. having atmospheric pressure at the outlet).

By using first-order extrapolation method, the simulation will be better at handling the large gradient between cells and will have a more accurate result. This approach starts by using first-order extrapolation of the physical variables toward the boundary.

$$\alpha_{l,Boundary} = \alpha_{l,M} + \frac{1}{2}(\alpha_{l,M} - \alpha_{l,M-1}) \quad 3.23$$

$$\rho_{l,Boundary} = \rho_{l,M} + \frac{1}{2}(\rho_{l,M} - \rho_{l,M-1}) \quad 3.24$$

$$v_{l,Boundary} = v_{l,M} + \frac{1}{2}(v_{l,M} - v_{l,M-1}) \quad 3.25$$

The above equations are similar for gas phase. Then we form the expression for the fluxes:

$$F_{3,Boundary,Liquid\ Mass}^{AUSMV} = \alpha_{l,Boundary} \rho_{l,Boundary} v_{l,Boundary} \quad 3.26$$

$$F_{3,Boundary,Gas\ Mass}^{AUSMV} = \alpha_{g,Boundary} \rho_{g,Boundary} v_{g,Boundary} \quad 3.27$$

$$F_{3,Boundary,Momentum}^{AUSMV} = \alpha_{l,Boundary} \rho_{l,Boundary} v_{l,Boundary}^2 + \alpha_{g,Boundary} \rho_{g,Boundary} v_{g,Boundary}^2 + P_{outlet} \quad 3.28$$

3.4 Compatibility relations

When using compatibility relations, the numerical scheme will give the solution in the interior of the domain. The outgoing characteristics compatibility relations together with the imposed physical conditions are used to determine the unknown variables at the boundary points. The compatibility relation equations are explicitly discretized in an upwind manner where the imposed physical conditions are added to the discretization [3].

There are four distinct compatibility relations methods used in this thesis and a better overview and explanation can be found in [3] and [17]. More details about how they were originally derived can be found in [16]:

- One-phase inlet
- One-phase outlet
- Two-phase inlet
- Two-phase outlet

One-phase inlet

For the one-phase inlet, the outgoing characteristics C1 can be seen in the figure below:

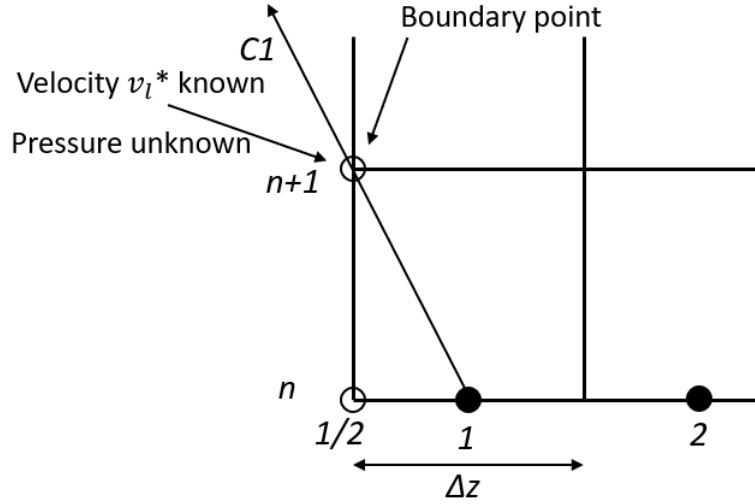


Figure 3.3: One-phase inlet boundary [3].

The velocity v_l^* is known at the inlet boundary. The corresponding compatibility equation is discretized with the imposed velocity v_l^* :

$$\frac{P_{1/2}^{n+1} - P_{1/2}^n}{\Delta t} + (v_l - a_l)_1^n \left(\frac{P_1^n - P_{1/2}^n}{\Delta z/2} \right) - (\rho_l a_l)_1^n \left[\frac{v_l^* - v_{l_{1/2}}^n}{\Delta t} + (v_l - a_l)_1^n \left(\frac{v_{l_1}^n - v_{l_{1/2}}^n}{\Delta z/2} \right) \right] = (g \rho_l a_l)_1^n + [f_{friction}(v_g - v_l + \omega)]_1^n \quad 3.29$$

This calculates the pressure for the inlet boundary for the next time step, $P_{1/2}^{n+1}$ [3].

One-phase outlet

For the one-phase outlet, the pressure P^* is known:

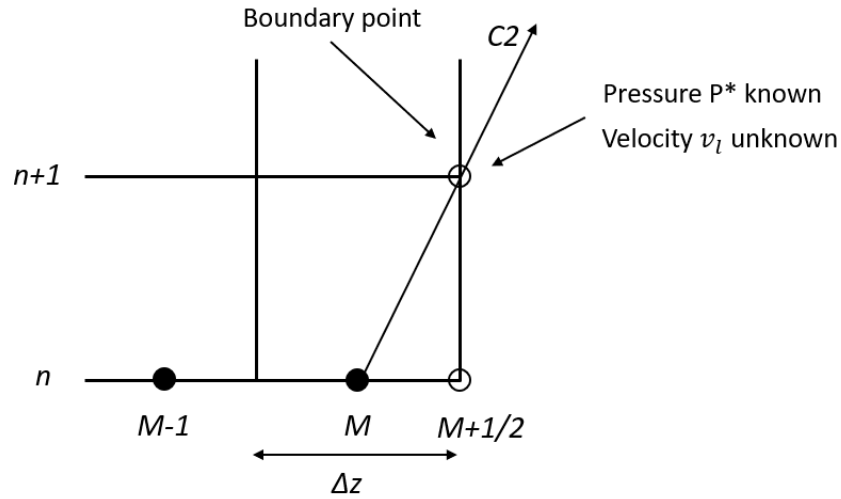


Figure 3.4: One-phase outlet boundary [3].

As the P^* is known at the outlet boundary, the compatibility equation is then discretized using the imposed outlet pressure P^* to find velocity at the new time step at the outlet boundary.

$$\frac{P^* - P_{M+1/2}^n}{\Delta t} + (v_l - a_l)_M^n \left(\frac{P_{M+1/2}^n - P_M^n}{\Delta z/2} \right) + (\rho_l a_l)_M^n \left[\frac{v_{l_{M+1/2}}^{n+1} - v_{l_{M+1/2}}^n}{\Delta t} + (v_l - a_l)_M^n \left(\frac{v_{l_{M+1/2}}^n - v_{l_M}^n}{\Delta z/2} \right) \right] = -(g \rho_l a_l)_M^n - (f_{friction} a_l)_M^n \quad 3.30$$

With this expression, it's possible to calculate the unknown velocity $v_{l_{M+1/2}}^{n+1}$ and use this to calculate the fluxes at the outlet boundary [3].

Two-phase inlet

In two-phase it becomes slightly more complex as there are more unknown variables. The mass rates at the inlet are given. From there, one can determine the superficial velocities by using the density from the previous time level. By using the gas slip relation, these can be used to determine phase velocities and gas volume fraction at the inlet $(v_l^*, v_g^*, \alpha_g^*)$

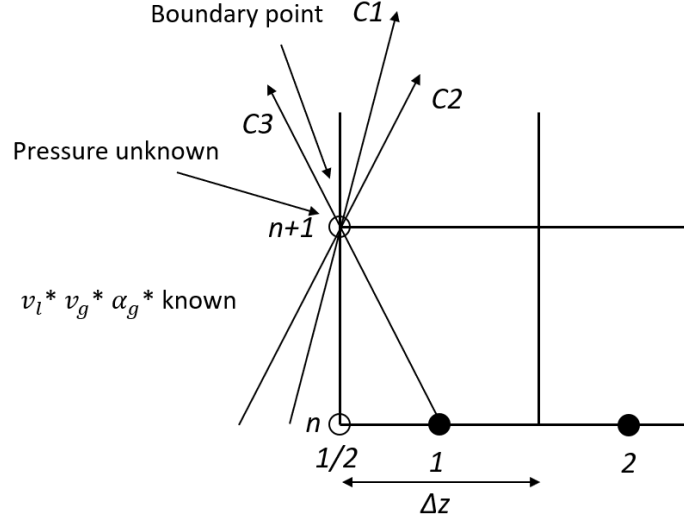


Figure 3.5: Two-phase inlet boundary [3].

By discretizing the compatibility relation, the inlet pressure $P_{1/2}^{n+1}$ is found. v_l^* and α_g^* are known variables and are added to the discretization [3].

$$\begin{aligned}
 & \frac{P_{1/2}^{n+1} - P_{1/2}^n}{\Delta t} + (v_l - \omega)_1^n \left(\frac{P_1^n - P_{1/2}^n}{\Delta z/2} \right) - (\rho_l \omega (v_g - v_l))_1^n \left[\frac{\alpha_g^* - \alpha_{g1/2}^n}{\Delta t} + \right. \\
 & \left. (v_l - w)_1^n \left(\frac{\alpha_{g1}^n - \alpha_{g1/2}^n}{\Delta z/2} \right) \right] - (\rho_l (v_g - v_l + \omega)(1 - \lambda))_1^n \left[\frac{v_l^* - v_{l1/2}^n}{\Delta t} + \right. \\
 & \left. (v_l - \omega)_1^n \left(\frac{v_{l1}^n - v_{l1/2}^n}{\Delta z/2} \right) \right] = g[(\rho_l \alpha_l + \rho_g \alpha_g)(v_g - v_l + \omega)]_1^n + \\
 & [f_{friction}(v_g - v_l + \omega)]_1^n
 \end{aligned} \tag{3.31}$$

Two-phase outlet

At the outlet boundary for two-phase, the pressure or zero velocity may be imposed as a physical boundary condition. In the experiments run in this thesis both open and closed well scenarios will be studied. For an open well, the atmospheric pressure p^* is known at the outlet boundary. Other unknown variables are $v_{lM+1/2}^{n+1}$, $v_{gM+1/2}^{n+1}$ and $\alpha_{gM+1/2}^{n+1}$ will be determined by using compatibility relations together with gas slip relation [3].

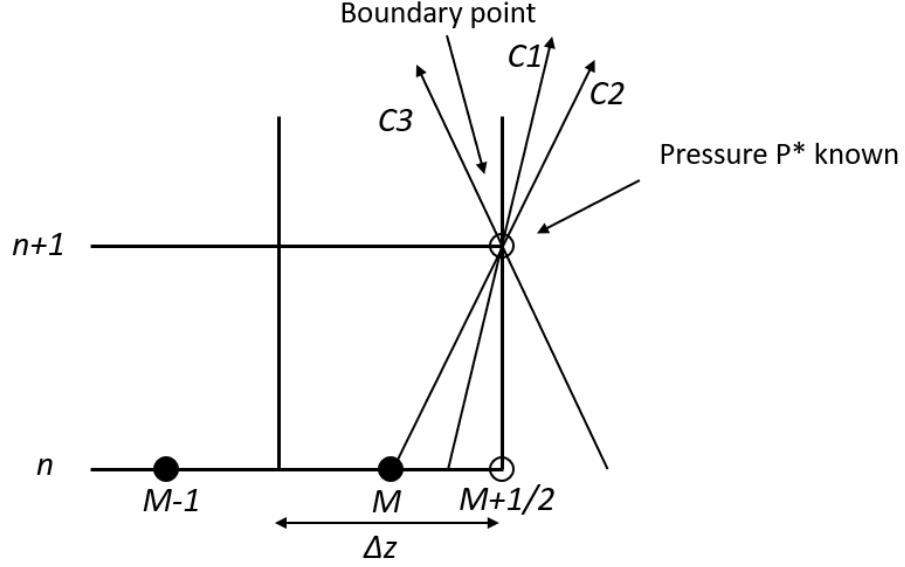


Figure 3.6: Two-phase outlet boundary [3].

The void fraction $\alpha_{g_{M+1/2}}^{n+1}$ can be found by using the following equation:

$$\begin{aligned} \frac{P^* - P_{M+1/2}^n}{\Delta t} + (v_g)_M^n \left(\frac{P_{M+1/2}^n - P_M^n}{\Delta z/2} \right) + (\rho_l \omega^2)_M^n \left[\frac{\alpha_{g_{M+1/2}}^{n+1} - \alpha_{g_{M+1/2}}^n}{\Delta t} + \right. \\ \left. (v_g)_M^n \left(\frac{\alpha_{g_{M+1/2}}^n - \alpha_{g_M}^n}{\Delta z/2} \right) \right] = 0 \end{aligned} \quad 3.32$$

The liquid velocity $v_{l_{M+1/2}}^{n+1}$ can then be found using the following equation where $\alpha_{g_{M+1/2}}^{n+1}$ is given by Eq. (3.32).

$$\begin{aligned} \frac{P^* - P_{M+1/2}^n}{\Delta t} + (v_l + w)_M^n \left(\frac{P_{M+1/2}^n - P_M^n}{\Delta z/2} \right) + (\rho_l w (v_g - v_l))_M^n \left[\frac{\alpha_{g_{M+1/2}}^{n+1} - \alpha_{g_{M+1/2}}^n}{\Delta t} + \right. \\ \left. (v_l + w)_M^n \left(\frac{\alpha_{g_{M+1/2}}^n - \alpha_{g_{M+1/2}}^n}{\Delta z/2} \right) \right] - (\rho_l \alpha_l (v_g - v_l - w))_M^n \left[\frac{v_{l_{M+1/2}}^{n+1} - v_{l_{M+1/2}}^n}{\Delta t} + \right. \\ \left. (v_l + w)_M^n \left(\frac{v_{l_{M+1/2}}^n - v_{l_M}^n}{\Delta z/2} \right) \right] = g[(\rho_l \alpha_l + \rho_g \alpha_g)(v_g - v_l - w)]_M^n + \\ [f_{friction}(v_g - v_l - \omega)]_M^n \end{aligned} \quad 3.33$$

The following equation can be used to find the gas velocity from the gas slip relation:

$$v_{g_{M+1/2}}^{n+1} = \frac{K\alpha_{l_{M+1/2}}^{n+1} v_{l_{M+1/2}}^{n+1} + S}{(1 - K\alpha_{g_{M+1/2}}^{n+1})} \quad 3.34$$

For the transition from single-phase to two-phase calculations, an if statement was used imposing single-phase calculations if $\alpha_{g_M}^n$ is less than 0.001 and two-phase calculations if the gas volume fraction was above 0.001.

In the case of a two-phase closed well, $v_{l_{M+1/2}}^{n+1}$ will equal to zero and both P^* and $\alpha_{g_{M+1/2}}^{n+1}$ will be unknowns in Eq. (3.33). In addition, due to the uncertainty of the importance of $\alpha_{g_{M+1/2}}^{n+1}$, a zero-order space-time extrapolation [7] as shown in Eq. (3.15) was used. This allows us to approximate the void fraction, $\alpha_{g_{M+1/2}}^{n+1} = \alpha_{g_M}^n$, and avoid the use of Eq. (3.32) completely for the two-phase closed well case. However, another method of solving this would be to solve Eq. (3.32) for $\frac{\alpha_{g_{M+1/2}}^{n+1} - \alpha_{g_{M+1/2}}^n}{\Delta t}$ and insert this solution into Eq. (3.33) in order to solve for P^* . With this method, one would solve for all the variables in Eq. (3.32) and Eq. (3.33) at the same time. This method may be a more accurate solution if $\alpha_{g_{M+1/2}}^{n+1}$ plays a larger role than initially assumed.

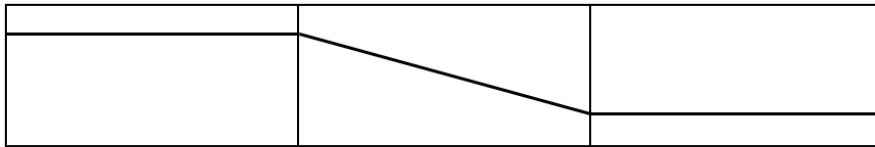
3.5 Slope Limiters

The basic AUSMV scheme is a first order scheme, which means numerical diffusion will have a significant impact on the simulation results. The numerical diffusion makes what should be very sharp edges in the liquid-gas transition very smooth and make the transition more gradual than it is supposed to be. This problem could be reduced by increasing the number of cells in the simulation but at the cost of increased computational execution time. However, a more reliable solution is to make the scheme second order by use of the slope limiter concept [11]. With this concept, the variables in a cell are no longer constants and a slope is formed throughout the cell and is used to calculate the boundary values in each cell. With the boundary values in the cell

calculated, they can be used to calculate the numerical fluxes between the cells. The figure below gives a good illustration of what it looks like in each cell [6].



First order method: Flow variables are treated as constant across a cell.



Second order method/ Slope limiter concept: Gives better resolution of gas distribution in a well.

Figure 3.7: Slope limiter concept [6].

4. Simulations & Discussion

The main idea with this thesis was to compare the use of compatibility relations with the use of both zero- and first-order extrapolation techniques. There are alternative approaches for handling the numerical boundary condition, i.e. transporting information from the interior computational domain to the boundary. The concept of compatibility relations is taken from [16] and [17] and these are defined in chapter 3.4. Here they have been implemented into the AUSMV scheme. Each simulation test case was partly taken or inspired from similar simulation cases defined before in [2] and [6]. They were chosen to cover quite different flow situations to provide a broad test basis for testing and comparing the different ways of treating the numerical boundary condition. One objective has been to test both rapid propagating pressure pulses and the more slow mass transport phenomena. For two-phase flow gas mass transport, one has considered both open and closed well condition to test both the effects. The idea has been to capture the situation where large gas volume expansion takes place (kick in open well) but also a situation where there is a 100% transition between a liquid and gas zone (a kick that has migrated to the top of the closed well). All cases used in thesis are generally very different from each other as the fluid behavior will be completely different for each case:

1. Generating pressure pulses through an open horizontal pipe.
2. Migrating kick in a vertical well that is closed at the top.
3. Migrating kick in a vertical well that is open at top.
4. Migrating kick in a vertical well that is open at top and circulating the kick out.

The horizontal pipe has no annulus and only pure liquid will be used for this case. All the cases have also been simulated with different amounts of cells. The number of cells will generally have a significant impact on the accuracy and execution time of the simulation. Three standard number of cells are used in the simulations, 25 cells, 50 cells and 100 cells for each case. For case 1, the CFL was kept constant for all simulation runs at $CFL = 0,15$. For the rest of the cases, the CFL was kept constant at $0,1875$.

Depth/Length

For the first case (1), the pipe will be horizontal, have no annulus and have a length of 10.000 meters with *modified* atmospheric pressure. In case two (2) and three (3), the well will be vertical with a depth of 4000 meters.

Diameter

The horizontal pipe in case one (1) has a diameter of 0,2 m and will result in a cross-sectional flow area of 0,0314 m². For case two (2) and three (3) the outer diameter is 0,311 m (12,25") and inner diameter is 0,127 m (5"). This gives a cross-sectional flow area of 0,0634 m² (98,22 in²).

Fluid properties

As the goal is to test the robustness of the boundary condition treatment in this thesis, water has been selected as the main fluid phase and air is used for the gas phase. These phases come with standard water and air properties.

Friction model

As case one (1) does not have an annulus, the Dukler friction factor model is used. For laminar and turbulent flow, respectively:

$$f = 16/Re \quad 4.1$$

$$f = 0,046Re^{-0,2} \quad 4.2$$

Case two (2) and three (3) has an annulus and uses an annulus friction factor model. For laminar and turbulent flow, respectively:

$$f = 24/Re \quad 4.3$$

$$f = 0,052Re^{-0,19} \quad 4.4$$

4.1 Case 1 – Horizontal Pipe Flow

The first case was set up similar to [2] with the main idea of quickly generating pressure pulses through the system that consists entirely of liquid. The entire pipe is filled with liquid and has an open end. The pump has a ramp up time of 0,5 seconds from 0 kg/s to 16,7 kg/s (1000 lpm). The ramp up will start at 1 second and end at 1,5 seconds. This quick (but unrealistic) ramp up of the pump will generate very powerful pressure pulses propagating throughout the system and should make it possible to see the effect of implementing a different boundary condition treatment. There was a slight issue with this simulation if it were run with atmospheric pressure of 1 bar. When the pressure pulse that was generated had bounced from the outlet and was on its way back, the pressure in the pipe would approach vacuum condition and possibly become negative. This would lead to an abnormal behavior as seen in the graph below:

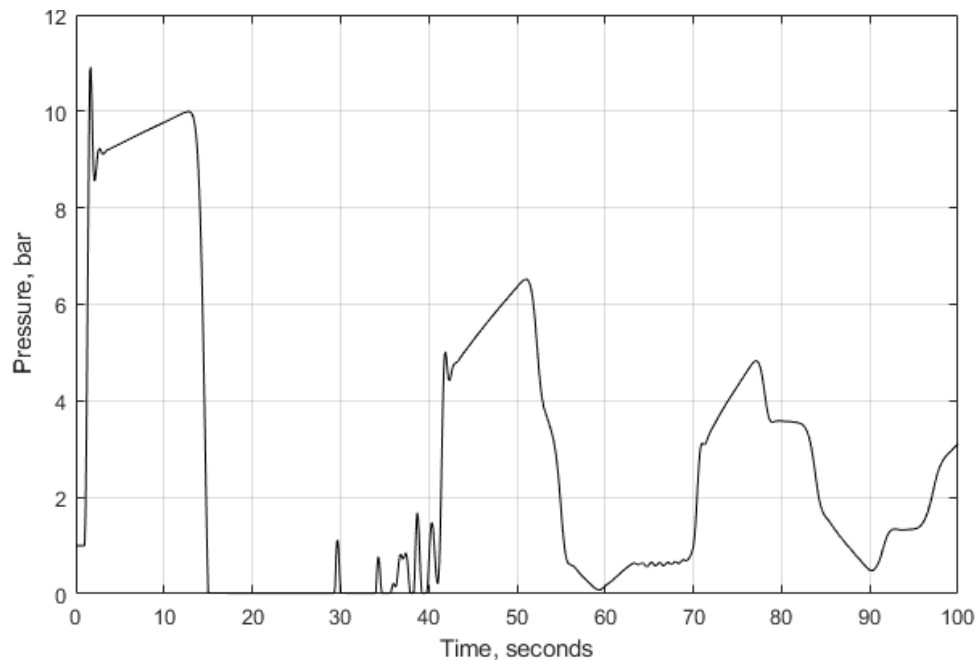


Figure 4.1: Graph showcasing the vacuum issue at inlet due to low atmospheric pressure.

There are probably several methods to work around this, but just for the sake of simplicity the atmospheric pressure for this case was set to 10 bars. After implementing the new atmospheric pressure, it is easier to see the generated pressure pulses propagating back and forth in the system. The amplitude of these will dampen when time passes due to friction effects as seen in Fig. 4.2.

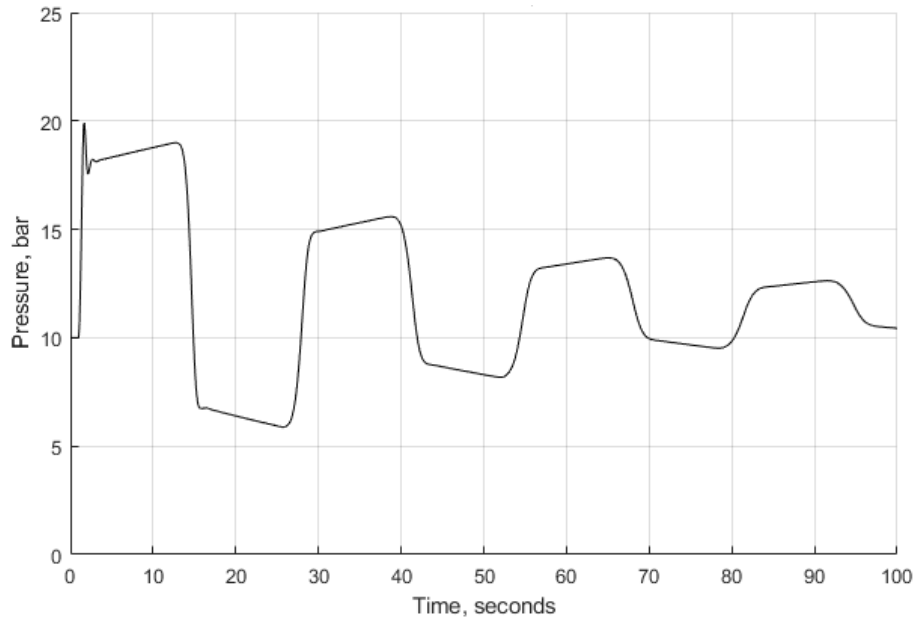


Figure 4.2: Inlet pressure vs. time with atmospheric pressure set to 10 bar.

With this implementation, it will be easier to compare the different boundary condition treatments. As case 1 has only a liquid phase, the propagating pressure pulses will have a speed of 1500 m/s and moving to the end of the pipe will take approximately 6,7 seconds. As seen in Fig. 4.2, the pressure pulse starts propagating forward at 1,0 seconds and uses about 13,3 seconds to the outlet of the pipe and back where it influences the inlet pressure of the pipe again.

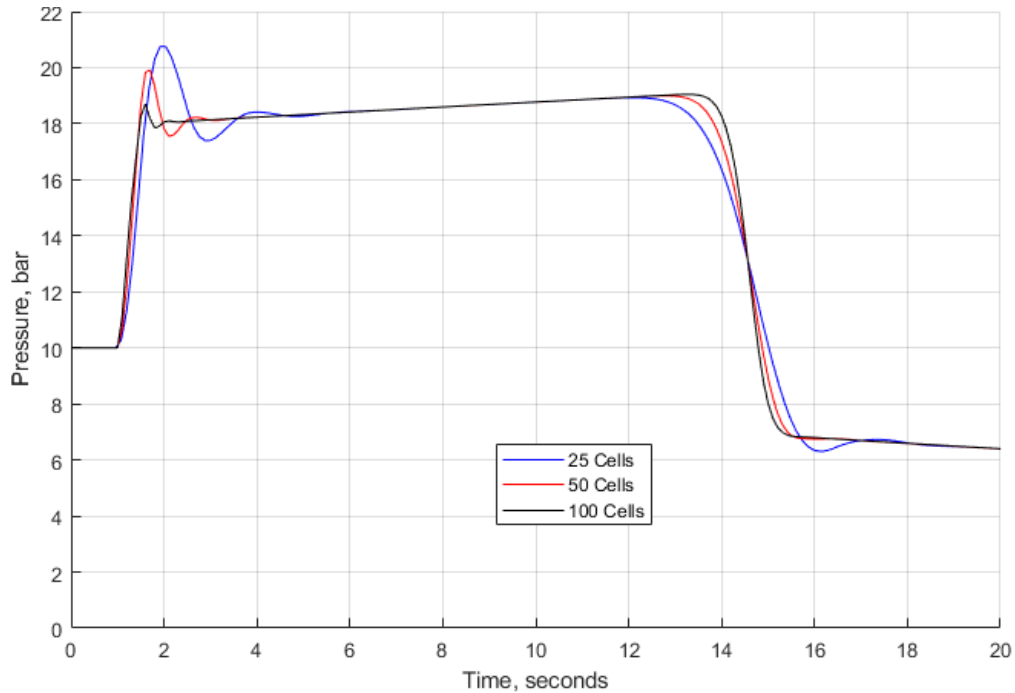


Figure 4.3: Inlet pressure vs. time of first-order extrapolation technique.

In Fig. 4.3, shows the inlet pressure using grid refinement when using first-order extrapolation technique in the model. With the lower cell count in the model, the oscillation is quite significant and is about 3 bars above the intended pressure after the pump has reached its maximum flow rate. The figure also shows that increasing the cell number for the model will reduce the oscillation issues quite significantly. In addition, by increasing the number of cells also reduces the numerical diffusion that occurs between 13 and 16 seconds. The same simulation has been run with compatibility relation method for the next graph to see the effect of different boundary condition treatment.

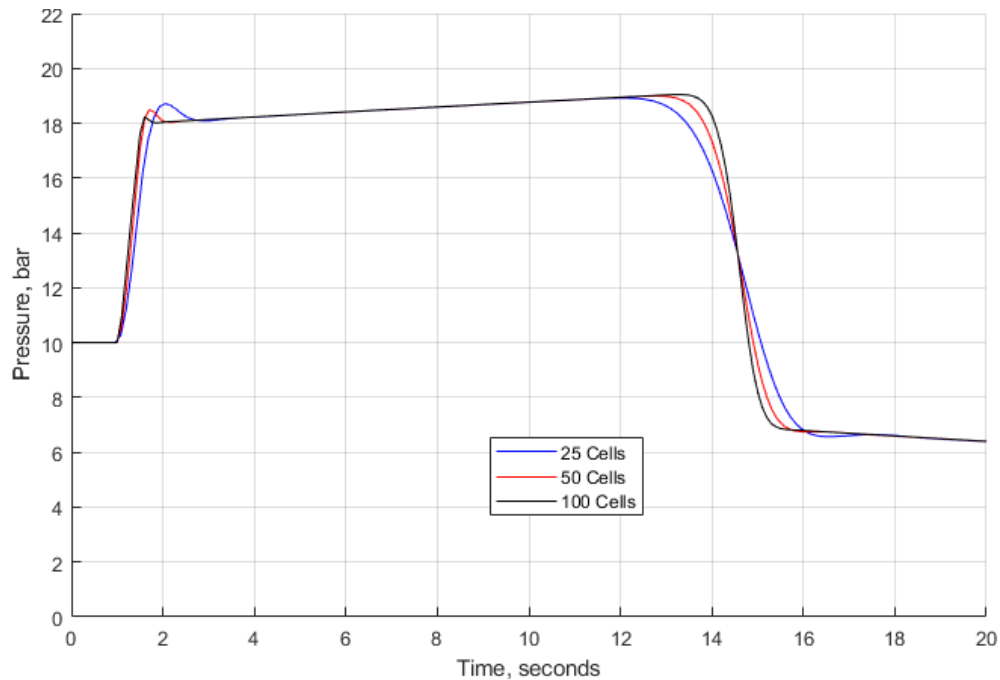


Figure 4.4: Inlet pressure vs. time of compatibility relations method.

Fig. 4.4, shows the inlet pressure vs. time of the same simulation run previously with compatibility relation method. As seen from this figure, the oscillation issue for the lower cell count grid, has been reduced by large margin and keeps improving by increasing the cell count. For 25 cells, the initial oscillation is just shy of 1 bar above the intended pressure and for the higher cell count grid, 100 cells, the oscillation is barely noticeable. However, the different boundary treatment does not affect the numerical diffusion issue occurring between 13 and 16 seconds and which is the same for both boundary condition treatment methods. Although it does not reduce the numerical diffusion, it does improve on the oscillation occurring after the pressure drop when the pressure pulse has propagated back. The large oscillation improvement here is due to the different method to treat the inlet boundary condition. The different outlet boundary condition treatment will only play a minor role as it does not affect the inlet pressure until the pressure pulse propagates back. The next graph has the two previous simulation runs plotted in the same figure to better show the improvements.

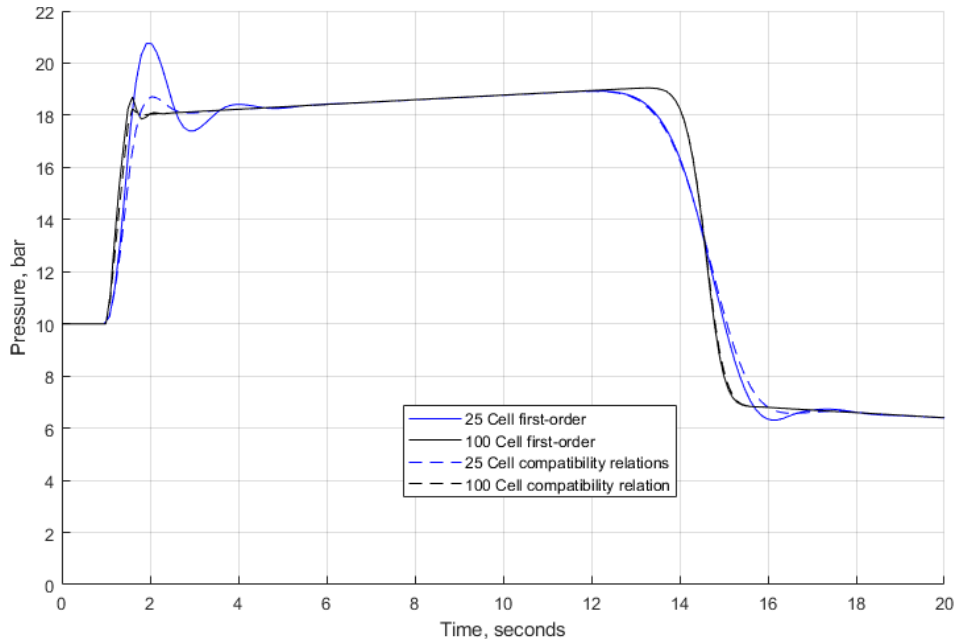


Figure 4.5: Inlet pressure vs. time for both boundary condition treatment methods.

As it clearly shows in Fig. 4.5, the different inlet boundary condition treatment improves the oscillation issue occurring early in the simulation but has practically not impact on the numerical diffusion in the simulation. In addition, it shows that the number of cells chosen for simulation has more importance than the boundary condition treatment.

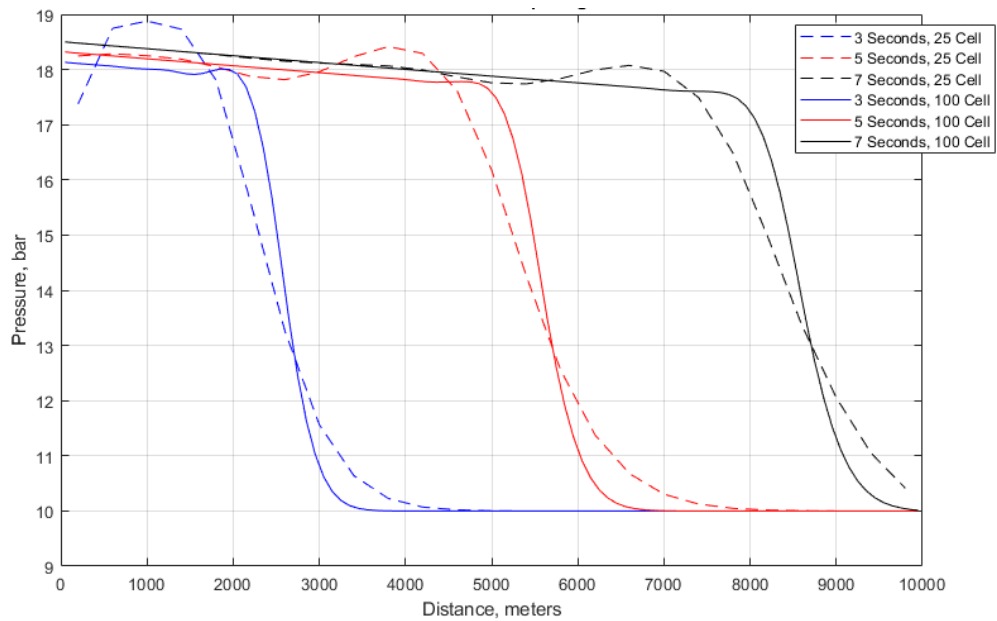


Figure 4.6: Pressure vs. distance for first-order extrapolation technique.

Fig. 4.6, shows where the pressure pulse is located throughout the pipe at specified times. This allows us to see the effect of the numerical diffusion between the 25 cell and 100 cell simulation runs. The case with the most diffusion (25 cells) will feel the pressure inlet decrease earlier when the pulse has travelled back. The pressure change that the pulse represents has been diffused in the spatial domain.

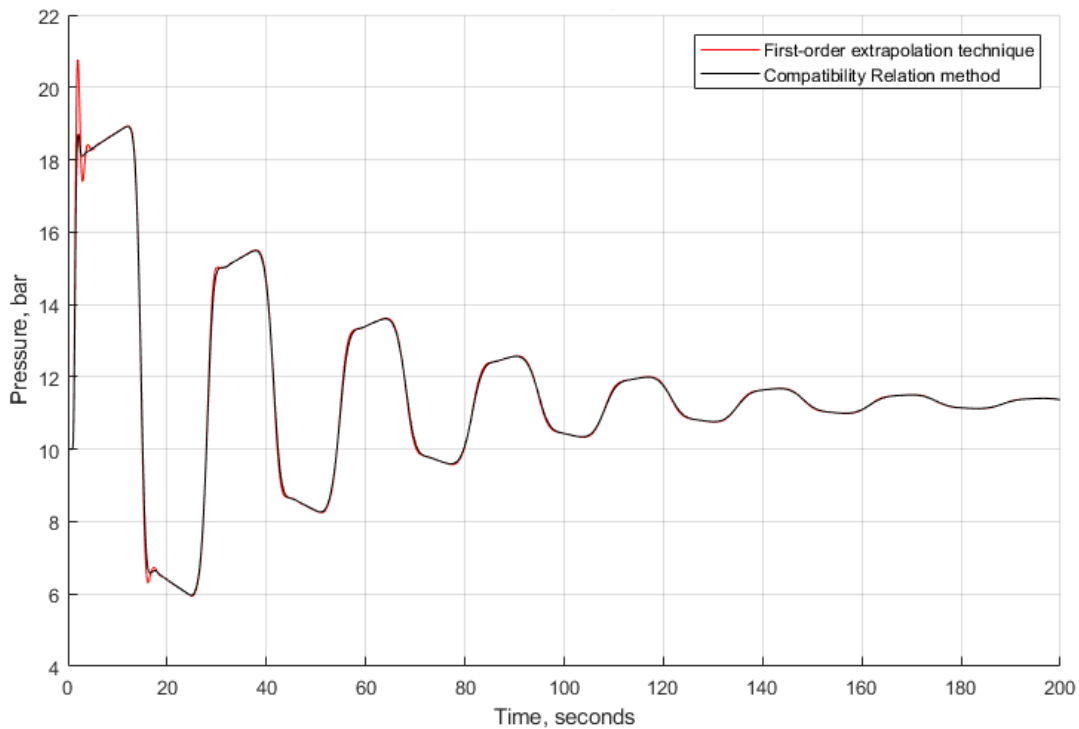


Figure 4.7: Inlet pressure vs. time comparing first-order extrapolation technique and compatibility relations method.

In Fig. 4.7, a longer simulation run of the two different boundary condition treatment methods for 25 cells is presented. As seen, the oscillation issue can be seen clearly after pump ramp up but diminishes quite quickly. From this analysis of case 1, it does appear that compatibility relations method has an advantage over the first-order extrapolation technique if a less refined grid is required. Hence, if we are to study severe impacts at sonic wave propagation (e.g. with water hammer effects etc), compatibility relations might give better results than extrapolation techniques.

4.2 Case 2 – Kick in Closed Vertical Well

Case 2 setup was similar to [6] in order to get a rigorous case to compare the results against. For the next cases the time between saving time data will be set at 1 seconds, and therefore plotted data will only have points every 1 seconds compared to 0,1 seconds in case 1. This will result in the graphs showing less of the extremely fine detail of the data but will work just fine as the next cases span over a much longer time interval compared to case 1.

For case 2, the well has a depth of 4000 meter and a geometry of 12.25" x 5" annulus. This gives an outer diameter of 0,331 m and inner diameter of 0,127 m which results in a cross-sectional area of 0,0634 m². The compressibility is such that the related sonic velocity $a_l = 1500 \text{ m/s}$ (water) and the kick influx is assumed to be an ideal gas with a sonic velocity of $a_g = 316 \text{ m/s}$. The kick volume is half of what it is in [6] for the purpose of having case 2, 3 and 4 setup similar. With a too large kick volume, the gas and liquid velocity at the outlet in an open well status may exceed the sound barrier and could make the simulation unstable.

The initial kick volume used here is 2,05 m³ and it is introduced using a gas mass rate of 8 kg/s. The well is first kept static for the first 10 seconds then the gas mass rate is ramped up linearly over a 10 second period. The gas mass rate is then kept static for the next 90 seconds until it is ramped down linearly over a 10 second period and the well is closed at 130 seconds. The gas will then start migrating upwards and lead to pressure buildup in the well until it reaches the top of the well. We start off the comparison in case 2 with a close look at the first 150 seconds of the simulation run with both first-order extrapolation order and compatibility relations method with grid refinement. With first-order extrapolation, we mean that Eq. (2.38) and (2.39) are used.

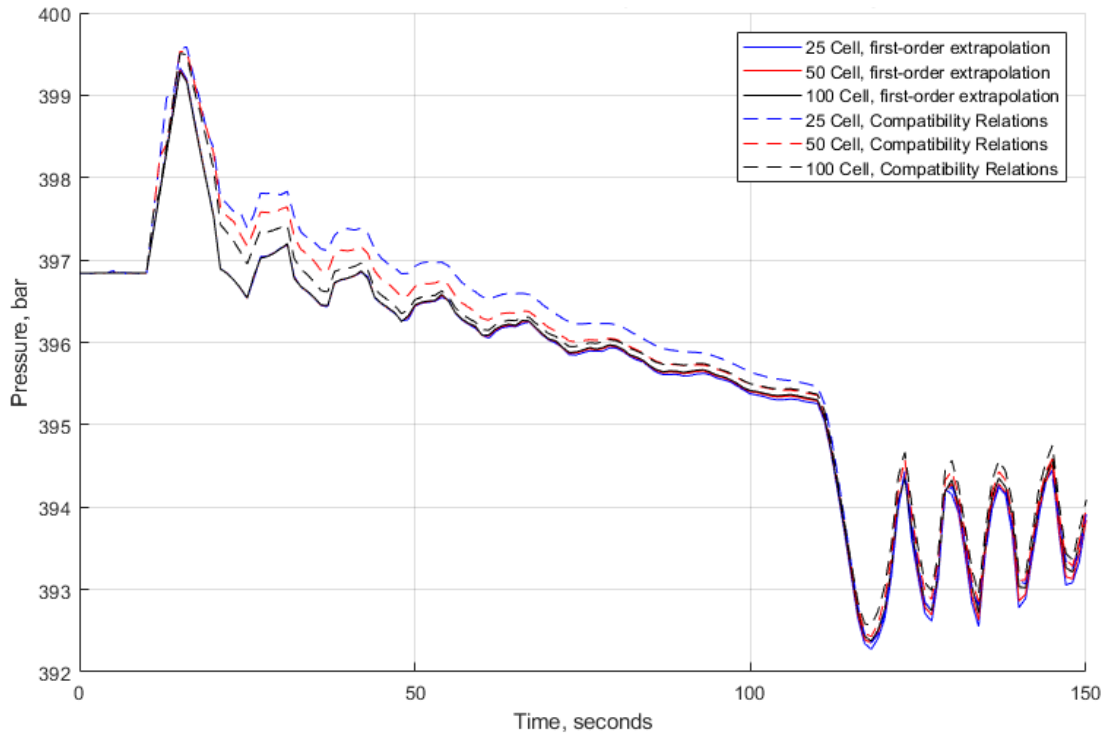


Figure 4.8: Bottomhole pressure vs. time of both first-order extrapolation technique and compatibility relation method and refinement.

As seen in Fig. 4.8, all the simulation runs using the first-order extrapolation techniques have precisely the same values for the 100 first seconds of simulation time. The compatibility relation method runs start approaching the values of first-order extrapolation runs when the grid is refined. This indicates that first-order extrapolation method does a better job at calculating the bottomhole (inlet) pressure in the early phase of this simulation run. Running this simulation over a longer time period, it does appear that the compatibility relations method using 100 cells has stability issues when the kick reaches the top of well as seen in figure below.

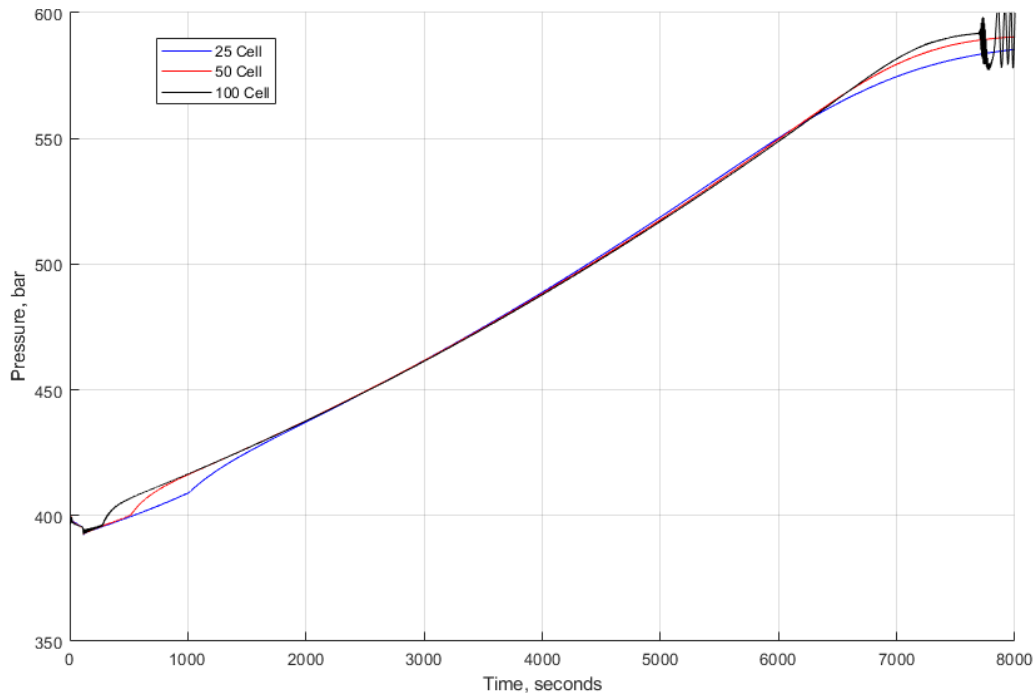


Figure 4.9: Bottomhole pressure vs. time for compatibility relations method boundary condition treatment.

In Fig. 4.9, we can see the bottomhole pressure of the simulation run to 8000 seconds using compatibility relations boundary condition treatment. The simulation behaves normal apart from the fact that the more refined grid of 100 cells started having stability issues once the kick reaches the top of the well. The cause of this is a little uncertain, but it may be possible the two-phase calculations cannot handle the high gas fraction that the 100 cells grid reaches toward the end of the simulation. A possible fix could be to introduce single gas phase compatibility relations and try to ensure a smooth transition between the two-phase and one-phase gas compatibility relations.

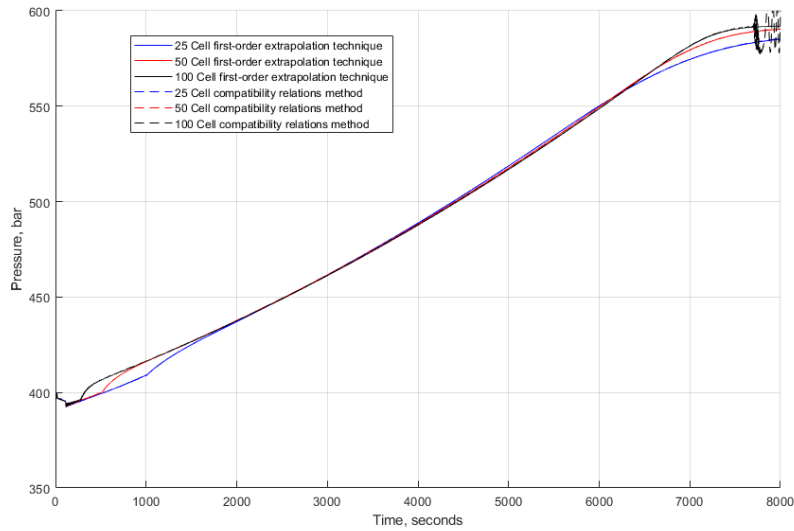


Figure 4.10: Bottomhole pressure vs. time comparing first-order extrapolation technique and compatibility relations method.

Fig. 4.10 shows the bottomhole pressure for both boundary condition treatment methods, and it shows that both methods simulate practically the same values for the bottomhole pressure. The instability issues seen in the Fig. 4.9 and 4.10 only happened for compatibility relation method boundary condition treatment and there were no issues with any of the first-order extrapolation technique runs. In addition to the instability issue, there has also been an issue with extra gas being added to all cases involving two-phase even though the well is closed and no gas is pumped into the system.

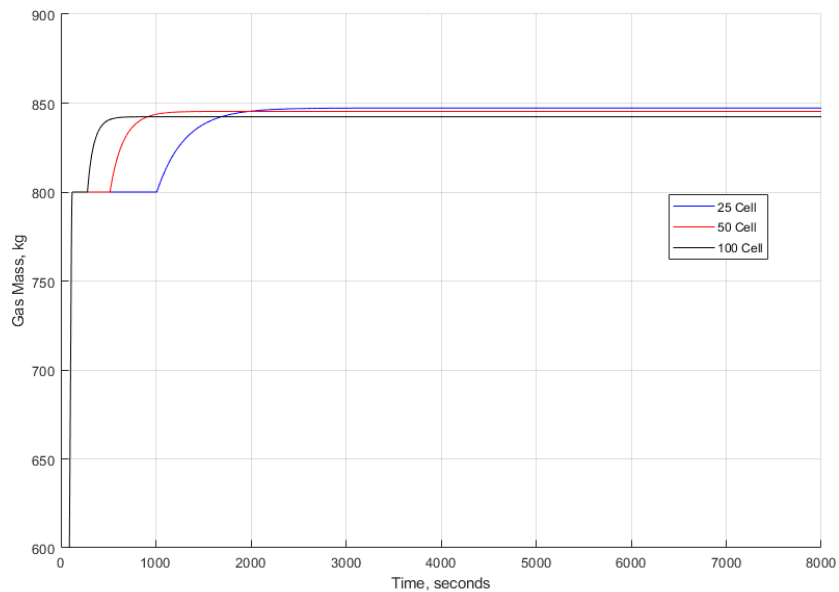


Figure 4.11: Gas mass vs. time for first-order extrapolation technique.

As seen in Fig. 4.11, there is extra gas being added for the simulation. For a 25 cells grid, the extra ~50 kg of gas is added at around 1000 seconds, but with grid refinement, it starts occurring earlier on in the simulation run. The cause of this is uncertain, but all cases with two-phase flow the same issue happens quite early in the simulation. This does show up in both the bottomhole pressure and gas volume graphs, but it has no real impact on the end of the simulation apart from being a slightly larger kick than initially introduced in the well. The issue will for the time being be overlooked and will be discussed in more detail in the next chapter.

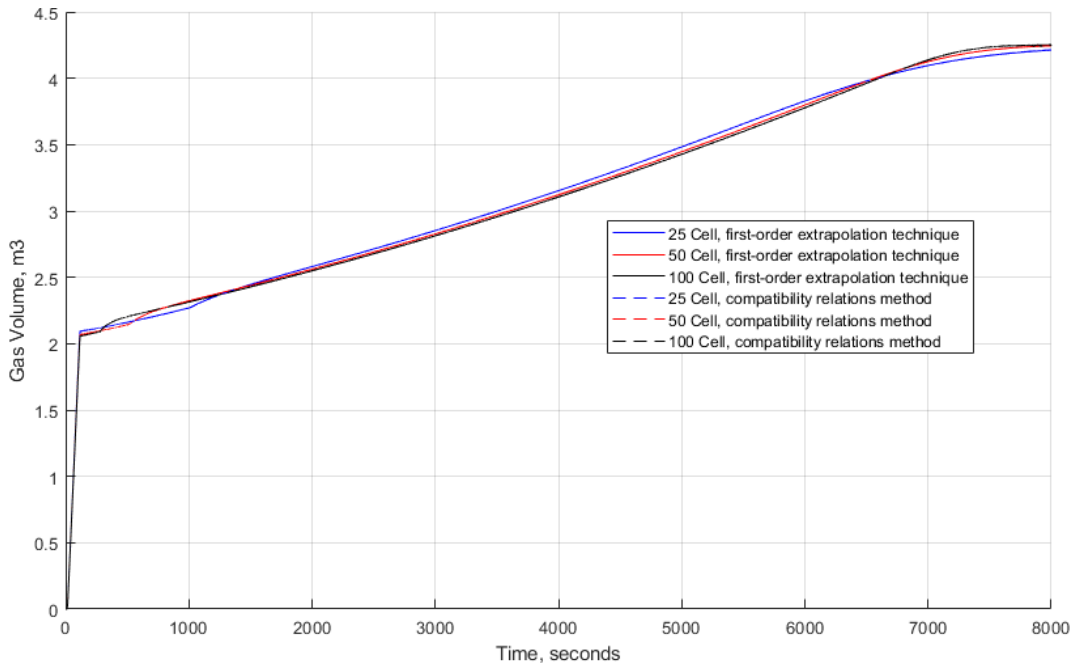


Figure 4.12: Gas volume vs. time grid refinement for first-order extrapolation technique and compatibility relation method.

Fig. 4.12, the kick volume vs time for the different simulations using both compatibility relations method and first-order extrapolation technique for different grid refinement can be seen. All simulations show that the kick volume is allowed to increase slightly since the liquid volume is compressed due to increased pressure. Both methods have very similar values for the gas volume throughout the well and cannot even be differentiated on the graph. For this case it seems hard to justify implementing compatibility relations method since the first-order extrapolation technique seems to provide a more correct result since it converges faster to a solution, especially in the early phase for the simulation. In addition, the first-order extrapolation technique is also much easier to implement.

4.3 Case 3 – Kick in Open Vertical Well

Case 3 has the same setup as case 2 with the exception that the well will remain open. This should be a rigorous numerical case for the different boundary condition treatments method to be tested for. In addition to first-order extrapolation technique and compatibility relations method, zero-order extrapolation will also be tested for the open hole well status to see how it compares.

The well has a depth of 4000 meter and a geometry of 12.25" x 5" annulus. This gives an outer diameter of 0,331 m and inner diameter of 0,127 m which results in a cross-sectional area of 0,0634 m². The compressibility is such that the related sonic velocity $a_l = 1500 \text{ m/s}$ (water) and the kick influx is assumed to be an ideal gas with a sonic velocity of $a_g = 316 \text{ m/s}$. The kick volume is half of what it is in [6] for the purpose of having case 2, 3 and 4 setup similar. If the kick volume is too large, the gas and liquid velocity at the outlet in an open well status may exceed the sound barrier and could make the simulation unstable.

The initial kick volume used here is 2,05 m³ and is introduced using a gas mass rate of 8 kg/s. The well is first kept static for the first 10 seconds then the gas mass rate is ramped up linearly over a 10 second period. The gas mass rate is then kept static for the next 90 seconds until it is ramped down linearly over a 10 second period. The gas will then start migrating upwards and lead to gas expansion and eventually liquid expulsion from the well. This will result in a large bottomhole pressure drop.

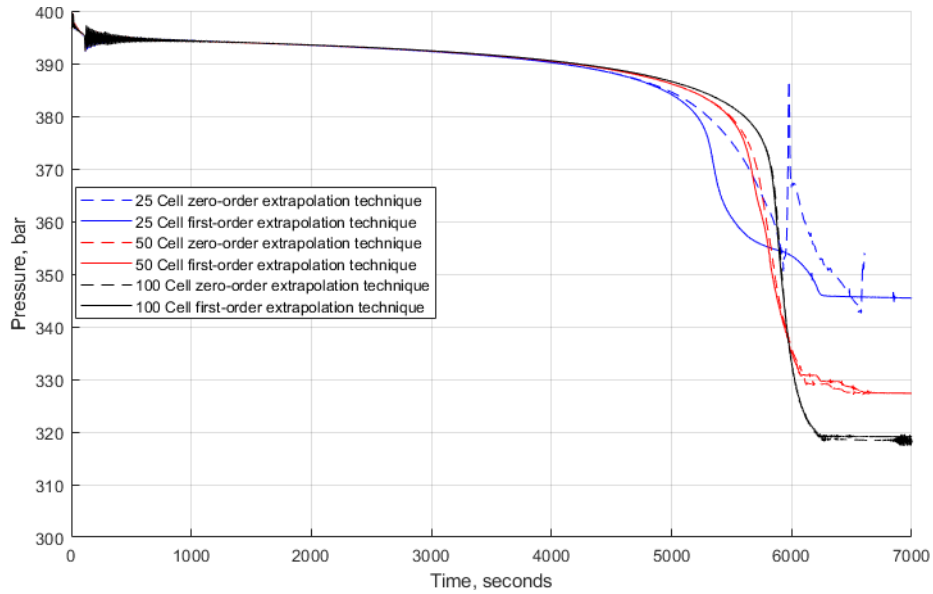


Figure 4.13: Bottomhole pressure vs time for zero-order and first-order extrapolation techniques with grid refinement.

As seen in Fig. 4.13, the bottomhole pressure can be seen plotted against time for zero-order and first-order extrapolation techniques. From these results, it shows that the zero-order extrapolation technique cannot handle this simulation with a non-refined grid (25 cells). The simulation was supposed to run until 7000 seconds, but the zero-order 25 cells grid became unreliable before 6000 seconds and at 6600 seconds the bottomhole pressure became undefined. For the less refined first-order extrapolation technique can predict the bottomhole pressure with a less refined grid, but it is about 25 bars off the results obtained with the more refined grid (100 cells). In addition, there is an inflection point before the liquid expulsion from the well has finished which can be hard to explain and may be related to the transition between one-phase and two-phase. Both the 50 and 100 cells zero- and first-order extrapolation techniques, do about equally well for predicting the bottomhole pressure.

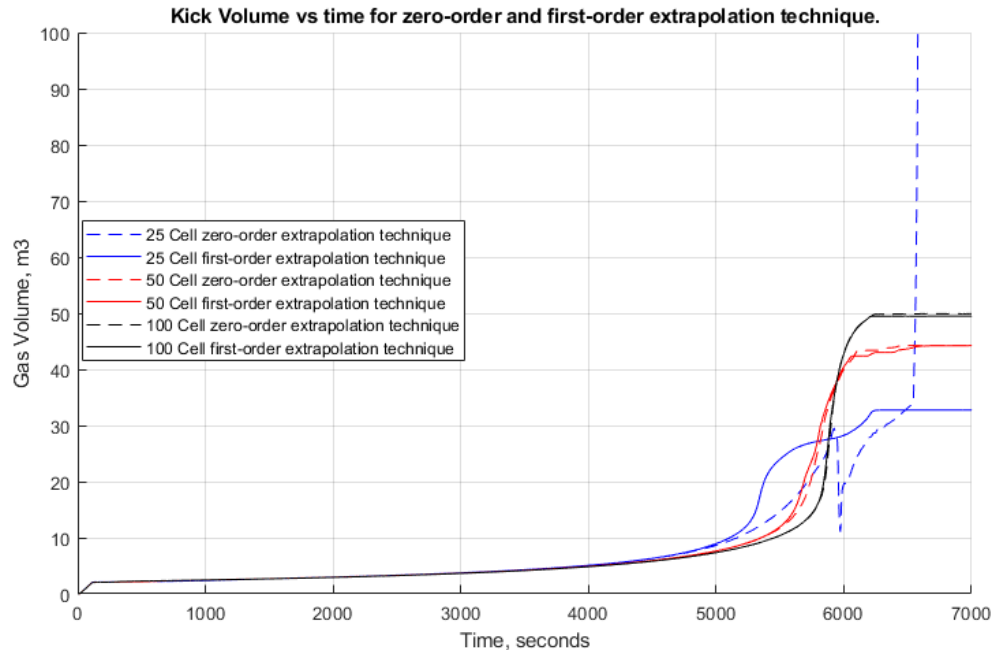


Figure 4.14: Kick volume vs. time for zero-order and first-order extrapolation techniques with grid refinement.

In Fig 4.14, the volume of gas in the well plotted against time, and the same issue as discussed for the bottomhole pressure can be seen here. The zero-order extrapolation technique becomes unstable and the kick volume goes to infinity. For the 50 and 100 cell grids, the zero- and first-order extrapolation technique predict kick volume about equally well. Similar to the bottomhole pressure figure, there is a big gap in results when comparing 25 cells and 100 cells grids in predicted final kick volume. The difference is 17m^3 . Also, here the inflection point can be seen on the less refined grid but disappears entirely with grid refinement.

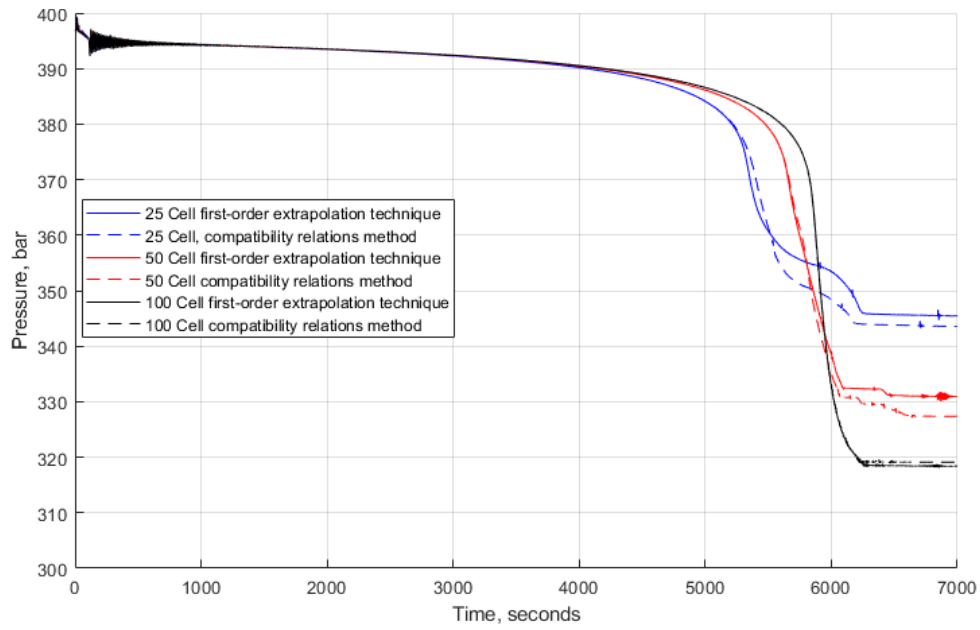


Figure 4.15: Bottomhole pressure vs. time for first-order extrapolation technique and compatibility relations method with grid refinement.

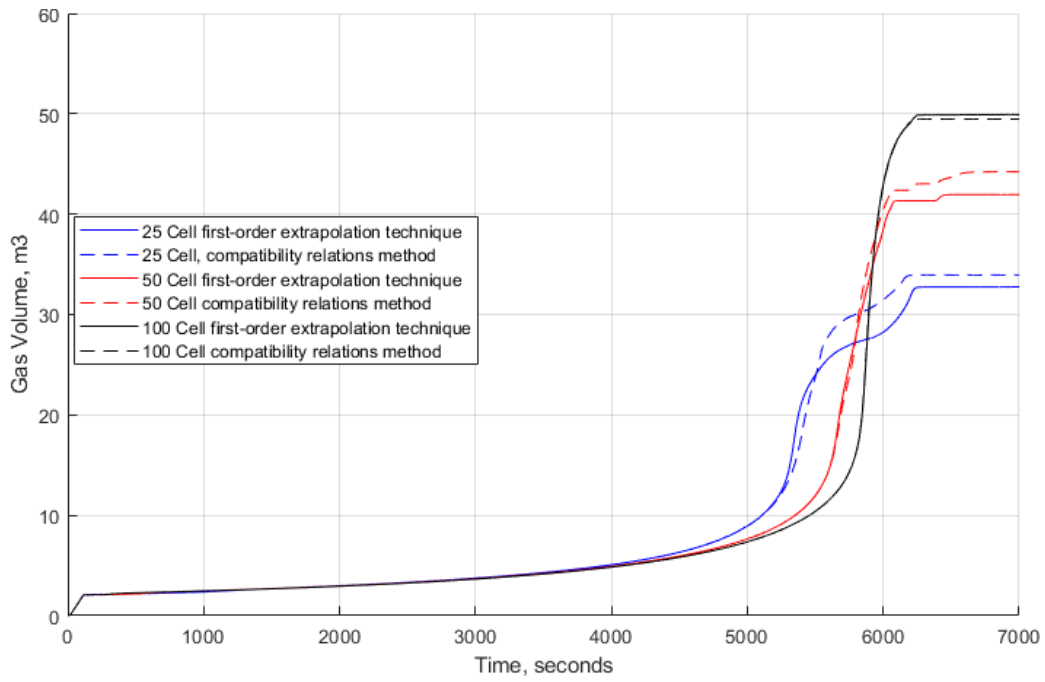


Figure 4.16: Kick volume vs. time for first-order extrapolation technique and compatibility relations method with grid refinement.

In Fig 4.15 and 4.16, the same simulation runs have been performed except by comparing the first-order extrapolation with compatibility relations method. For bottomhole pressure, even with a less refined grid, the compatibility relations method does do a slightly better job than first-order extrapolation method since the final pressure is slightly lower. The inflection point is still present when using the compatibility relations method but is less evident. The trends as seen in bottomhole pressure can also be seen in the gas volume figure. The compatibility relations method does a slightly better job at predicting the gas volume in the well, but it proves that grid refinement is still much more important than which boundary condition treatment that is used.

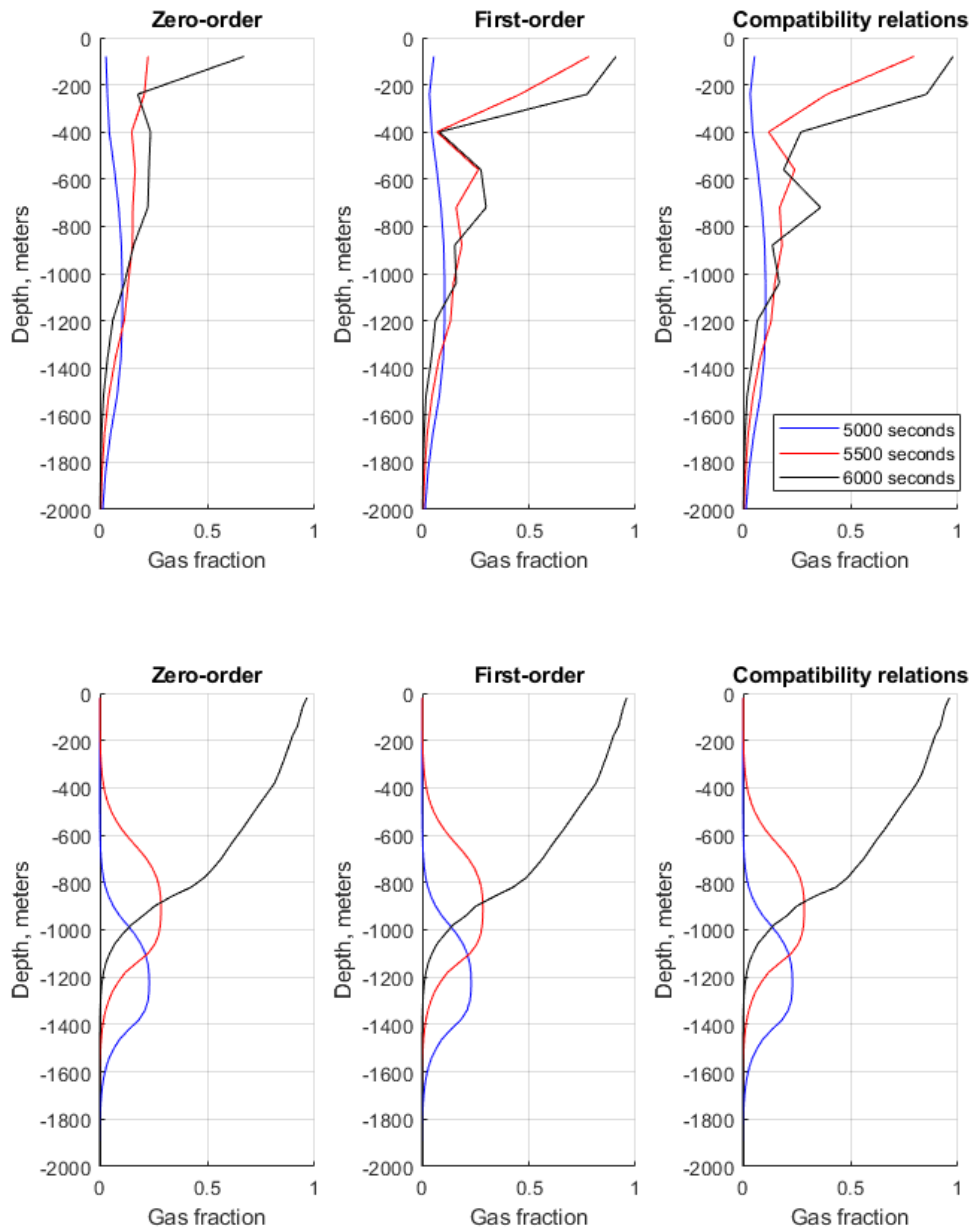


Figure 4.17: Depth vs. gas fraction for 25 cells (top row) and 100 cells (bottom row).

In Fig. 4.17, the depth is plotted against gas fraction for the least refined grid and the most refined grid. From this figure, the improvement in numerical diffusion in the simulation can be spotted quite easily. For the 25 cells grid, the gas volume fraction at outlet start reaching high gas fraction volume at both 5500 and 6000 seconds while for the 100 cells grid, the gas fraction at outlet is first reaching a high volume at 6000 seconds. For 25 cells grid, it is seen that first-

order extrapolation technique and compatibility relations method lead to large gas expansion at surface compared to the use of zero-order extrapolation technique. Remember, that for zero-order extrapolation technique, there will be no gas expansion in the last half of the last cell. This can also be seen in the first row of the figure since it is the mid-point values that are plotted.

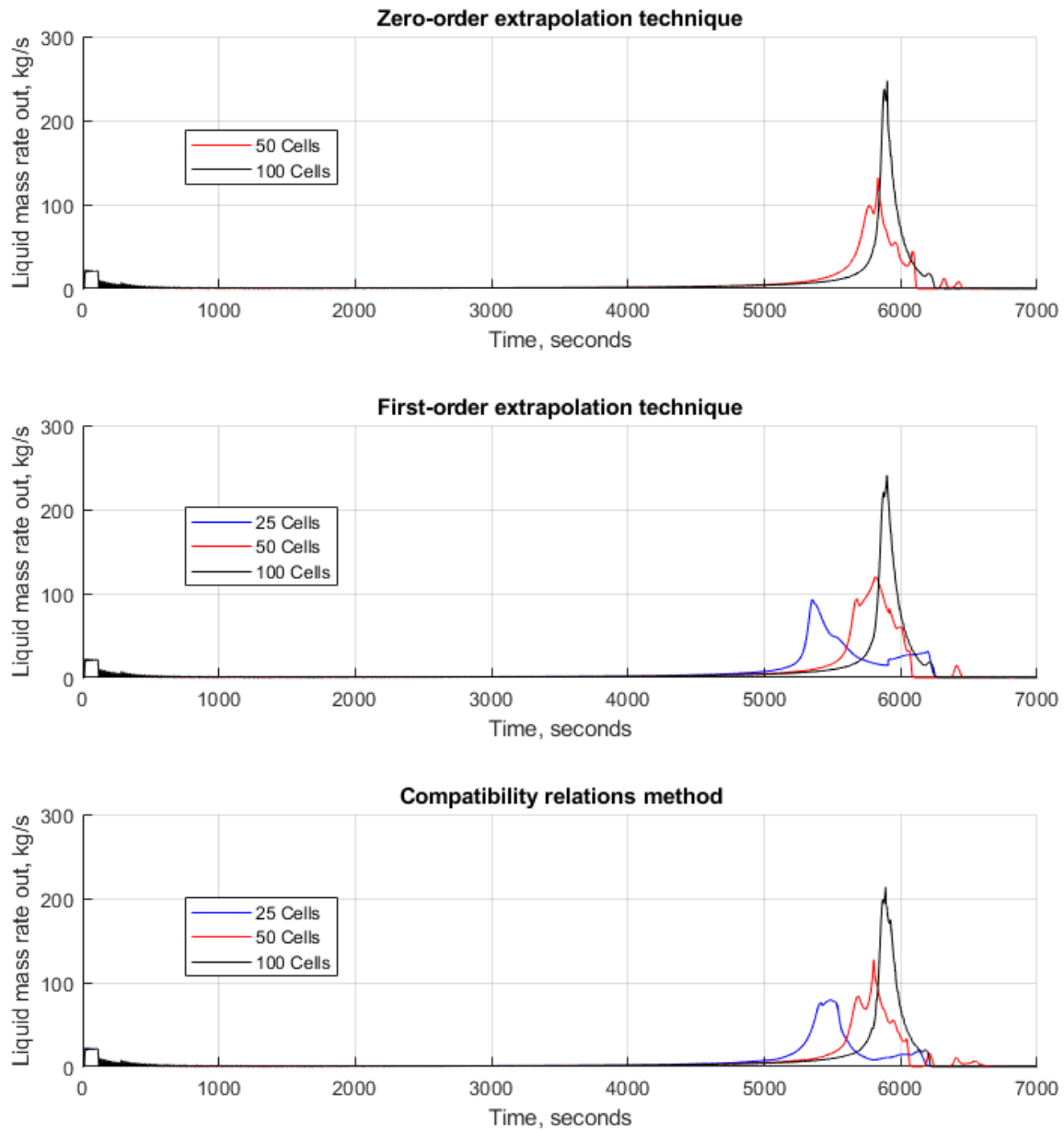


Figure 4.18: Liquid mass rate out vs. time for all methods over grid refinement.

In Fig. 4.18, the liquid mass rate out has been plotted against time with the different boundary condition treatment using different grid refinements. The 25 cells zero-extrapolation technique has been omitted due to its instability. The graph shows that grid refinement is much more dominant than boundary condition treatment. The zero-order extrapolation technique has the highest liquid mass rate out while the compatibility relations method has the lowest peak of the three. The figure also displays the improvements in regarding reduction of numerical diffusion when refining the grid. For the simulation using 25 cells grid, almost all the liquid has been expelled compared to the situation when using 100 cells grid, where the liquid expulsion has just started.

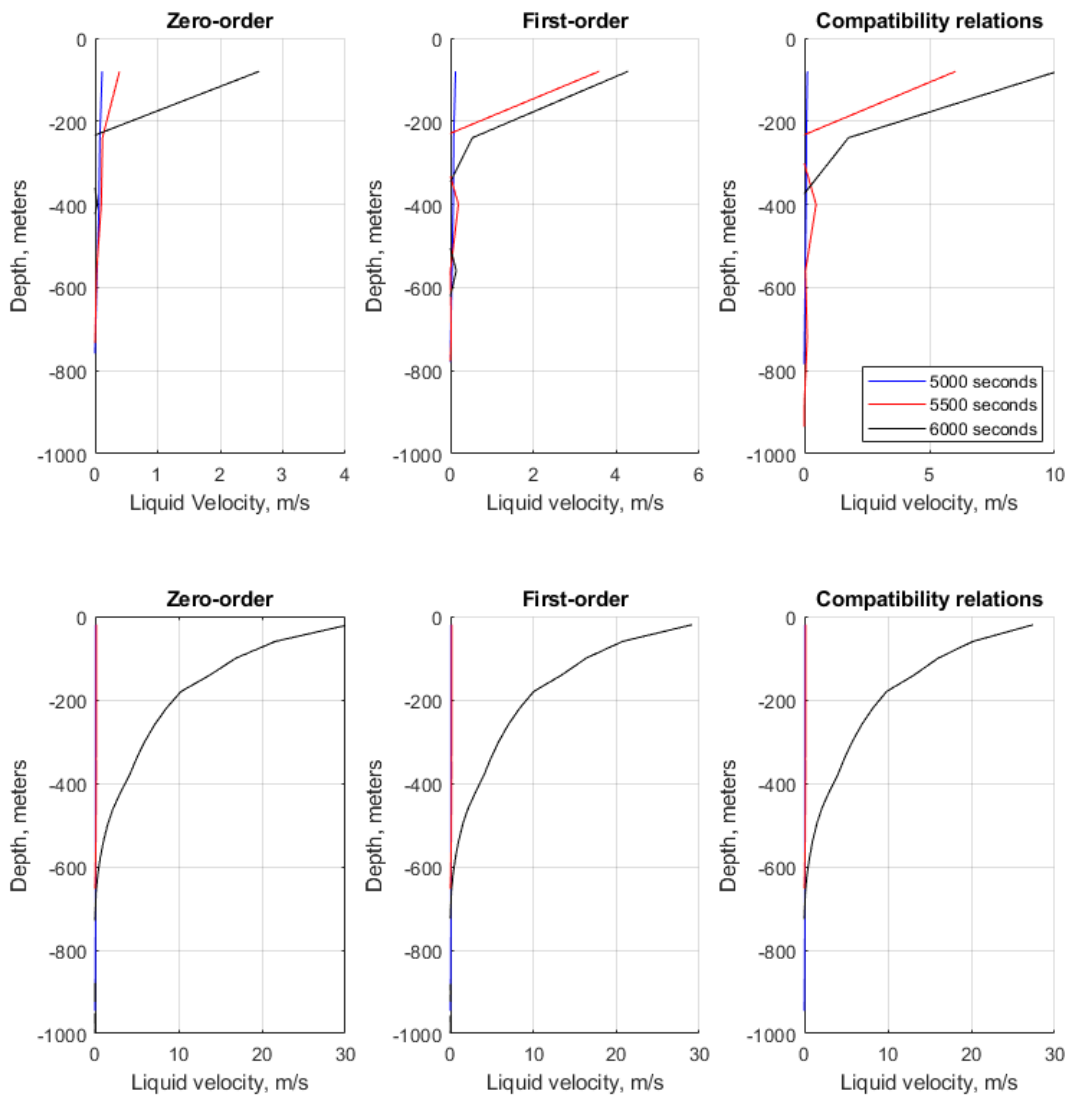


Figure 4.19: Depth vs. liquid velocity for 25 cells (top row) and 100 cells (bottom row) with grid refinement.

In Fig. 4.19, the depth is plotted against the liquid velocity for the least refined grid and the most refined grid. From the lower count cell grid, the compatibility relation predicts a higher liquid velocity at the outlet. However, with grid refinement, the method used for boundary condition treatment does not have that much of an impact. The grid refinement reduces the numerical diffusion significantly. This allows for the gas to be less spread throughout the well and gain a much higher blowout velocity. As seen from the bottom row, the velocity at outlet does spike very high, and the liquid has most likely broken the sound barrier in two-phase flow. For open well status, if a fast, low cell grid is required, compatibility relations method would probably give better simulation results than zero-order and first-order extrapolation techniques.

4.4 Case 4 – Circulate Kick Out in Vertical Well

Case 4 is similar setup to the previous case, but instead of the letting the kick migrate up to the surface by itself, here the kick will be circulated out. Like in case 3, zero-order extrapolation will also be compared to the other boundary condition treatment methods.

The well has a depth of 4000 meter and a geometry of 12.25" x 5" annulus. This gives an outer diameter of 0,331 m and inner diameter of 0,127 m which results in a cross-sectional area of 0,0634 m². The compressibility is such that the related sonic velocity $a_l = 1500 \text{ m/s}$ (water) and the kick influx is assumed to be an ideal gas with a sonic velocity of $a_g = 316 \text{ m/s}$. The kick volume is half of what it is in [6] for the purpose of having case 2, 3 and 4 setup similar. If the kick volume is too large, the gas and liquid velocity at the outlet in an open well status may exceed the sound barrier and could make the simulation unstable.

The initial kick volume used here is 2,05 m³ and is introduced using a gas mass rate of 8 kg/s. The well is first kept static for the first 10 seconds then the gas mass rate is ramped up linearly over a 10 second period. The gas mass rate is then kept static for the next 90 seconds until it is ramped down linearly over a 10 second period. At the same time gas mass rate is ramped down, the liquid mass rate will be linearly ramped up over a 10 second period, up to a maximum of 40 kg/s (2400 lpm). The liquid mass rate injected will continue to the end of the simulation to circulate the gas out of the well. This will force the gas up the well faster and will lead to gas expansion and eventually expulsion from the well. Right before gas reaches surfaces and a little

after, there will be an increasing amount of liquid expelled from the well and then the liquid rate will drop to zero with only gas flowing out.

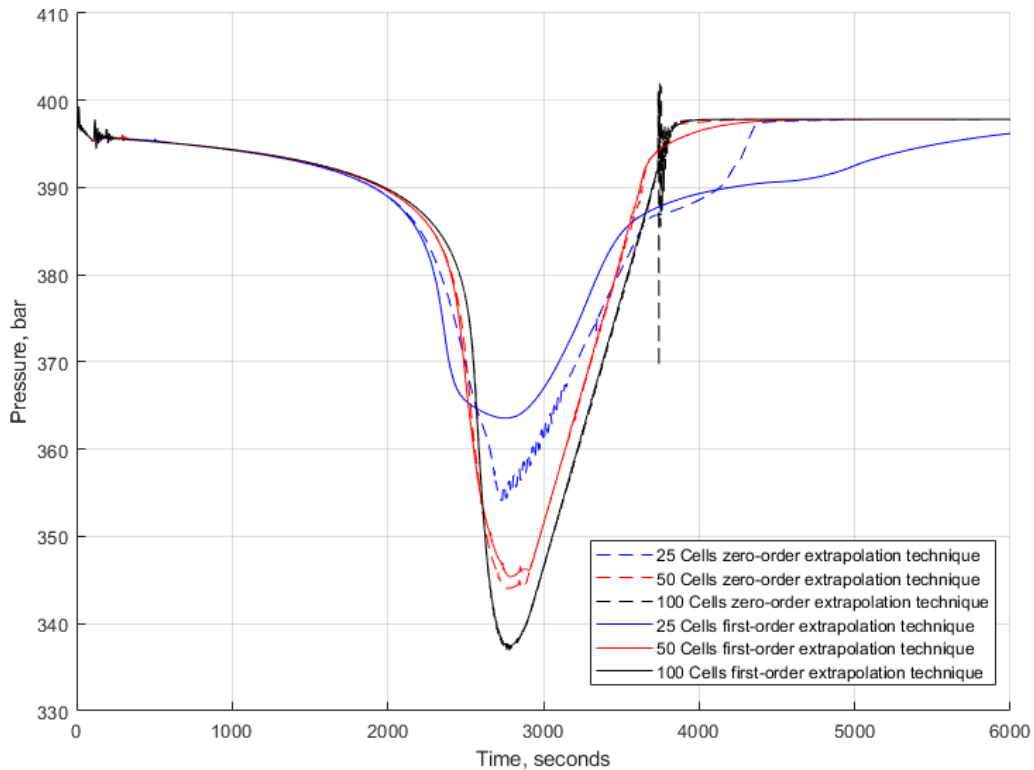


Figure 4.20: Bottomhole pressure vs. time for zero-order and first-order extrapolation technique with grid refinement.

In Fig. 4.20, the bottomhole pressure has been plotted against time for zero-order and first-order extrapolation techniques. There are a few interesting things going on with this setup. The simulation based on 25 cells zero-order extrapolation technique does predict a lower bottomhole pressure than the first-order extrapolation technique between 2600 and 3600 seconds. The lower cell count grids seem to have issues performing the single-phase gas to single-phase liquid transition. The first-order extrapolation technique using 25 cells seems to fail to properly circulate out the gas and refill the well with fluid as it is supposed to. Both zero-order and first-order extrapolation techniques do a decent job for a 50 cells grid, and they do a fairly similar job at predicting the bottomhole pressure. For 100 cells grid, the first-order extrapolation technique does perform better than zero-order as the zero-order runs into some stability issues at around 3700 seconds when the gas has almost left the well. Apart from the

slight instability for zero-order extrapolation technique for 100 cells grid, it does a similar job as the first-order extrapolation technique.

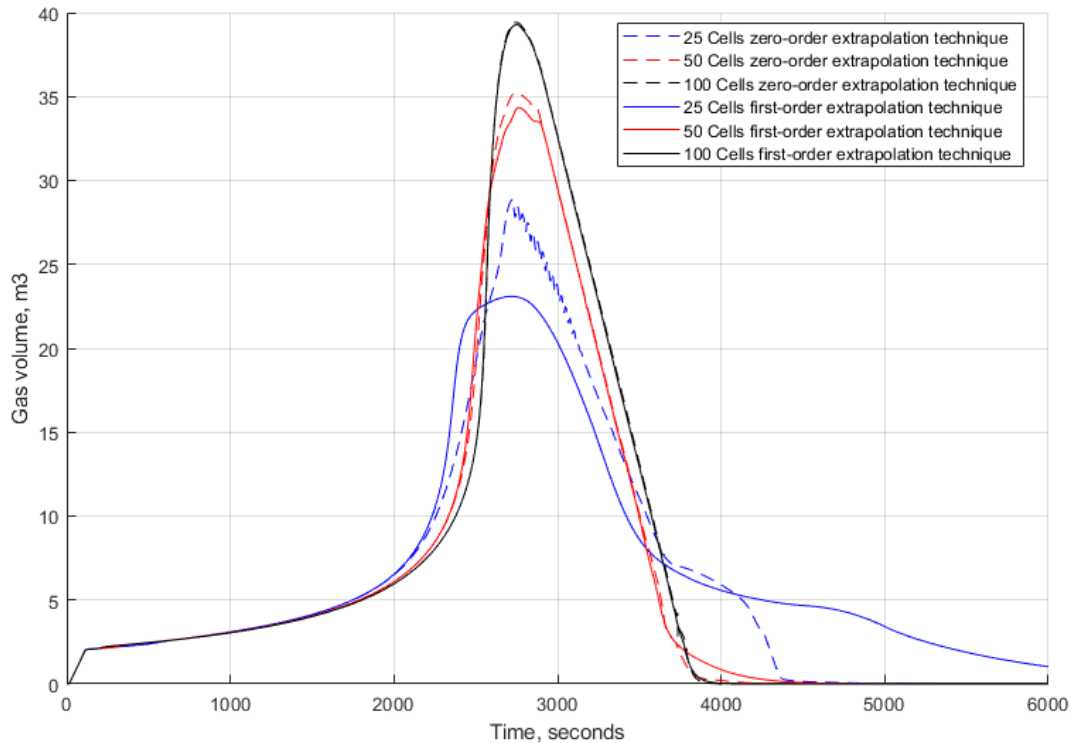


Figure 4.21: Gas volume vs. time for zero-order and first-order extrapolation techniques.

In Fig. 4.21, the kick volume has been plotted against the time for zero-order and first-order extrapolation techniques. For the first-order extrapolation technique using 25 cells, there is still gas left in the well when the simulation has ended. This should have been removed at this stage since we are circulating liquid from below. However, the simulation seems to work well until about 3500 seconds, but it is not able to handle the transition to pure liquid in the well. The gas takes a very long time to be removed from the well. For the zero-order extrapolation technique for 25 cells, we can see the inflection point like in the bottomhole pressure figure. In the simulations using zero-order and first-order extrapolation techniques with 50 cells grid, the maximum gas volume is only 4-5 m³ below the maximum obtained with both methods using a 100 cells grid. The simulation using 50 or 100 cells do a pretty good job in predicting the kick volume in the well vs. time. However, the transition from two-phase flow to one-phase liquid flow seen at around 3800 seconds is much sharper for the 100 cells grid.

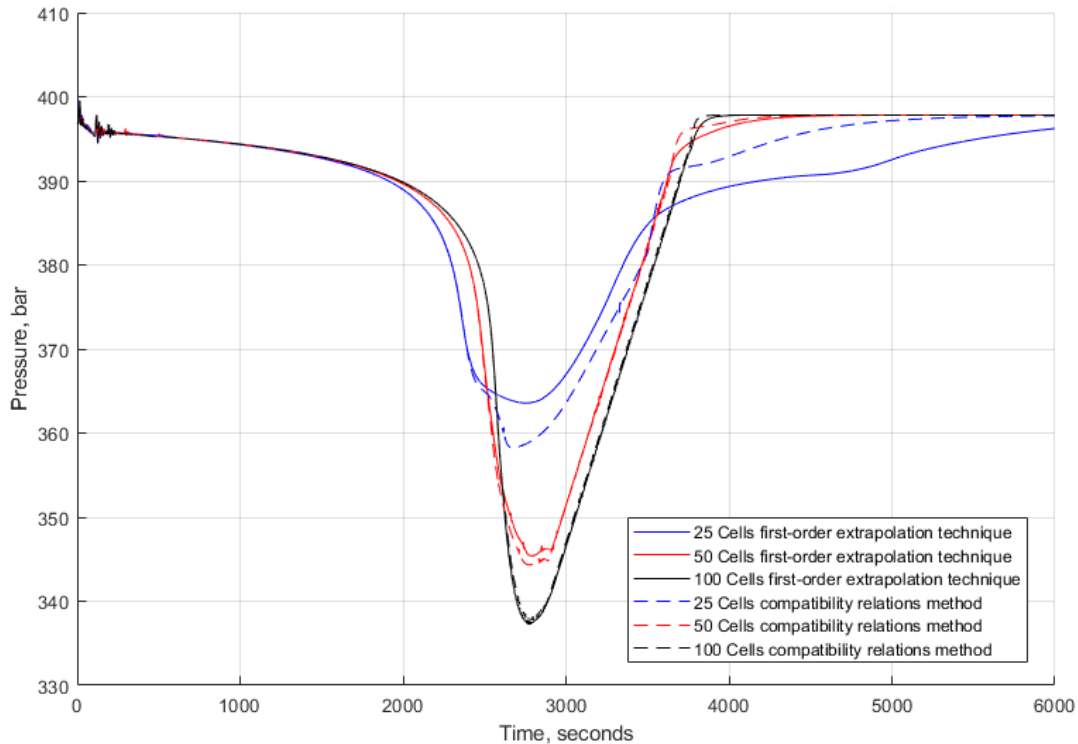


Figure 4.22: Bottomhole pressure vs time for first-order extrapolation technique and compatibility relations method.

In Fig. 4.22, the bottomhole pressure has been plotted against time comparing first-order extrapolation technique and compatibility relations method. For the rougher grids (25 cells), the compatibility relations method does to give lower bottomhole pressure predictions than the extrapolation technique. However, there seems to be an inflection point at around 2600 seconds that may be a bit hard to explain. The more refined grid using 50 and 100 cells in combination with compatibility relations seems to predict the bottomhole pressure very well without any stability issues. The exact same conclusion as drawn earlier may not apply as well to this case, where increasing the cell count in the grid is better no matter what. The reason is that for some of the simulation cases using extrapolation techniques some stability issues occurred. In one case, the gas was also not properly removed. The problems typically occurred when the gas almost had left the well.

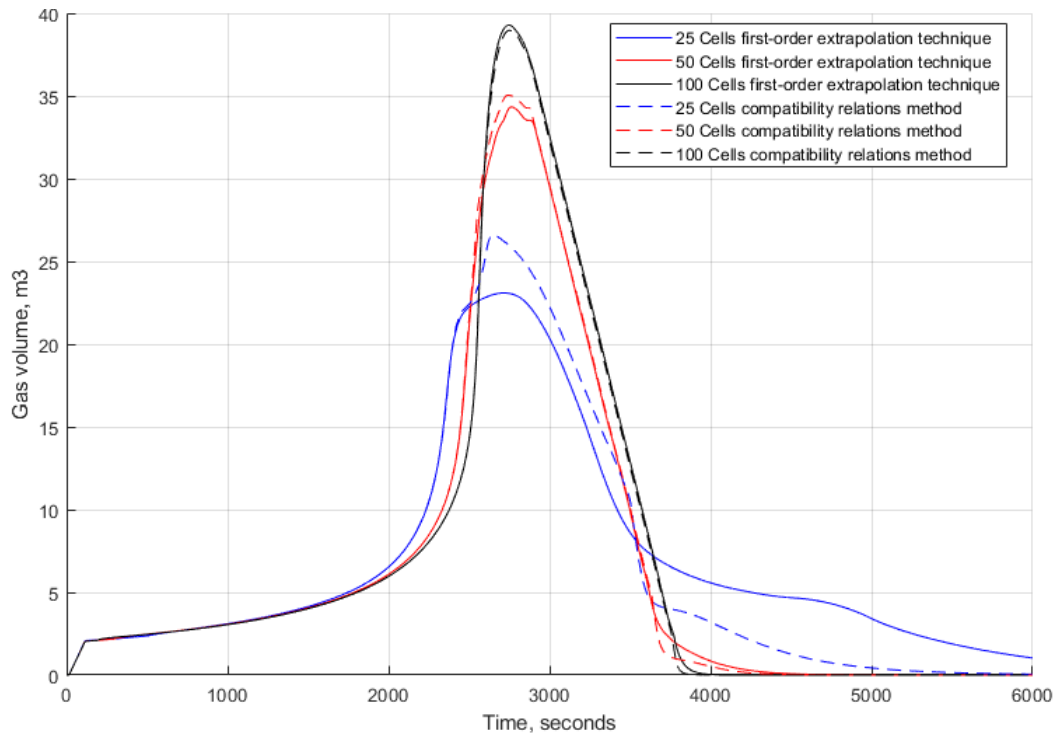


Figure 4.23: Gas volume vs time for first-order extrapolation technique and compatibility relations method.

In Fig. 4.23, the gas volume has been plotted against time comparing the first-order extrapolation technique and compatibility relations method. This is also a good representation of how the compatibility relations method does a better job with this case than the two different extrapolation techniques. The 25 cells grid compatibility relations method predicts a higher gas volume for the peak than the first-order extrapolation technique and here the gas is also completely removed from the well. For the 50 and 100 cell count grids, the two methods do a similar job with a slight advantage for compatibility relations method as it seems to have a sharper transition between the two-phase region and the one-phase liquid region when the final gas leaves the well. However, in this scenario, a higher cell count is more important than which boundary condition treatment method is chosen for the simulation.

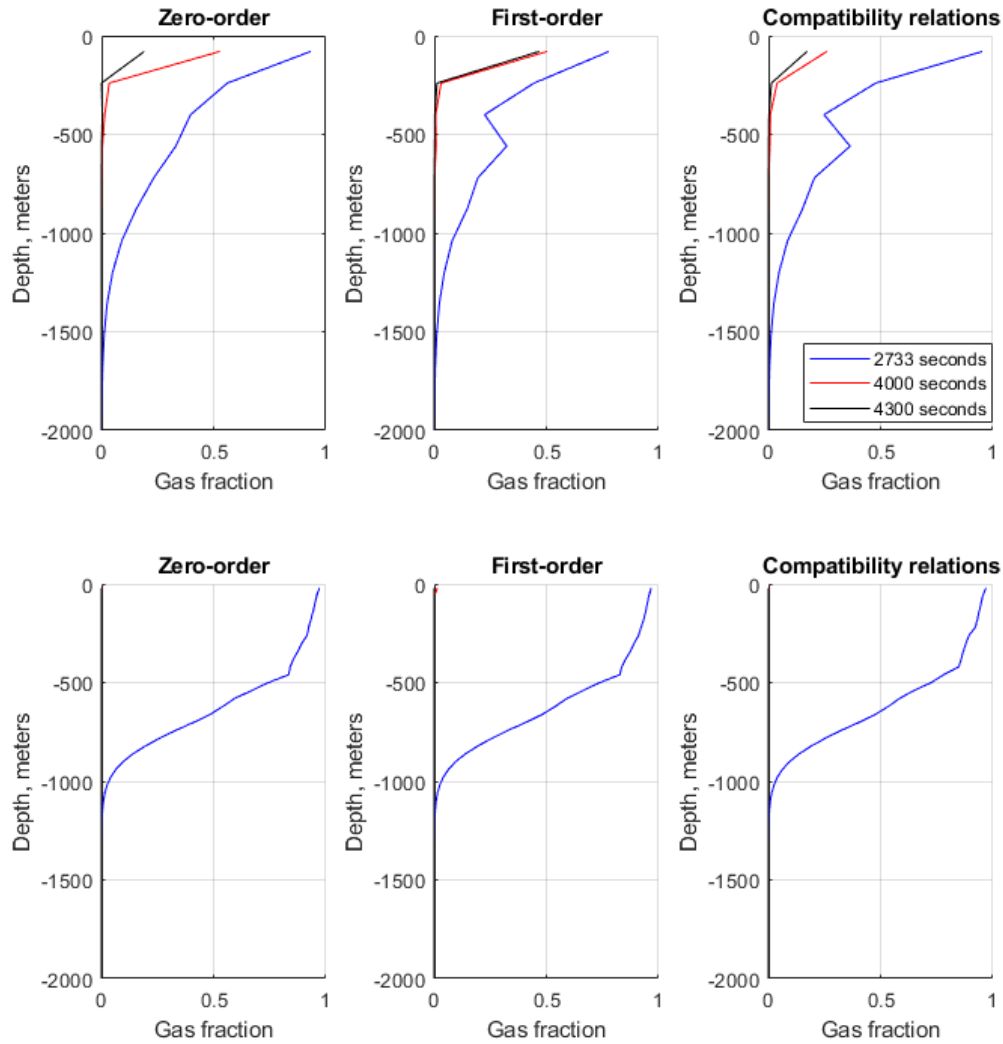


Figure 4.24: Depth vs. gas fraction for 25 cells (top row) and 100 cells (bottom row).

In Fig. 4.24, the depth has been plotted against the gas fraction for the low- and high-resolution grids for all the boundary condition treatment methods. The 2733 seconds point in the simulation has been picked because this is exactly where the bottomhole pressure is the lowest and consequently will be where the gas velocity out of the well will be at its highest. As seen from the lower resolution grids, the numerical diffusion has a significant impact on the results and there will still be gas present in the upper part of the well at both 4000 and 4300 seconds. However, for the simulation using 100 cells, all the gas has left the well. As for the higher resolution grids, one can see that the gas fraction during the peak approaches a gas fraction of 1, which is quite similar for all the different boundary condition treatments. The zero-order

extrapolation method does seem to be predict a more smooth gas fraction profile compared to the two other methods. For simulations using 25 cells grid, there will be some variation in gas volume profile for the different methods. The compatibility relations method predicts a lower gas fraction at the outlet for 4000 and 4300 seconds for the lower resolution grid.

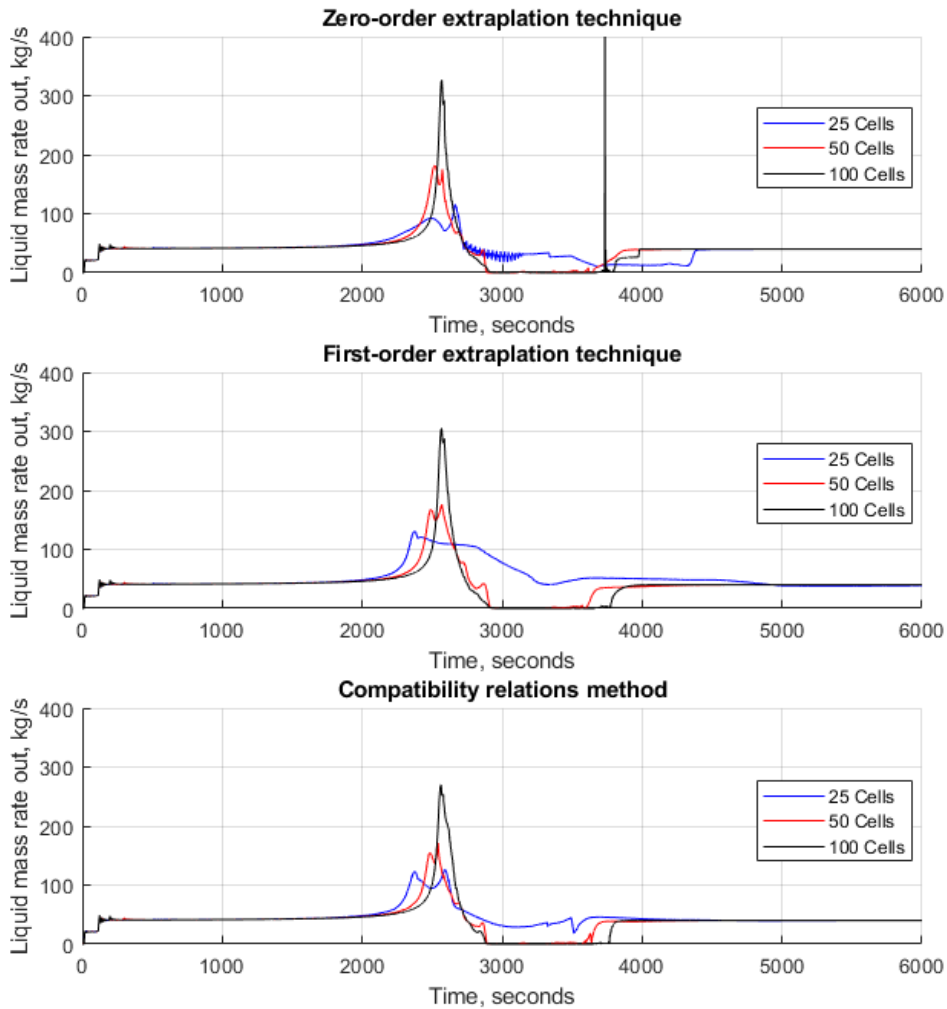


Figure 4.25: Liquid mass rate out vs. time for all boundary condition treatment methods for different grid refinement.

In Fig. 4.25, the liquid mass rate out of the well has been plotted against time for all the different boundary condition treatments with grid refinement. As we can see from the figure, the liquid mass rate out is at a steady 40 kg/s as is expected due to the continuous flow of injected fluid from 120 seconds. After 2000 seconds, a large quantity of liquid will be pushed out in front of the gas and then drop to zero as only gas is exiting the well. Once all the gas has left the well, the liquid rate will increase again to the steady state condition with 40 kg/s with liquid

flow. From this figure, it's certainly possible to confirm that 25 cells grid for this type of simulation is not nearly enough. The reason for that is because the simulation results for 25 cells grids severely underestimate the max mass flowrate at outlet. All of the boundary condition treatments for 50 cells and 100 cells grid are able to achieve single-phase of gas at the outlet when the gas is exiting the well. Maximum liquid rate is higher for a more refined grid but there is a tendency that the compatibility relations method leads to a slightly lower prediction compared to the extrapolation techniques. For zero-order extrapolation technique for 25 cells grid, there seems to be stability issues right after the peak in liquid rate and the liquid rate continues to behave slightly odd. The zero-order extrapolation technique for 100 cells grid predicts the largest maximum flowrate at outlet with first-order extrapolation technique and compatibility relations method following slightly behind. The zero-order extrapolation technique for 100 cells grid also has an issue at around 3600 seconds with a spike in the liquid rate.

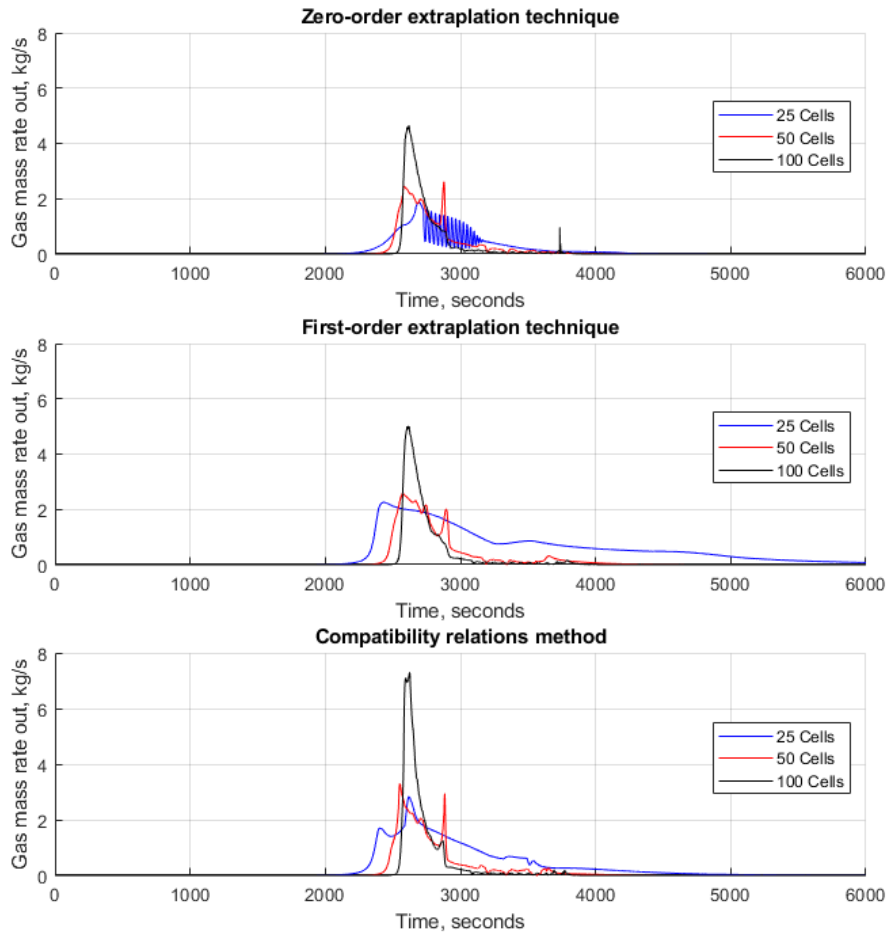


Figure 4.26: Gas mass rate out vs. time for all boundary condition treatment methods over grid refinement.

In Fig. 4.26, the gas mass rate out has been plotted against time comparing all the boundary condition treatment methods against each other. In this figure, the issue with numerical diffusion is shown very well. For a more refined grid, the maximum gas rate occurring increase and the period the gas is flowing out of the well is reduced. For the zero-order extrapolation technique for 25 cells grid, the stability issue is much more pronounced. For 100 cells grid, one can see that the compatibility relation does predict the outlet gas mass rate to be much higher than zero-order and first-order extrapolation techniques. Compatibility relations predict almost a 60% higher gas mass rate out than zero-order extrapolation technique and 45% higher than the first-order extrapolation technique. For the lower resolution grids, the percentages are not as high, but still the gas rate predictions by using the compatibility relations method are higher than the ones obtained with the extrapolation techniques.

still predicts quite a bit over than the extrapolation techniques.

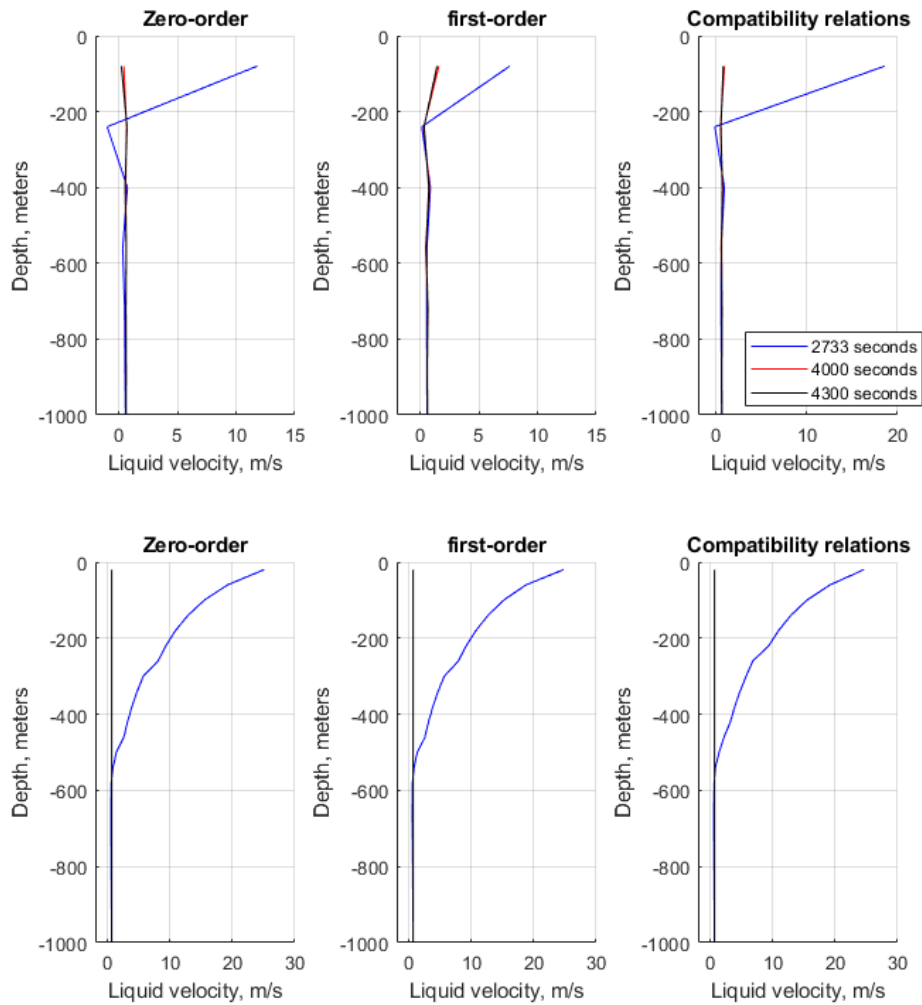


Figure 4.27: Depth vs. liquid velocity for 25 cells (top row) and 100 cells (bottom row) with grid refinement.

In Fig. 4.27, the liquid velocities have been plotted for the three different boundary condition treatment methods for 25 cells and 100 cells grids. The liquid velocity is peaking at 2700 seconds as this is when the gas and liquids are leaving the well. As seen from this figure as well, the compatibility relations method is predicting a higher liquid velocity for a lower resolution grid. The three different boundary condition treatments provide the same velocity profile for a higher resolution grid.

5. Conclusion and Future Work

The cases for this thesis have had generally similar setup but a major key difference between them. The reason for that is because the cases have been built in order to be very rigorous to test the boundary condition treatment methods against one another. One of the goals in this work has been to create a set of benchmark cases for the boundary condition treatment methods to be tested against. The cases consider both pressure pulse propagation as well as mass transport. For two-phase flow, the cases test pure transition zones between phases. In both case 2 and case 3, a pure transition zone between gas and liquid is obtained in the end. In case 2, a closed well is considered and kick migration will lead to redistribution of fluids in the well. In case 3 and case 4, the effect of gas expansion is considered where case 3 is the most extreme situation. So, we have also tested both open and closed end conditions at the outlet for a two-phase flow situation.

Previous work prior to this thesis has generally used zero-order extrapolation technique for boundary condition treatment – especially for open hole wells. As shown in this work, zero-order extrapolation technique may not give the optimal simulation results.

Conclusions:

Case 1:

- Both the compatibility relations method and first-order extrapolation technique are able to handle pressure pulse propagating back and forth in the pipeline.
- For a rough grid, the compatibility relations method seemed to behave better than the first-order extrapolation technique avoiding some spikes in the predicted pressure. Hence, it may be a better approach for modelling water hammer effect.

Case 2:

- Compatibility relations method for 100 cells grid had stability issues for case 2, most likely due to lacking specific single-phase gas compatibility relations method equations in the code.

- Compatibility relations method provided no specific benefit over first-order extrapolation technique for a closed well.

Case 3:

- Zero-order extrapolation technique may run into major stability issues if grid resolution is too low but will generally perform well if the grid resolution is high enough as seen in case 3.
- For 25 cells grid, all boundary condition treatment methods have an inflection point in the bottomhole pressure when the gas starts to leave the well. This is hard to explain and does not seem physical. The compatibility relations method seems to reduce the effect.
- For 50- and 100-cells grids, the first-order extrapolation technique and compatibility relations did perform quite similar. The grid count has a much greater impact than which boundary condition treatment method is used.
- Zero-order and first-order extrapolation technique predicts a higher liquid mass rate out than the compatibility relations method. This difference is also present for a fine grid.

Case 4:

- When using a rough grid of 25 cells, all extrapolation techniques have some problems when the final gas is about to leave the well. The gas will not leave the well as expected.
- Zero-order extrapolation technique encountered stability issues for a finer grid in the two-phase to one-phase liquid transition taking place when the final gas is leaving the well.
- Compatibility relations method seems to perform better than zero-order and first-order extrapolation techniques for a lower resolution grid for predicting bottomhole pressure.
- However, as seen from liquid mass rate out, a 25-cell grid is too rough for this type of simulation. The maximum liquid and gas rates will be highly under predicted in this case.
- Similar to case 3, the zero-order and first-order extrapolation techniques predicts a higher liquid mass rate out than compatibility relations method even for a refined grid.

- However, compatibility relations method predicts a much higher gas mass rate out of the well – achieving up to 60% and 45% more gas mass rate out than zero-order and first-order extrapolation techniques, respectively for a refined grid.

General:

- Generally, increasing the grid resolution has a greater impact than which boundary condition treatment is chosen. The most important is to use a sufficient number of cells in the simulation. This reduces numerical diffusion and there is a tendency of also having more stable simulations. When using a refined grid, the numerical error introduced by using zero-order extrapolation technique is reduced.
- Zero-order extrapolation generally caused more problems numerically when considering two-phase flow and rough grids.
- There is a difference in outlet maximum liquid and gas rates during unloading scenarios when comparing the compatibility relations method with the results from the extrapolation techniques. The tendency was a reduced maximum liquid rate and an increased maximum gas rate. It may be that the compatibility relations method provided the most correct results as the gas slip relation is used directly in the boundary treatment.
- To implement the compatibility relations method is quite complex compared to using the much simpler extrapolation technique.

Future Work

There was an issue with this particular AUSMV scheme that we were not able to solve. For some reason throughout case 2, 3 and 4 there was gas just added to the system where it was not supposed to. This issue has nothing to do with the implementation of compatibility relations method as this issue was present prior to its implementation.

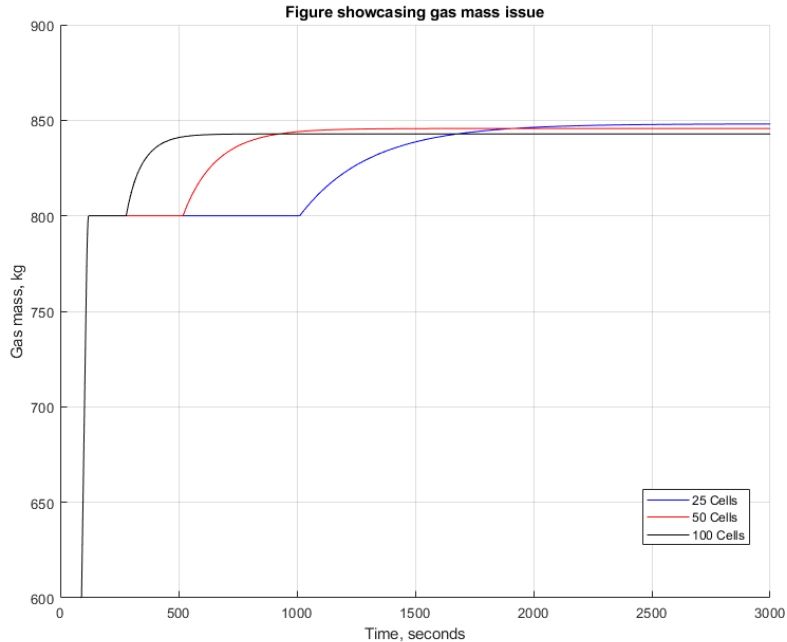


Figure 5.1: Gas mass vs. time showing gas being added to the system unintentional.

In Fig. 5.1, the gas mass has been plotted against time for three different grid resolutions. This type of behavior was similar for all the cases that have been run for this work. One can see that after certain amount of time depending on which grid resolution is chosen, about 45 kg of gas is added in the system. The finer the grid the earlier the gas is added to the system. For a lower grid resolution, the extra gas is added right after 1000 second mark, and for 50 and 100 cells grid the extra gas is added at 500 and 300 seconds, respectively. The actual cause of this is unknown but may have something to do with an if statement in the code that makes sure there cannot be negative amount of gas in a cell. However, when trying to correct this, the code execution time went up significantly and the simulation results were completely off.

The work for this thesis has only used water and ideal gas to base the simulation on. For a better representation for the oil & gas industry, it is vital to adapt other models that capture different flow conditions and behaviors of different fluids.

6. References

1. Evje, S. and Fjelde, K. K. 2002. Hybrid Flux-Splitting Schemes for a Two-Phase Flow Model. *J. Computational Physics* **175**(2) 674-701. <http://dx.doi.org/10.1006/jcph.2001.6962>
2. Fjelde, K. K. and Karlsen, K. H. 2001. High-resolution hybrid primitive-conservative upwind schemes for the drift flux model. *Computer & Fluids*. 31. 335-367.
3. Master Thesis by Kjell Kåre Fjelde: "Numerical methods for simulating a kick." University of Bergen, 1995.
4. Fjelde, K. K. Modelling of Wellflow, PETE510 Compendium.
5. Fjelde, K. K. and Evje, S., Work Note. 2010. The AUSMV Scheme – Simple But Robust Model For Analyzing Two-Phase Flow. University of Stavanger
6. Fjelde, K. K. et al. 2016, April 20. A Numerical Study of Gas Kick Migration Velocities and Uncertainty. Society of Petroleum Engineers. doi:10.2118/180053-MS
7. C, Hirsch. 1990. Numerical computation of internal and external flows, volume 2, chapter 16, 17 and 19. John Wiley and Sons
8. Udegbumam, J. E. et al. (2015, June 1). On the Advection-Upstream-Splitting-Method Hybrid Scheme: A Simple Transient-Flow Model for Managed-Pressure-Drilling and Underbalanced-Drilling Applications. Society of Petroleum Engineers. doi:10.2118/168960-PA
9. Gomes, D. et al. (2018, April 18). A Numerical Comparison and Uncertainty Analysis of Two Transient Models for Kick Management in a Backpressure MPD System. Society of Petroleum Engineers. doi:10.2118/191345-MS
10. Ghauri A. et al. (2016, June 19-24). A Transient model for Hydraulic Simulation of Bullheading and Pressurized Mud Cap Drilling. ASME. International Conference on Offshore Mechanics and Arctic Engineering, Volume 8: Polar and Arctic Science and Technology; Petroleum Technology():V008T11A029. doi:10.1115/OMAE2016-54293.
11. LeVeque, R.J. 1992. *Numerical Methods for Conservation Laws*. Second Edition. Birkhauser Verlag. Basel.
12. Evje, S. and Fjelde, K.K. 2003. On a Rough AUSM Scheme for a One-Dimensional Two-Phase Model. *Computers & Fluids* **32** (10): 1497-1530. [http://dx.doi.org/10.1016/S0045-7930\(02\)00113-5](http://dx.doi.org/10.1016/S0045-7930(02)00113-5).
13. Lorentzen, R.J. and Fjelde, K.K. 2005. Use of Slope-limiter Techniques in Traditional Numerical methods for Multi-phase Flow in Pipelines and Wells. *Int. J. Numer. Meth. Fluids* **48** (7): 723-745. <http://dx.doi.org/10.1002/fld.952>.
14. Yuan, Z. et al (2016, June 1). Mitigating Gas-in-Riser Rapid Unloading for Deepwater-Well Control. Society of Petroleum Engineers. doi:10.2118/185179-PA
15. Rommetveit, R. et al. (2003, January 1). HPHT Well Control; An Integrated Approach. Offshore Technology Conference. doi:10.4043/15322-MS
16. Benzoni-Gavage, S. 1991. Analyse Numerique des Modèles Hydrodynamiques d'Écoulements Diphasique Instationnaires dans les Réseaux de Production Pétrolière. These, ENS Lyon, France.

17. Caussade, B. et al. (1989). Unsteady Phenomena in Horizontal Gas-Liquid Flow in Pipes. Int. Conference on Multi-Phase flow, Nice, France, Cranfield, BHRA.

7. Appendices

Appendix 1

The AUSMV code for case 1

First-order extrapolation technique and compatibility relations method

```
% Transient two-phase code based on AUSMV scheme: Gas and Water
% The code assumes uniform geometry

% time - Seconds

% p - pressure at new time level (Pa)
% dl - density of liquid at new time level (kg/m3)
% dg - density of gas at new time level (kg/m3)
% eg - phase volume fraction of liquid at new time level (0-1)
% ev - phase volume fraction of gas at new time level (0-1)
% vg - phase velocity of gas at new time level (m/s)
% vl - phase velocity of liquid at new time level (m/s)
% qv - conservative variables at new time level ( 3 in each cell)
% temp - temperature in well (K)

% po - pressure at old time level (Pa)
% dlo - density of liquid at old time level (kg/m3)
% dgo - density of gas at new old level (kg/m3)
% ego - phase volume fraction of liquid at old time level (0-1)
% evo - phase volume fraction of gas at old time level (0-1)
% vgo - phase velocity of gas at old time level (m/s)
% vlo - phase velocity of liquid at old time level (m/s)
% qvo - conservative variables at old time level ( 3 in each cell)
% temp - temperature in well (K)

clear;
t = cputime
tic,

% Geometry data/ Must be specified
welldepth = 10000;
nobox = 25; %Number of boxes in the well

% Note that one can use more refined grid, 50, 100 boxes.
% When doing this, remember to reduce time step to keep the CFL number
% fixed below 0.25.. dt < cfl x dx/ speed of sound in water. If boxes are
% doubled, then half the time step.

nofluxes = noibox+1; % Number of cell boundaries
dx = welldepth/nobox; % Boxlength
%dt = 0.005;

% Welldepth. Cell 1 start at bottom
x(1)= -1.0*welldepth+0.5*dx;
for i=1:nobox-1
    x(i+1)=x(i)+ dx;
```

```

end

dt= 0.01*4; % Timestep (seconds)
dtdx = dt/dx;
time = 0.0; % initial time.
endtime = 3; % Time for ending simulation (seconds)
nosteps = endtime/dt; %Number of total timesteps. Used in for loop.
timebetweensavingtimedata = 0.1; % How often in s we save data vs time for
plotting.
nostepsbeforesavingtimedata = timebetweensavingtimedata/dt;

% Slip parameters used in the gas slip relation.  $v_g = K v_{mix} + S$ 
k = 1.2;
s = 0.55;

% Boundary condition at outlet
pbondout=1000000; % Pascal (10 bar)          ATMOSPHERIC PRESSURE MODIFIED

% Initial temperature distribution. (Kelvin)

tempbot = 110+273;
temptop = 50+273;
tempgrad= (tempbot-temptop)/welldepth;
tempo(1)=tempbot-dx/2*tempgrad;
for i = 1:nobox-2
    tempo(i+1)=tempo(i)-dx*tempgrad;
end
tempo(nobox)=tempo(nobox-1)-dx*tempgrad;

temp = tempo;

% Different fluid density parameters
% Note how we switch between different models later.
% These parameters are used when finding the
% primitive variables pressure, densities in an analytical manner.
% Changing parameters here, you must also change parameters inside the
% density routines roliq and rogas.

% Simple Water density model & Ideal Gas. See worknote Extension of AUSMV
% scheme.

rho0=1000; % Water density at STC (Standard Condition) kg/m3
Bbeta=2.2*10^9; % Parameter that depend on the compressibility of water
Alpha=0.000207; % Parameter related to thermal expansion/compression
R = 286.9; % Ideal gas parameter
P0=100000; % Pressure at STC (Pa)
T0=20+273.15; % Temperature at STC (K)

% Very simple models (PET510 compendium)

al = 1500; % Speed of sound in water.
rt= 100000; % Ideal gas parameter in model  $\rho_{og} = p/rt$ 

```

```

rho0=1000; % Water density at STC (Standard Condition) kg/m3
P0=100000; % Pressure at STC (Pa)
T0=20+273.15; % Temperature at STC (K)

% Viscosities (Pa*s)/Used in the frictional pressure loss model (dpfric).
viscl = 0.001; % Liquid phase
viscg = 0.0000182; % Gas phase

% Gravity constant

g = 0; % Gravitational constant m/s2 (g = 0 makes the well horizontal)

% Well opening. opening = 1, fully open well, opening = 0 (<0.01), the well
% is fully closed. This variable will control what boundary conditions that
% will apply at the outlet (both physical and numerical): We must change
% this further below in the code if we want to change status on this.

wellopening = 1.0; % This variable determines if
%the well is closed or not, wellopening = 1.0 -> open. wellopening = 0
%-> Well is closed. This variable affects the boundary treatment.

bullheading = 0.0; % This variable can be set to 1.0 if we want to simulate
% a bullheading operation. But the normal is to set this to zero.

% Specify if the primitive variables shall be found either by
% a numerical or analytical approach. If analytical = 1, analytical
% solution is used. If analytical = 0. The numerical approach is used.
% using the itsolver subroutine where the bisection numerical method
% is used. We use analytical.

analytical = 1;

% Initialization of rest of geometry.
% Here we specify the outer and inner diameter and the flow area
% We assume 8.5 x 5 inch annulus.

for i = 1:nobox

do(i)=0.2;
di(i) = 0; % pipe

```

```

    area(i) = 3.14/4*(do(i)*do(i)- di(i)*di(i));
%   ang(i)=3.14/2;
end

% Initialization of slope limiters. These are used for
% reducing numerical diffusion and will be calculated for each timestep.
% They make the numerical scheme second order.
for i = 1:nobox
    sl1(i)=0;
    sl2(i)=0;
    sl3(i)=0;
    sl4(i)=0;
    sl5(i)=0;
    sl6(i)=0;
end

% Now comes the intialization of the physical variables in the well.
% First primitive variables, then the conservative ones.

% Below we intialize pressure and fluid densities. We start from top of
% the well and calculated downwards. The calculation is done twice with
% updated values to get better approximation. Only hydrostatic
% considerations since we start with a static well.

for i = 1:nobox
    eg(i)=0.0; % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
end

p(nobox)= pbondout+0.5*g*dx*(ev(nobox)*rho0+eg(nobox)*1); % Pressure
dl(nobox)=rholiq(p(nobox),tempo(nobox)); % Liquid density
dg(nobox)=rogas(p(nobox),tempo(nobox)); % Gas density

for i=nobox-1:-1:1
p(i)=p(i+1)+dx*g*(ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
dl(i)=rholiq(p(i),tempo(i));
dg(i)=rogas(p(i),tempo(i));
end

```



```

for i=nobox-1:-1:1
    rhoavg1= (ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    rhoavg2= (ev(i)*dl(i)+eg(i)*dg(i));
    p(i)=p(i+1)+dx*g*(rhoavg1+rhoavg2)*0.5;
    dl(i)=rholiq(p(i),tempo(i));
    dg(i)=rogas(p(i),tempo(i));

end

% Intitalize phase velocities, volume fractions, conservative variables
% and friction and hydrostatic gradients.
% The basic assumption is static fluid, one phase liquid.

for i = 1:nobox
    vl(i)=0; % Liquid velocity new time level.
    vg(i)=0; % Gas velocity at new time level
    eg(i)=0.0; % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
    qv(i,1)=dl(i)*ev(i)*area(i);
    qv(i,2)=dg(i)*eg(i)*area(i);
    qv(i,3)=(dl(i)*ev(i)*vl(i)+dg(i)*eg(i)*vg(i))*area(i);
    fricgrad(i)=0;
    hydgrad(i)=g*(dl(i)*ev(i)+eg(i)*dg(i));
end

% Section where we also initialize values at old time level

for i=1:nobox
    dlo(i)=dl(i);
    dgo(i)=dg(i);
    po(i)=p(i);
    ego(i)=eg(i);
    evo(i)=ev(i);
    vlo(i)=vl(i);
    vgo(i)=vg(i);
    qvo(i,1)=qv(i,1);
    qvo(i,2)=qv(i,2);
    qvo(i,3)=qv(i,3);
end

dlandinold = dlo(1); %Initializing for compatibility relation,
prandinold = po(1); %For inlet
vlandinold = vlo(1);
dlandinnew = dlandinold;
prandinnew = prandinold;
vlandinnew = vlandinold;

dgrandinold = dgo(1);
vgrandinold = vgo(1);
egrandinold = ego(1);
dgrandinnew = dgrandinold;

```

```

vgrandinnew = vgrandinold;
egrandinnew = egrandinold;

dlandoutold = dlo(nobox);           %For outlet
prandoutold = po(nobox);
vlandoutold = vlo(nobox);
egrandoutold = ego(nobox);
dgrandoutold = dgo(nobox);
vgrandoutold = vgo(nobox);

dlandoutnew = dlandoutold;
prandoutnew = prandoutold;
vlandoutnew = vlandoutold;
egrandoutnew = egrandoutold;
dgrandoutnew = dgrandoutold;
vgrandoutnew = vgrandoutold;

% Intialize fluxes between the cells/boxes

for i = 1:nofluxes
    for j =1:3
        flc(i,j)=0.0; % Flux of liquid over box boundary
        fgc(i,j)=0.0; % Flux of gas over box boundary
        fp(i,j)= 0.0; % Pressure flux over box boundary
    end
end

% Main program. Here we will progress in time. First som intializations
% and definitions to take out results. The for loop below runs until the
% simulation is finished.

countsteps = 0;
counter=0;
printcounter = 1;
pin(printcounter) = (p(1)+dx*0.5*hydgrad(1))/100000; % Pressure at bottom for
time storage
pout(printcounter)= pbondout/100000;
pnobox(printcounter)= p(nobox)/100000;
liquidmassrateout(printcounter) = 0;
gasmassrateout(printcounter)=0;
tempbott(printcounter)=tempo(1)-273;
timeplot(printcounter)=time; % Array for time and plotting of variables vs
time
pitvolume=0;
pitrates =0;
pitgain(printcounter)=0;

kickvolume=0;
bullvolume=0;

% The temperature is not updated but kept fixed according to the
% initialization.

```

```

for i = 1:nosteps
    countsteps=countsteps+1;
    counter=counter+1;
    time = time+dt; % Step one timestep and update time.

% Then a section where specify the boundary conditions.
% Here we specify the inlet rates of the different phases at the
% bottom of the pipe in kg/s. We interpolate to make things smooth.
% It is also possible to change the outlet boundary status of the well
% here. First we specify rates at the bottom and the pressure at the outlet
% in case we have an open well. This is a place where we can change the
% code to control simulations. If the well shall be close, wellopening must
% be set to 0.

% In the example below, we take a gas kick and then circulate this
% out of the well without closing the well. (how you not should perform
% well control)

XX = 0; % Gasrate in kg/s

YY= 16.7; % Liquidrate in kg/s

if (time < 1)

    inletligmassrate=0.0;
    inletgasmassrate=0.0;

% elseif ((time>=1) & (time < 5))
%     inletligmassrate = YY*(time-10)/10; % Interpolate the rate from 0 to
%     value wanted.
%     inletgasmassrate = XX*(time-10)/10;
%
% elseif ((time >=5) && (time<10))
%     inletligmassrate = YY;
%     inletgasmassrate = XX;
elseif ((time >=1) & (time<1.5))
%     inletligmassrate = YY-YY*(time-200)/10;
%     inletligmassrate = YY;
    inletligmassrate = YY*(time-1)/0.5;
elseif (time > 1.5)
    inletligmassrate=YY;
    inletgasmassrate=0;
end

kickvolume = kickvolume+inletgasmassrate/dgo(1)*dt;

```

```

% specify the outlet pressure /Physical. Here we have given the pressure as
% constant. It would be possible to adjust it during openwell conditions
% either by giving the wanted pressure directly (in the command lines
% above) or by finding it indirectly through a chokemodel where the
wellopening
% would be an input parameter. The wellopening variable would equally had
% to be adjusted inside the command line structure given right above.

pressureoutlet = pbondout;

% Based on these boundary values combined with use of extrapolations
techniques
% for the remaining unknowns at the boundaries, we will define the mass and
% momentum fluxes at the boundaries (inlet and outlet of pipe).

% inlet/bottom fluxes first.
  if (bullheading<=0)
    % Here we pump from bottom

      flc(1,1)= inletligmassrate/area(1);
      flc(1,2)= 0.0;
      flc(1,3)= flc(1,1)*vlo(1);

      fgc(1,1)= 0.0;
      fgc(1,2)= inletgasmassrate/area(1);
      fgc(1,3)= fgc(1,2)*vgo(1);

      fp(1,1)= 0.0;
      fp(1,2)= 0.0;

      vrandinnew = inletligmassrate / (dlandinnew * area(1));

      a = (vlo(1) - al) * (po(1) - prandinold) / ( dx / 2 );
      b = dlo(1) * al;
      c = (vrandinnew - vrandinold) / ( dt );
      d = (vlo(1) - al) * (vlo(1) - vrandinold) / ( dx / 2 );
%     e = g * dlo(1) * al;
      e = al * (hydgrad(1) + fricgrad(1));

      prandinnew = prandinold + dt * ( e - a + b * ( c + d ));

      dlandinnew = rholiq(prandinnew);

% Old way of treating the boundary
%   fp(1,3)= po(1)+0.5*(po(1)-po(2)); %Interpolation used to find the
% pressure at the inlet/bottom of the well.

```

```

% New way of treating the boundary
    fp(1,3)= po(1)...
            +0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g...
            +0.5*dx*fricgrad(1);

%compability relation
%     fp(1,3) = prandinnew;

else
    % Here we pump from the top. All masses are assumed to flow out of the
    % well into the formation. We use first order extrapolation.
    flc(1,1)=dlo(1)*evo(1)*vlo(1);
    flc(1,2)=0.0;
    flc(1,3)=flc(1,1)*vlo(1);

    fgc(1,1)=0.0;
    fgc(1,2)=dgo(1)*ego(1)*vgo(1);
    fgc(1,3)=fgc(1,2)*vgo(1);

    fp(1,1)=0.0;
    fp(1,2)=0.0;
    fp(1,3)=20000000; % This was a fixed pressure set at bottom when
bullheading
end

% Outlet fluxes (open & closed conditions)

    if (wellopening>0.01)

% Here open end condtions are given. We distinguish between bullheading
% & normal circulation.

        if (bullheading<=0)

            % Here the is normal ciruclation and open well)

                aa = (prandoutnew - prandoutold) / dt;
                bb = vlo(nobox) + al;
                cc = (prandoutold - po(nobox)) / (dx/2);
                dd = dlo(nobox) * al;
                ff = (vlrandoutold - vlo(nobox)) / (dx/2);
                gg = g*dlo(nobox)*al;
                %
                gg = al * (hydgrad(nobox) + fricgrad(nobox));

```

```

vrandoutnew = vrandoutold - dt/dd * (gg+aa+bb*cc+dd*bb*ff);
%equation 6.12

egrandoutnew = 0;
egrandoutnew = ego(nobox);

vgrandoutnew = (k*(1-egrandoutnew)*vrandoutnew+s)/(1-
k*egrandoutnew); %equation 6.17
dlandoutnew = rholiq(prandoutnew);
dgrandoutnew = rogas(prandoutnew);

%      Extrapolation
%      flc(nofluxes,1)= dlo(nobox)*evo(nobox)*vlo(nobox);
%      flc(nofluxes,2)= 0.0;
%      flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);
%
%      fgc(nofluxes,1)= 0.0;
%      fgc(nofluxes,2)= dgo(nobox)*ego(nobox)*vgo(nobox);
%      %      fgc(nofluxes,2)=0; Activate if gas is sucked in!
%      fgc(nofluxes,3)= fgc(nofluxes,2)*vgo(nobox);
%
%      fp(nofluxes,1)= 0.0;
%      fp(nofluxes,2)= 0.0;
%      fp(nofluxes,3)= pressureoutlet;

evvv = evo(nobox)+0.5*(evo(nobox)-evo(nobox-1));
vvvv = vlo(nobox)+0.5*(vlo(nobox)-vlo(nobox-1));
dlll = dlo(nobox)+0.5*(dlo(nobox)-dlo(nobox-1));

%      flc(nofluxes,1)= rholiq(100000,293.15)*evvv*vvvv;
%      flc(nofluxes,1)= dlll*evvv*vvvv;
%      flc(nofluxes,2)= 0.0;
%      flc(nofluxes,3)= flc(nofluxes,1)*vvvv;

gvvv = 1-evvv;
dggg = dgo(nobox)+0.5*(dgo(nobox)-dgo(nobox-1));
vgvv = vgo(nobox)+0.5*(vgo(nobox)-vgo(nobox-1));

%      fgc(nofluxes,1)= 0.0;
%      fgc(nofluxes,2)= rogas(100000,293.15)*gvvv*vgvv;
%      fgc(nofluxes,2)= dggg*gvvv*vgvv;
%      fgc(nofluxes,3)= fgc(nofluxes,2)*vgvv;

fp(nofluxes,1)= 0.0;
fp(nofluxes,2)= 0.0;
fp(nofluxes,3)= pressureoutlet;

%      Compatibility Relations
%      flc(nofluxes,1) = dlandoutnew*(1-egrandoutnew)*vrandoutnew;

```

```

%         flc(nofluxes,2) = 0.0;
%         flc(nofluxes,3) = flc(nofluxes,1)*vlrandoutold;
%
%         fgc(nofluxes, 1) = 0.0;
%         fgc(nofluxes, 2) = dgrandoutnew*egrandoutnew*vgrandoutnew;
%         fgc(nofluxes, 3) = fgc(nofluxes, 2)*vgrandoutnew;
%
%         fp(nofluxes,1)= 0.0;
%         fp(nofluxes,2)= 0.0;
%         fp(nofluxes,3)= pressureoutlet;
else
    % Here we are bullheading.
    flc(nofluxes,1)= inletligmassrate/area(nobox);
    flc(nofluxes,2)= 0.0;
    flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);

    fgc(nofluxes,1)=0.0;
    fgc(nofluxes,2)=0.0;
    fgc(nofluxes,3)=0.0;

    fp(nofluxes,1)=0.0;
    fp(nofluxes,2)=0.0;
    fp(nofluxes,3)= po(nobox)...
    -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g...
    +0.5*dx*fricgrad(nobox); %check sign here on friction
end
else
% Here closed end conditions are given

    flc(nofluxes,1)= 0.0;
    flc(nofluxes,2)= 0.0;
    flc(nofluxes,3)= 0.0;

    fgc(nofluxes,1)= 0.0;
    fgc(nofluxes,2)= 0.0;
    fgc(nofluxes,3)= 0.0;

    fp(nofluxes,1)=0.0;
    fp(nofluxes,2)=0.0;

%     Old way of treating the boundary
%     fp(nofluxes,3)= po(nobox)-0.5*(po(nobox-1)-po(nobox));

%     New way of treating the boundary
    fp(nofluxes,3)= po(nobox)...
    -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g;
%     -0.5*dx*fricgrad(nobox); % Neglect friction since well is closed.
end

% Implementation of slopelimiters. They are applied on the physical
% variables like phase densities, phase velocities and pressure.

```

```

% It was found that if the slopelimiters were set to zero in
% the boundary cells, the pressure in these became wrong. E.g. the upper
% cell get an interior pressure that is higher than it should be e.g. when
% being static (hydrostatic pressure was too high). The problem was reduced
% by copying the slopelimiters from the interior cells. However, both
% approaches seems to give the same BHP pressure vs time but the latter
% approach give a more correct pressure vs depth profile. It is also better
% to use when simulating pressure build up where the upper cell pressure
% must be monitored. It should be checked more in detail before concluding.

```

```

for i=2:nobox-1
    s11(i)=minmod(dlo(i-1),dlo(i),dlo(i+1),dx);
    s12(i)=minmod(po(i-1),po(i),po(i+1),dx);
    s13(i)=minmod(vlo(i-1),vlo(i),vlo(i+1),dx);
    s14(i)=minmod(vgo(i-1),vgo(i),vgo(i+1),dx);
    s15(i)=minmod(ego(i-1),ego(i),ego(i+1),dx);
    s16(i)=minmod(dgo(i-1),dgo(i),dgo(i+1),dx);
end

```

```

% Slopelimiters in outlet boundary cell are set to zero!
% %     s11(nobox)=0;
% %     s12(nobox)=0;
% %     s13(nobox)=0;
% %     s14(nobox)=0;
% %     s15(nobox)=0;
% %     s16(nobox)=0;

```

```

% Slopelimiters in outlet boundary cell are copied from neighbour cell!
s11(nobox)=s11(nobox-1);
s12(nobox)=s12(nobox-1);
s13(nobox)=s13(nobox-1);
s14(nobox)=s14(nobox-1);
s15(nobox)=s15(nobox-1);
s16(nobox)=s16(nobox-1);

```

```

% Slopelimiters in inlet boundary cell are set to zero!
% %     s11(1)=0;
% %     s12(1)=0;
% %     s13(1)=0;
% %     s14(1)=0;
% %     s15(1)=0;
% %     s16(1)=0;

```

```

% Slopelimiters in inlet boundary cell are copied from neighbour cell!
s11(1)=s11(2);
s12(1)=s12(2);
s13(1)=s13(2);
s14(1)=s14(2);
s15(1)=s15(2);
s16(1)=s16(2);

```

```

% Now we will find the fluxes between the different cells.

```



```

% NB - IMPORTANT - Note that if we change the compressibilities/sound
velocities of
% the fluids involved, we may need to do changes inside the csound function.
% But the effect of this is unclear.

```

```

for j = 2:nofluxes-1

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First order method is from here: If you want to test this, activate this
% and comment the second order code below.

```

```

%   cl = csound(ego(j-1),po(j-1),dlo(j-1),k);
%   cr = csound(ego(j),po(j),dlo(j),k);
%   c = max(cl,cr);
%   pll = psip(vlo(j-1),c,evo(j));
%   plr = psim(vlo(j),c,evo(j-1));
%   pgl = psip(vgo(j-1),c,ego(j));
%   pgr = psim(vgo(j),c,ego(j-1));
%   vmixr = vlo(j)*evo(j)+vgo(j)*ego(j);
%   vmixl = vlo(j-1)*evo(j-1)+vgo(j-1)*ego(j-1);

```

```

%
%   pl = pp(vmixl,c);
%   pr = pm(vmixr,c);
%   mll= evo(j-1)*dlo(j-1);
%   mlr= evo(j)*dlo(j);
%   mgl= ego(j-1)*dgo(j-1);
%   mgr= ego(j)*dgo(j);

```

```

%
%   flc(j,1)= mll*pll+mrl*plr;
%   flc(j,2)= 0.0;
%   flc(j,3)= mll*pll*vlo(j-1)+mrl*plr*vlo(j);

```

```

%
%   fgc(j,1)=0.0;
%   fgc(j,2)= mgl*pgl+mgr*pgr;
%   fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);

```

```

%
%   fp(j,1)= 0.0;
%   fp(j,2)= 0.0;
%   fp(j,3)= pl*po(j-1)+pr*po(j);

```

```

% First order methods ends here

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% Second order method starts here:
% Here sloplimiter is used on all variables except phase velocities

```

```

psll = po(j-1)+dx/2*s12(j-1);
pslr = po(j)-dx/2*s12(j);
dsll = dlo(j-1)+dx/2*s11(j-1);

```

```

dslr = dlo(j)-dx/2*s11(j);
dgl1 = dgo(j-1)+dx/2*s16(j-1);
dglr = dgo(j)-dx/2*s16(j);

vlv = vlo(j-1)+dx/2*s13(j-1);
vlh = vlo(j)-dx/2*s13(j);
vgv = vgo(j-1)+dx/2*s14(j-1);
vgh = vgo(j)-dx/2*s14(j);

gvv = ego(j-1)+dx/2*s15(j-1);
gvh = ego(j)-dx/2*s15(j);
lvv = 1-gvv;
lvh = 1-gvh;

cl = csound(gvv,ps11,ds11,k);
cr = csound(gvh,pslr,dslr,k);
c = max(cl,cr);

pll = psip(vlo(j-1),c,lvh);
plr = psim(vlo(j),c,lvv);
pgl = psip(vgo(j-1),c,gvh);
pgr = psim(vgo(j),c,gvv);
vmixr = vlo(j)*lvh+vgo(j)*gvh;
vmixl = vlo(j-1)*lvv+vgo(j-1)*gvv;

pl = pp(vmixl,c);
pr = pm(vmixr,c);

m11= lvv*ds11;
m1r= lvh*dslr;
mgl= gvv*dgl1;
mgr= gvh*dglr;

flc(j,1)= m11*pll+m1r*plr;
flc(j,2)= 0.0;
flc(j,3)= m11*pll*vlo(j-1)+m1r*plr*vlo(j);

fgc(j,1)=0.0;
fgc(j,2)= mgl*pgl+mgr*pgr;
fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);

fp(j,1)= 0.0;
fp(j,2)= 0.0;
fp(j,3)= pl*ps11+pr*pslr;

```

```

%%% Second order method ends here

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% Here sloplimiters is used on all variables. This
% has not worked so well yet. Therefore it is commented away.

```

```

%      psll = po(j-1)+dx/2*s12(j-1);
%      pslr = po(j)-dx/2*s12(j);
%      dsll = dlo(j-1)+dx/2*s11(j-1);
%      dslr = dlo(j)-dx/2*s11(j);
%      dgll = dgo(j-1)+dx/2*s16(j-1);
%      dglr = dgo(j)-dx/2*s16(j);
%
%      vlv = vlo(j-1)+dx/2*s13(j-1);
%      vlh = vlo(j)-dx/2*s13(j);
%      vgv = vgo(j-1)+dx/2*s14(j-1);
%      vgh = vgo(j)-dx/2*s14(j);
%
%      gvv = ego(j-1)+dx/2*s15(j-1);
%      gvh = ego(j)-dx/2*s15(j);
%      lvv = 1-gvv;
%      lvh = 1-gvh;
%
%      cl = csound(gvv,psll,dsll,k);
%      cr = csound(gvh,pslr,dslr,k);
%      c = max(cl,cr);
%
%      pll = psip(vlv,c,lvh);
%      plr = psim(vlh,c,lvv);
%      pgl = psip(vgv,c,gvh);
%      pgr = psim(vgh,c,gvv);
%      vmixr = vlh*lvh+vgh*gvh;
%      vmixl = vlv*lvv+vgv*gvv;
%
%      pl = pp(vmixl,c);
%      pr = pm(vmixr,c);
%      mll= lvv*dsll;
%      mlr= lvh*dslr;
%      mgl= gvv*dgll;
%      mgr= gvh*dglr;
%
%      flc(j,1)= mll*pll+mlr*plr;
%      flc(j,2)= 0.0;
%      flc(j,3)= mll*pll*vlv+mlr*plr*vlh;
%
%      fgc(j,1)=0.0;
%      fgc(j,2)= mgl*pgl+mgr*pgr;
%      fgc(j,3)= mgl*pgl*vgv+mgr*pgr*vgh;
%
%      fp(j,1)= 0.0;
%      fp(j,2)= 0.0;
%      fp(j,3)= pl*psll+pr*pslr;

```

```

end

```

```

% Fluxes have now been calculated. We will now update the conservative
% variables in each of the numerical cells.

```

```

% The source terms can be calculated by using a
% for loop.
% Note that the model is sensitive to how we treat the model
% for low Reynolds numbers (possible discontinuity in the model)
    for j=1:nobox
        fricgrad(j)=dpfric(vlo(j),vgo(j),evo(j),ego(j),dlo(j),dgo(j), ...
            po(j),do(j),di(j),viscl,viscg);
        hydgrad(j)=g*(dlo(j)*evo(j)+dgo(j)*ego(j));
    end

    sumfric = 0;
    sumhyd= 0;

    for j=1:nobox

% Here we solve the three conservation laws for each cell and update
% the conservative variables qv

        ar = area(j);

% Liquid mass conservation
        qv(j,1)=qvo(j,1)-dtdx*( (ar*flc(j+1,1)-ar*flc(j,1)) ...
            + (ar*fgc(j+1,1)-ar*fgc(j,1)) ...
            + (ar*fp(j+1,1)-ar*fp(j,1)));

% Gas mass conservation:
        qv(j,2)=qvo(j,2)-dtdx*( (ar*flc(j+1,2)-ar*flc(j,2)) ...
            + (ar*fgc(j+1,2)-ar*fgc(j,2)) ...
            + (ar*fp(j+1,2)-ar*fp(j,2)));

% Mixture momentum conservation:
        qv(j,3)=qvo(j,3)-dtdx*( (ar*flc(j+1,3)-ar*flc(j,3)) ...
            + (ar*fgc(j+1,3)-ar*fgc(j,3)) ...
            + (ar*fp(j+1,3)-ar*fp(j,3)) ...
            -dt*ar*(fricgrad(j)+hydgrad(j));

% Add up the hydrostatic pressure and friction in the whole well.
        sumfric=sumfric+fricgrad(j)*dx;
        sumhyd=sumhyd+hydgrad(j)*dx;

    end

% Section where we find the physical variables (pressures, densities etc)
% from the conservative variables. Some trickes to ensure stability. These
% are induced to avoid negative masses.

```

```

gasmass=0;
liqmass=0;

for j=1:nobox

% Remove the area from the conservative variables to find the
% the primitive variables from the conservative ones.

qv(j,1)= qv(j,1)/area(j);
qv(j,2)= qv(j,2)/area(j);

if (qv(j,1)<0.00000001)
    qv(j,1)=0.00000001;
end

if (qv(j,2)< 0.00000001)
    qv(j,2)=0.00000001;
end

% Here we summarize the mass of gas and liquid in the well respectively.
gasmass = gasmass+qv(j,2)*area(j)*dx;
liqmass = liqmass+qv(j,1)*area(j)*dx;

% Below, we find the primitive variables pressure and densities based on
% the conservative variables q1,q2. One can choose between getting them by
% analytical or numerical solution approach specified in the beginning of
% the program. Ps. For more advanced density models, this must be changed.

if (analytical == 1)
%     % Analytical solution:

    t1=rho0-P0/a1^2;

% Coefficients:
    a = 1/(a1*a1);
    b = t1-qv(j,1)-rt*qv(j,2)/(a1*a1);
    c = -1.0*t1*rt*qv(j,2);
%

%     Note here we use the very simple models from the PET510 course
    p(j)=(-b+sqrt(b*b-4*a*c))/(2*a); % Pressure
    dl(j)=rho_liq(p(j),temp(j)); % Density of liquid
    dg(j)=rho_gas(p(j),temp(j)); % Density of gas

% The code below can be activated if we want to switch to the other set
% of density models. Also then remember to do the changes inside
% functions rogas og rho_liq.

```

```

%          x1=rho0-P0*rho0/Bheta-rho0*Alpha*(temp(j)-T0);
%          x2=rho0/Bheta;
%          x3=-qv(j,2)*R*temp(j);

%          a = x2;
%          b = x1+x2*x3-qv(j,1);
%          c = x1*x3;

%          p(j)=(-b+sqrt(b*b-4*a*c))/(2*a); % Pressure
%          dl(j)=rholiq(p(j),temp(j));
%          dg(j)=rogas(p(j),temp(j));
else

%Numerical Solution: This might be used if we use more complex
%density models

[p(j),error]=itsolver(po(j),qv(j,1),qv(j,2)); % Pressure
dl(j)=rholiq(p(j),temp(j)); % Density of liquid
dg(j)=rogas(p(j)); % Density of gas

% Incase a numerical solution is not found, the program will write out
"error":
    if error > 0
        error
    end
end

% Find phase volume fractions
eg(j)= qv(j,2)/dg(j);
ev(j)=1-eg(j);

% Reset average conservative variables in cells with area included in the
variables.

qv(j,1)=qv(j,1)*area(j);
qv(j,2)=qv(j,2)*area(j);

end % end of loop

% Below we find the phase velocities by combining the
% conservative variable defined by the mixture momentum equation
% with the gas slip relation.
% At the same time we try to summarize the gas volume in the well. This
% also measure the size of the kick.

gasvol=0;

```

```

for j=1:nobox

% The interpolations introduced below are included
% to omit a singularity in the slip relation when the gas volume
% fraction becomes equal to 1/K. In addition, S is interpolated to
% zero when approaching one phase gas flow. In the transition to
% one phase gas flow, we have no slip conditons (K=1, S=0)

    ktemp=k;
    stemp=s;

    k0(j) = ktemp;
    s0(j) = stemp;

% Interpolation to handle that (1-Kxgasvolumefraction) does not become
zero
    if ((eg(j)>=0.7) & (eg(j)<=0.8))
        xint = (eg(j)-0.7)/0.1;
        k0(j) = 1.0*xint+k*(1-xint);
    elseif(eg(j)>0.8)
        k0(j)=1.0;
    end

% Interpolate S to zero in transition to pure gas phase
    if ((eg(j)>=0.9) & (eg(j)<=1.0))
        xint = (eg(j)-0.9)/0.1;
        s0(j) = 0.0*xint+s*(1-xint);
    end

%
    if (eg(j)>=0.999999)
        % Pure gas
        k1(j) = 1.0;
        s1(j) = 0.0;
    else
        %Two phase flow
        k1(j) = (1-k0(j)*eg(j))/(1-eg(j));
        s1(j) = -1.0*s0(j)*eg(j)/(1-eg(j));
    end

    help1 = dl(j)*ev(j)*k1+dg(j)*eg(j)*k0;
    help2 = dl(j)*ev(j)*s1+dg(j)*eg(j)*s0;

    vmixhelp1 = (qv(j,3)/area(j)-help2)/help1;
    vg(j)=k0(j)*vmixhelp1+s0(j);
    vl(j)=k1(j)*vmixhelp1+s1(j);

```

```

    % Variable for summarizing the gas volume content in the well.
    gasvol=gasvol+eg(j)*area(j)*dx;

end

% Old values are now set equal to new values in order to prepare
% computation of next time level.

po=p;
dlo=dl;
dgo=dg;
vlo=vl;
vgo=vg;
ego=eg;
evo=ev;
qvo=qv;

dlrandinold = dlrandinnew;           %for inlet
prandinold = prandinnew;
vlrandinold = vlrandinnew;

dgrandinold = dgrandinnew;
vgrandinold = vgrandinnew;
egrandinold = egrandinnew;

dlrandoutold = dlrandoutnew;         %for outlet
prandoutold = prandoutnew;
vlrandoutold = vlrandoutnew;
egrandoutold = egrandoutnew;
dgrandoutold = dgrandoutnew;
vgrandoutold = vgrandoutnew;

% % Here we calculate the increase in pitgain. This has been deactivated
% since it is sufficient to use the volgas variable.
% pitrate = (dl(nobox)*ev(nobox)*vl(nobox)*area(nobox))/rho0-...
% inletligmassrate/rho0; %m3/s %NNNNNNNNNNBBBBBBBBBBBBBB
% pitvolume=pitvolume+pitrate*dt; %m3

% Section where we save some timedependent variables in arrays.
% e.g. the bottomhole pressure. They will be saved for certain
% timeintervalls defined in the start of the program in order to ensure
% that the arrays do not get to long!

if (counter>=nostepsbeforesavingtimedata)
    printcounter=printcounter+1;
    time % Write time to screen.

    % Outlet massrates (kg/s) vs time
    liquidmassrateout(printcounter)=dl(nobox)*ev(nobox)*vl(nobox)*area(nobox);

```



```

gasmassrateout(printcounter)=dg(nobox)*eg(nobox)*vg(nobox)*area(nobox);

% Outlet flowrates (lpm) vs time
liquidflowrateout(printcounter)=liquidmassrateout(printcounter)/...
    rho_liq(P0,T0)*1000*60;
gasflowrateout(printcounter)=gasmassrateout(printcounter)/...
    rho_gas(P0,T0)*1000*60;

% Hydrostatic and friction pressure (bar) in well vs time
hyd(printcounter)=sumhyd/100000;
fric(printcounter)=sumfric/100000;

% Volume of gas in well vs time (m3). Also used for indicating kick
% size

volgas(printcounter)=gasvol;

% Total phase masses (kg) in the well vs time
massgas(printcounter)=gasmass;
massliq(printcounter)=liqmass;

% pout defines the exact pressure at the outletboundary!
pout(printcounter)=(p(nobox)-0.5*dx*...
    (dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g-
dx*0.5*fricgrad(nobox))/100000;

% pin (bar) defines the exact pressure at the bottom boundary
pin(printcounter)=
(p(1)+0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g+0.5*dx*fricgrad(1))/100000;

% Pressure in the middle of top box (bar).
pnobox(printcounter)=p(nobox)/100000; %
tempbott(printcounter)=temp(1)-273; % temperature at bottom - fixed
% pitgain(printcounter)=pitvolume; Use volgas for kick size.

% Time variable
timeplot(printcounter)=time;

counter = 0;

end
end

% end of stepping forward in time.

% Printing of resultssection

```

```

countsteps % Marks number of simulation steps.

% Plot commands for variables vs time. The commands can also
% be copied to command screen where program is run for plotting other
% variables.

toc,
e = cputime-t

% Plot bottomhole pressure
plot(timeplot,pin)

% Show cfl number used.
disp('cfl')
cfl = al*dt/dx

set(gcf, 'color', 'w')
figure(1)
plot(timeplot,pin)
title('Pressure at inlet')
ylabel('Pressure, bar')
xlabel('Time, seconds')
grid on
axis([0 20 6 22])
%plot(timeplot,hyd)
%plot(timeplot,fric)
%plot(timeplot,liquidmassrateout)
%plot(timeplot,gasmassrateout)
%plot(timeplot,volgas)
% figure(2)
% plot(timeplot,liquidflowrateout)
% title('liquidflowrateout')
%plot(timeplot,gasflowrateout)
%plot(timeplot,massgas)
% figure(3)
% plot(timeplot,massliq)
% title('massliq')
%plot(timeplot,pout)
%plot(timeplot,pnobox)

%Plot commands for variables vs depth/Only the last simulated
%values at endtime is visualised

% plot(x,vl);
%plot(vg,x);
%plot(eg,x);
% plot(x,p/100000);
%plot(dl,x);
%plot(dg,x);

```

Appendix 2

The AUSMV code for case 2, 3 and 4

Zero-order extrapolation technique, first-order extrapolation technique and compatibility relations method

```
% Transient two-phase code based on AUSMV scheme: Gas and Water
% The code assumes uniform geometry

% time - Seconds

% p - pressure at new time level (Pa)
% dl - density of liquid at new time level (kg/m3)
% dg - density of gas at new time level (kg/m3)
% eg - phase volume fraction of liquid at new time level (0-1)
% ev - phase volume fraction of gas at new time level (0-1)
% vg - phase velocity of gas at new time level (m/s)
% vl - phase velocity of liquid at new time level (m/s)
% qv - conservative variables at new time level ( 3 in each cell)
% temp - temperature in well (K)

% po - pressure at old time level (Pa)
% dlo - density of liquid at old time level (kg/m3)
% dgo - density of gas at new old level (kg/m3)
% ego - phase volume fraction of liquid at old time level (0-1)
% evo - phase volume fraction of gas at old time level (0-1)
% vgo - phase velocity of gas at old time level (m/s)
% vlo - phase velocity of liquid at old time level (m/s)
% qvo - conservative variables at old time level ( 3 in each cell)
% temp - temperature in well (K)

clear;
clf
close all

t = cputime
tic,

% Geometry data/ Must be specified
welldepth = 4000;
nobox = 100; %Number of boxes in the well

% Note that one can use more refined grid, 50, 100 boxes.
% When doing this, remember to reduce time step to keep the CFL number
% fixed below 0.25.. dt < cfl x dx/ speed of sound in water. If boxes are
% doubled, then half the time step.

nofluxes = nobox+1; % Number of cell boundaries
dx = welldepth/nobox; % Boxlength
%dt = 0.005;

% Welldepth. Cell 1 start at bottom
x(1)= -1.0*welldepth+0.5*dx;
```

```

for i=1:nobox-1
    x(i+1)=x(i)+ dx;
end

dt= 0.005; % Timestep (seconds)

dtdx = dt/dx;
time = 0.0; % initial time.
endtime = 6000; % Time for ending simulation (seconds)
nosteps = endtime/dt; %Number of total timesteps. Used in for loop.
timebetweensavingtimedata = 1; % How often in s we save data vs time for
plotting.
nostepsbeforesavingtimedata = timebetweensavingtimedata/dt;

% Slip parameters used in the gas slip relation.  $v_g = K v_{mix} + S$ 
k = 1.2;
s = 0.55;

% Boundary condition at outlet
pbondout=100000; % Pascal (1 bar)

% Initial temperature distribution. (Kelvin)

tempbot = 110+273;
temptop = 50+273;
tempgrad= (tempbot-temptop)/welldepth;
tempo(1)=tempbot-dx/2*tempgrad;
for i = 1:nobox-2
    tempo(i+1)=tempo(i)-dx*tempgrad;
end
tempo(nobox)=tempo(nobox-1)-dx*tempgrad;

temp = tempo;

% Different fluid density parameters
% Note how we switch between different models later.
% These parameters are used when finding the
% primitive variables pressure, densities in an analytical manner.
% Changing parameters here, you must also change parameters inside the
% density routines roliq and rogas.

% Simple Water density model & Ideal Gas. See worknote Extension of AUSMV
% scheme.

rho0=1000; % Water density at STC (Standard Condition) kg/m3
Bbeta=2.2*10^9; % Parameter that depend on the compressibility of water
Alpha=0.000207; % Parameter related to thermal expansion/compression
R = 286.9; % Ideal gas parameter
P0=100000; % Pressure at STC (Pa)
T0=20+273.15; % Temperature at STC (K)

% Very simple models (PET510 compendium)

```

```

al = 1500; % Speed of sound in water.
rt= 100000; % Ideal gas parameter in model  $\rho g = p/rt$ 
rho0=1000; % Water density at STC (Standard Condition) kg/m3
P0=100000; % Pressure at STC (Pa)
T0=20+273.15; % Temperature at STC (K)

% Viscosities (Pa*s)/Used in the frictional pressure loss model (dpfric).
viscl = 0.001; % Liquid phase
viscg = 0.0000182; % Gas phase

% Gravity constant

g = 9.81; % Gravitational constant m/s2

% Well opening. opening = 1, fully open well, opening = 0 (<0.01), the well
% is fully closed. This variable will control what boundary conditions that
% will apply at the outlet (both physical and numerical): We must change
% this further below in the code if we want to change status on this.

wellopening = 1.0; % This variable determines if
%the well is closed or not, wellopening = 1.0 -> open. welllopening = 0
%-> Well is closed. This variable affects the boundary treatment.

bullheading = 0.0; % This variable can be set to 1.0 if we want to simulate
% a bullheading operation. But the normal is to set this to zero.

% Specify if the primitive variables shall be found either by
% a numerical or analytical approach. If analytical = 1, analytical
% solution is used. If analytical = 0. The numerical approach is used.
% using the itsolver subroutine where the bisection numerical method
% is used. We use analytical.

analytical = 1;

% Initialization of rest of geometry.
% Here we specify the outer and inner diameter and the flow area
% We assume 8.5 x 5 inch annulus.

for i = 1:nobox

```

```

do(i)=12.25*2.54/100;
di(i) = 5*2.54/100;      % pipe

area(i) = 3.14/4*(do(i)*do(i)- di(i)*di(i));
% ang(i)=3.14/2;
end

% Initialization of slope limiters. These are used for
% reducing numerical diffusion and will be calculated for each timestep.
% They make the numerical scheme second order.
for i = 1:nobox
    sl1(i)=0;
    sl2(i)=0;
    sl3(i)=0;
    sl4(i)=0;
    sl5(i)=0;
    sl6(i)=0;
end

% Now comes the initialization of the physical variables in the well.
% First primitive variables, then the conservative ones.

% Below we initialize pressure and fluid densities. We start from top of
% the well and calculated downwards. The calculation is done twice with
% updated values to get better approximation. Only hydrostatic
% considerations since we start with a static well.

for i = 1:nobox
    eg(i)=0.0; % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
end

p(nobox)= pbondout+0.5*g*dx*(ev(nobox)*rho0+eg(nobox)*1); % Pressure
dl(nobox)=rholiq(p(nobox),tempo(nobox)); % Liquid density
dg(nobox)=rogas(p(nobox),tempo(nobox)); % Gas density

for i=nobox-1:-1:1
    p(i)=p(i+1)+dx*g*(ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    dl(i)=rholiq(p(i),tempo(i));
    dg(i)=rogas(p(i),tempo(i));
end

for i=nobox-1:-1:1
    rhoavg1= (ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    rhoavg2= (ev(i)*dl(i)+eg(i)*dg(i));

```

```

p(i)=p(i+1)+dx*g*(rhoavg1+rhoavg2)*0.5;
dl(i)=rholiq(p(i),tempo(i));
dg(i)=rogas(p(i),tempo(i));

end

% Intitalize phase velocities, volume fractions, conservative variables
% and friction and hydrostatic gradients.
% The basic assumption is static fluid, one phase liquid.

for i = 1:nobox
    vl(i)=0; % Liquid velocity new time level.
    vg(i)=0; % Gas velocity at new time level
    eg(i)=0.0; % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
    qv(i,1)=dl(i)*ev(i)*area(i);
    qv(i,2)=dg(i)*eg(i)*area(i);
    qv(i,3)=(dl(i)*ev(i)*vl(i)+dg(i)*eg(i)*vg(i))*area(i);
    fricgrad(i)=0;
    hydgrad(i)=g*(dl(i)*ev(i)+eg(i)*dg(i));
end

% Section where we also initialize values at old time level

for i=1:nobox
    dlo(i)=dl(i);
    dgo(i)=dg(i);
    po(i)=p(i);
    ego(i)=eg(i);
    evo(i)=ev(i);
    vlo(i)=vl(i);
    vgo(i)=vg(i);
    qvo(i,1)=qv(i,1);
    qvo(i,2)=qv(i,2);
    qvo(i,3)=qv(i,3);
end

dlandinold = dlo(1); %Initializing for compatibility relation,
prandinold = po(1); %For inlet
vlandinold = vlo(1);
dlandinnew = dlandinold;
prandinnew = prandinold;
vlandinnew = vlandinold;

dgrandinold = dgo(1);
vgrandinold = vgo(1);
egrandinold = ego(1);
dgrandinnew = dgrandinold;
vgrandinnew = vgrandinold;
egrandinnew = egrandinold;

dlandoutold = dlo(nobox); %For outlet

```

```

prandoutold = po(nobox);
vrandoutold = vlo(nobox);
egrandoutold = ego(nobox);
dgrandoutold = dgo(nobox);
vgrandoutold = vgo(nobox);

dlandoutnew = dlandoutold;
prandoutnew = prandoutold;
vrandoutnew = vrandoutold;
egrandoutnew = egrandoutold;
dgrandoutnew = dgrandoutold;
vgrandoutnew = vgrandoutold;

% Intialize fluxes between the cells/boxes

for i = 1:nofluxes
    for j =1:3
        flc(i,j)=0.0; % Flux of liquid over box boundary
        fgc(i,j)=0.0; % Flux of gas over box boundary
        fp(i,j)= 0.0; % Pressure flux over box boundary
    end
end

% Main program. Here we will progress in time. First som intializations
% and definitions to take out results. The for loop below runs until the
% simulation is finished.

countsteps = 0;
counter=0;
printcounter = 1;
pin(printcounter) = (p(1)+dx*0.5*hydgrad(1))/100000; % Pressure at bottom for
time storage
pout(printcounter)= pbondout/100000;
pnobox(printcounter)= p(nobox)/100000;
liquidmassrateout(printcounter) = 0;
gasmassrateout(printcounter)=0;
tempbott(printcounter)=tempo(1)-273;
timeplot(printcounter)=time; % Array for time and plotting of variables vs
time
pitvolume=0;
pitrates =0;
pitgain(printcounter)=0;

kickvolume=0;
bullvolume=0;

% The temperature is not updated but kept fixed according to the
% initialization.

for i = 1:nosteps
    countsteps=countsteps+1;

```



```

counter=counter+1;
time = time+dt; % Step one timestep and update time.

% Then a section where specify the boundary conditions.
% Here we specify the inlet rates of the different phases at the
% bottom of the pipe in kg/s. We interpolate to make things smooth.
% It is also possible to change the outlet boundary status of the well
% here. First we specify rates at the bottom and the pressure at the outlet
% in case we have an open well. This is a place where we can change the
% code to control simulations. If the well shall be close, wellopening must
% be set to 0.

% In the example below, we take a gas kick and then circulate this
% out of the well without closing the well. (how you not should perform
% well control)

XX = 8; % Gasrate in kg/s

YY= 40; % Liquidrate in kg/s (for case 4)

if (time < 10)

    inletligmassrate=0.0;
    inletgasmassrate=0.0;

elseif ((time >=10) && (time<20))
% inletligmassrate = YY-YY*(time-200)/10;
% inletligmassrate = YY;
    inletgasmassrate = XX*(time-10)/10;
elseif ((time >=20) && (time<110))
    inletgasmassrate=XX;
elseif ((time >=110) && (time<120))
% inletligmassrate=YY;
    inletgasmassrate = XX-XX*(time-110)/10;
    inletligmassrate = YY*(time-110)/10;
    elseif(time>=120)
    inletligmassrate=0;
    inletgasmassrate=0;
    inletligmassrate=YY;
% elseif(time>=120 && time <130)
% inletligmassrate=0;
% inletgasmassrate=0;
% inletligmassrate=YY;
% wellopening=0.0;
% elseif(time>=130)
% wellopening =0;
end

kickvolume = kickvolume+inletgasmassrate/dgo(1)*dt;

```

```

% specify the outlet pressure /Physical. Here we have given the pressure as
% constant. It would be possible to adjust it during openwell conditions
% either by giving the wanted pressure directly (in the command lines
% above) or by finding it indirectly through a chokemodel where the
wellopening
% would be an input parameter. The wellopening variable would equally had
% to be adjusted inside the command line structure given right above.

```

```

pressureoutlet = pbondout;

```

```

% Based on these boundary values combined with use of extrapolations
techniques
% for the remaining unknowns at the boundaries, we will define the mass and
% momentum fluxes at the boundaries (inlet and outlet of pipe).

```

```

% inlet/bottom fluxes first.

```

```

    if (bullheading<=0)

```

```

        % Here we pump from bottom

```

```

            flc(1,1)= inletligmassrate/area(1);

```

```

            flc(1,2)= 0.0;

```

```

            flc(1,3)= flc(1,1)*vlo(1);

```

```

            fgc(1,1)= 0.0;

```

```

            fgc(1,2)= inletgasmassrate/area(1);

```

```

            fgc(1,3)= fgc(1,2)*vgo(1);

```

```

            fp(1,1)= 0.0;

```

```

            fp(1,2)= 0.0;

```

```

        if (eg(1) < 0.001)

```

```

            w = 1500;

```

```

        elseif ((eg(1) >= 0.001) && (eg(1) < 0.999))

```

```

            if (eg(1) <= 0.7)

```

```

                ktemp = k;

```

```

            elseif ((eg(1) > 0.7) && (eg(1) < 0.8))

```

```

                xint = (eg(1) - 0.7)/0.1;

```

```

                ktemp = (1-xint)*k + 1.0*xint;

```

```

            else

```

```

                ktemp = 1.0;

```

```

            end

```

```

        w = sqrt(po(1)/(eg(1)*dlo(1)*(1-ktemp*eg(1))));

```

```

        if (eg(1) < 0.5)

```

```

            w = min(1500, w);

```

```

        else

```

```

            w = min(316, w);

```

```

        end

```

```

    else

```

```

        w = 316;

```

```

    end

```

```

% vlrandinnew = inletligmassrate / (dlrandinnew * area(1));

if (eg(1) < 0.001)
    vlrandinnew = inletligmassrate / (dlrandinnew * area(1));
    aaa = (vlo(1) - al);
    bbb= (po(1) - prandinold) / ( dx / 2 );
    ccc = dlo(1) * al;
    ddd = (vlrandinnew - vlrandinold) / ( dt );
    fff = (vlo(1) - vlrandinold) / ( dx / 2 );
%     e = g * dlo(1) * al;           %no friction included
    ggg = al * (hydgrad(1) + fricgrad(1));

    prandinnew = prandinold + dt * ( ggg - aaa * bbb + ccc * (ddd + aaa *
fff) );

    dlrandinnew = rho(liq(prandinnew));
    egrandinnew = eg(1);

else %(eg(1) >= 0.001) && (eg(1) < 0.999)

% MERK DETTE ER SUPERFICIALE HASTIGHETER
    vlrandinnew = inletligmassrate / (dlrandinnew * area(1));
    vgrandinnew = inletgasmassrate / (dgrandinnew * area(1));
% egrandinnew = vgrandinnew/ (vgrandinnew + vlrandinnew + s);
% egrandinnew = vgrandinnew/ (vgrandinnew + vlrandinnew);

%           % VLRANDINNEW BLIR HER KORRIGERT SLIK AT DET BLIR FASEHASTIGHET.
%           % MERK DENNE BRYTER SAMMEN HVIS GASFRAKSJONEN BLIR 1 MEN DET VIL
%           % DEN BARE BLI HVIS K = 1.0. S=0 I KOMB MED NULL VÆSKERATE
    vgtemp = ktemp*(vlrandinnew+vgrandinnew)+s;
    egrandinnew = vgrandinnew/vgtemp;
    vlrandinnew = vlrandinnew/(1-egradinnew);

    aa = (vlo(1) - w) * (po(1) - prandinold) / ( dx / 2 );
    bb = (dlo(1) * w * (vgo(1) - vlo(1)));
    %bb = 0;
    cc = (egradinnew - egrandinold) / (dt);
%     cc = 0.0;
    dd = (vlo(1) - w) * (ego(1) - egrandinold) / (dx / 2);
%     dd = 0.0;
    ee = dlo(1) * (vgo(1) - vlo(1) + w) * (1 - ego(1));
    ff = (vlrandinnew - vlrandinold) / dt;
    gg = (vlo(1) - w) * (vlo(1) - vlrandinold) / (dx/2);
    hh = g * ((dlo(1) * (1-ego(1)) + dgo(1)*ego(1)) * (vgo(1)-vlo(1)+w));
% LEGGER TIL FRIKSJONEN
    hh = hh + fricgrad(1)*(vgo(1)-vlo(1)+w);

```

```

prandinnew = prandinold + dt * (hh - aa + bb * ( cc + dd ) + ee * (ff
+ gg));

dlandinnew = rholiq(prandinnew);
dgrandinnew = rogas (prandinnew);

%
end
% Old way of treating the boundary
% fp(1,3)= po(1)+0.5*(po(1)-po(2)); %Interpolation used to find the
% pressure at the inlet/bottom of the well.
%
% New way of treating the boundary
% fp(1,3)= po(1)...
% +0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g...
% +0.5*dx*fricgrad(1);

%compatibility relation

fp(1,3) = prandinnew;

else
% Here we pump from the top. All masses are assumed to flow out of the
% well into the formation. We use first order extrapolation.
flc(1,1)=dlo(1)*evo(1)*vlo(1);
flc(1,2)=0.0;
flc(1,3)=flc(1,1)*vlo(1);

fgc(1,1)=0.0;
fgc(1,2)=dgo(1)*ego(1)*vgo(1);
fgc(1,3)=fgc(1,2)*vgo(1);

fp(1,1)=0.0;
fp(1,2)=0.0;
fp(1,3)=20000000; % This was a fixed pressure set at bottom when
bullheading
end

% Outlet fluxes (open & closed conditions)

if (wellopening>0.01)

% Here open end conditons are given. We distinguish between bullheading
% & normal circulation.

if (bullheading<=0)

% Here the is normal ciruclation and open well)

```

```

if (eg(nobox) < 0.001)
    w = 1500;
elseif ((eg(nobox) >= 0.001) && (eg(nobox) < 0.999))
    if (eg(nobox) <= 0.6)
        ktemp = k;
    elseif ((eg(nobox) > 0.6) && (eg(nobox) < 0.8))
        xint = (eg(nobox) - 0.6)/0.2;
        ktemp = (1-xint)*k + 1.0*xint;
    else
        ktemp = 1.0;
    end
end

w = sqrt(po(nobox)/(eg(nobox)*dlo(nobox)*(1-ktemp*eg(nobox))));

if (eg(nobox) < 0.5)
    w = min(1500, w);
else
    w = min(316, w);
end

else
    w = 316;
end

prandoutnew = 100000;    %1 bar at top of well - open
% prandoutold = prandoutnew;

if eg(nobox) < 0.001

    aa = (prandoutnew - prandoutold) / dt;
    bb = vlo(nobox) + al;
    cc = (prandoutold - po(nobox)) / (dx/2);
    dd = dlo(nobox) * al;
    ff = (vlrandoutold - vlo(nobox)) / (dx/2);
% gg = g*dlo(nobox)*al;
    gg = al * (hydgrad(nobox) + fricgrad(nobox));

    vlrandoutnew = vlrandoutold - dt/dd * (gg+aa+bb*cc+dd*bb*ff);
%6.12

% egrandoutnew = 0;
    egrandoutnew = ego(nobox);

    vgrandoutnew = (k*(1-egrandoutnew)*vlrandoutnew+s)/(1-
k*egrandoutnew); %6.17
% dlrandoutnew = rholiq(prandoutnew);
% dgrandoutnew = rogas(prandoutnew);

```

```

elseif ((eg(nobox) > 0.001) &&(eg(nobox)<0.99))

    aa = (prandoutnew - prandoutold) / dt;
    bb = (vgo(nobox));
    cc = (prandoutold - po(nobox)) / (dx/2);
    dd = dlo(nobox) * w*w;
    ee = (egrandoutold - ego(nobox)) / (dx/2);

    egrandoutnew = egrandoutold - dt/dd * (aa+bb*cc+dd*bb*ee);

%6.15

    aa = (prandoutnew -prandoutold) / dt;
    bb = vlo(nobox) + w;
    cc = (prandoutold - po(nobox)) / (dx/2);
    dd = (dlo(nobox)*w*(vgo(nobox)-vlo(nobox)));
    ii = (egrandoutnew - egrandoutold) / dt;
    ee = (egrandoutold - ego(nobox)) / (dx/2);
    ff = dlo(nobox)*(vgo(nobox)-vlo(nobox)-w)*(1-ego(nobox));
    gg = (vrandoutold - vlo(nobox)) / (dx/2);
    hh = g * ((dlo(nobox)*(1-ego(nobox)))*(vgo(nobox)-vlo(nobox)-
w));

    hh = hh + fricgrad(nobox)*(vgo(nobox)-vlo(nobox)-w);

    vrandoutnew = vrandoutold + dt/ff * (aa+bb*cc+dd*(ii+bb*ee)-
hh-ff*bb*gg); %6.16

    if eg(nobox) > 0.8
        vgrandoutnew = vrandoutnew;
    else
        vgrandoutnew = (ktemp*(1-egrandoutnew)*vrandoutnew + s) /
(1-ktemp*egrandoutnew); %6.17
    end

    else
% enfase gass:
% Her er det brukt både 1st og 2 ordens komprelasjoner.
    egrandoutnew=eg(nobox);
    egrandoutnew=ego(nobox)+0.5*(ego(nobox)-ego(nobox-1));
    vrandoutnew=vlo(nobox);
    vrandoutnew=vlo(nobox)+0.5*(vlo(nobox)-vlo(nobox-1));
    vgrandoutnew= vrandoutnew;

end

    dlrandoutnew = rholiq(prandoutnew);
    dgrandoutnew = rogas(prandoutnew);

%

    evvv = evo(nobox)+0.5*(evo(nobox)-evo(nobox-1));
    vvvv = vlo(nobox)+0.5*(vlo(nobox)-vlo(nobox-1));
    dlll = dlo(nobox)+0.5*(dlo(nobox)-dlo(nobox-1));

% %
%
    evvv = evo(nobox);

```

```

%          vvvv = vlo(nobox) ;
%          d111 = dlo(nobox) ;

%          flc(nofluxes,1)= rholiq(100000,293.15)*evvv*vvvv;
flc(nofluxes,1)= d111*evvv*vvvv;
flc(nofluxes,2)= 0.0;
flc(nofluxes,3)= flc(nofluxes,1)*vvvv;

gvvv = 1-evvv;
dggg = dgo(nobox)+0.5*(dgo(nobox)-dgo(nobox-1));
vgvv = vgo(nobox)+0.5*(vgo(nobox)-vgo(nobox-1));

%          gvvv = 1-evvv;
%          dggg = dgo(nobox) ;
%          vgvv = vgo(nobox) ;

fgc(nofluxes,1)= 0.0;
%          fgc(nofluxes,2)= rogas(100000,293.15)*gvvv*vgvv;
fgc(nofluxes,2)= dggg*gvvv*vgvv;
fgc(nofluxes,3)= fgc(nofluxes,2)*vgvv;

fp(nofluxes,1)= 0.0;
fp(nofluxes,2)= 0.0;
fp(nofluxes,3)= pressureoutlet;

%Comp relation
flc(nofluxes,1) = dlrandoutnew*(1-egrandoutnew)*vlrandoutnew;
flc(nofluxes,2) = 0.0;
flc(nofluxes,3) = flc(nofluxes,1)*vlrandoutold;

fgc(nofluxes, 1) = 0.0;
fgc(nofluxes, 2) = dgrandoutnew*egrandoutnew*vgrandoutnew;
fgc(nofluxes, 3) = fgc(nofluxes, 2)*vgrandoutnew;

fp(nofluxes,1)= 0.0;
fp(nofluxes,2)= 0.0;
fp(nofluxes,3)= pressureoutlet;

%

else
% Here we are bullheading.
flc(nofluxes,1)= inletligmassrate/area(nobox) ;
flc(nofluxes,2)= 0.0;
flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox) ;

fgc(nofluxes,1)=0.0;
fgc(nofluxes,2)=0.0;
fgc(nofluxes,3)=0.0;

fp(nofluxes,1)=0.0;

```

```

        fp(nofluxes,2)=0.0;
        fp(nofluxes,3)= po(nobox) ...
        -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g...
        +0.5*dx*fricgrad(nobox); %check sign here on friction
    end
else

% Here closed end conditions are given

    flc(nofluxes,1)= 0.0;
    flc(nofluxes,2)= 0.0;
    flc(nofluxes,3)= 0.0;

    fgc(nofluxes,1)= 0.0;
    fgc(nofluxes,2)= 0.0;
    fgc(nofluxes,3)= 0.0;

    fp(nofluxes,1)=0.0;
    fp(nofluxes,2)=0.0;

    if (eg(nobox) < 0.001)
        w = 1500;
    elseif ((eg(nobox) >= 0.001) && (eg(nobox) < 0.999))
        if (eg(nobox) <= 0.7)
            ktemp = k;
        elseif ((eg(nobox) > 0.7) && (eg(nobox) < 0.8))
            xint = (eg(nobox) - 0.7)/0.1;
            ktemp = (1-xint)*k + 1.0*xint;
        else
            ktemp = 1.0;
        end
    end

    w = sqrt(po(nobox)/(eg(nobox)*dlo(nobox)*(1-ktemp*eg(nobox))));

    if (eg(nobox) < 0.5)
        w = min(1500, w);
    else
        w = min(316, w);
    end

else
    w = 316;
end

if eg(nobox) < 0.001

    aaa = vlo(nobox) + al;
    bbb = (prandoutold - po(nobox))/(dx/2);
    ccc = dlo(nobox)*al;
    ddd = aaa;

```



```

eee = (0 - vlo(nobox)) / (dx/2);
fff = -(g * dlo(nobox) * al);
vrandoutnew = 0;
vrandoutold = 0;

prandoutnew = prandoutold + dt*(fff-aaa*bbb-ccc*(aaa*eee));

dlrandoutnew =rholiq(prandoutnew);
dgrandinnew =rogas(prandoutnew);
egrandoutnew = eg(nobox);

else

    egrandoutnew = ego(nobox);
    vrandoutnew = 0;
    vrandoutold = 0;

    aaa = vlo(nobox) + w;
    bbb = (prandoutold - po(nobox)) / (dx/2);
    ccc = dlo(nobox) * w * (vgo(nobox) - vlo(nobox));
    ddd = (egrandoutnew - egrandoutold) / dt;
    eee = (egrandoutold - ego(nobox)) / (dx/2);
    fff = dlo(nobox) * (vgo(nobox) - vlo(nobox) - w) * (1 - ego(nobox));
    ggg = (vrandoutnew - vrandoutold) / dt;
    hhh = (vrandoutold - vlo(nobox)) / (dx/2);
    iii = g * ((dlo(nobox) * (1 -
ego(nobox) + dgo(nobox) * ego(nobox)) * (vgo(nobox) - vlo(nobox) - w));

    prandoutnew = prandoutold + dt*(iii - aaa*bbb - ccc*(ddd+aaa*eee)+
fff*(ggg+aaa*hhh));

end
% Old way of treating the boundary
% fp(nofluxes,3) = po(nobox) - 0.5*(po(nobox-1) - po(nobox));

% New way of treating the boundary
fp(nofluxes,3) = po(nobox) ...
-0.5*dx*(dlo(nobox)*evo(nobox) + dgo(nobox)*ego(nobox))*g;
% -0.5*dx*fricgrad(nobox); % Neglect friction since well is closed.

% fp(nofluxes,3) = prandoutnew;
end

% Implementation of slopelimiters. They are applied on the physical
% variables like phase densities, phase velocities and pressure.

% It was found that if the slopelimiters were set to zero in
% the boundary cells, the pressure in these became wrong. E.g. the upper
% cell get an interior pressure that is higher than it should be e.g. when

```

```

% being static (hydrostatic pressure was too high). The problem was reduced
% by copying the slopelimiters from the interior cells. However, both
% approaches seems to give the same BHP pressure vs time but the latter
% approach give a more correct pressure vs depth profile. It is also better
% to use when simulating pressure build up where the upper cell pressure
% must be monitored. It should be checked more in detail before concluding.

```

```

for i=2:nobox-1
    s11(i)=minmod(dlo(i-1),dlo(i),dlo(i+1),dx);
    s12(i)=minmod(po(i-1),po(i),po(i+1),dx);
    s13(i)=minmod(vlo(i-1),vlo(i),vlo(i+1),dx);
    s14(i)=minmod(vgo(i-1),vgo(i),vgo(i+1),dx);
    s15(i)=minmod(ego(i-1),ego(i),ego(i+1),dx);
    s16(i)=minmod(dgo(i-1),dgo(i),dgo(i+1),dx);
end

```

```

% Slopelimiters in outlet boundary cell are set to zero!
% %      s11(nobox)=0;
% %      s12(nobox)=0;
% %      s13(nobox)=0;
% %      s14(nobox)=0;
% %      s15(nobox)=0;
% %      s16(nobox)=0;

```

```

% Slopelimiters in outlet boundary cell are copied from neighbour cell!
s11(nobox)=s11(nobox-1);
s12(nobox)=s12(nobox-1);
s13(nobox)=s13(nobox-1);
s14(nobox)=s14(nobox-1);
s15(nobox)=s15(nobox-1);
s16(nobox)=s16(nobox-1);

```

```

% Slopelimiters in inlet boundary cell are set to zero!
% %      s11(1)=0;
% %      s12(1)=0;
% %      s13(1)=0;
% %      s14(1)=0;
% %      s15(1)=0;
% %      s16(1)=0;

```

```

% Slopelimiters in inlet boundary cell are copied from neighbour cell!
s11(1)=s11(2);
s12(1)=s12(2);
s13(1)=s13(2);
s14(1)=s14(2);
s15(1)=s15(2);
s16(1)=s16(2);

```

```

% Now we will find the fluxes between the different cells.
% NB - IMPORTANT - Note that if we change the compressibilities/sound
velocities of
% the fluids involved, we may need to do changes inside the csound function.
% But the effect of this is unclear.

```

```

for j = 2:nofluxes-1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First order method is from here: If you want to test this, activate this
% and comment the second order code below.
%     cl = csound(ego(j-1),po(j-1),dlo(j-1),k);
%     cr = csound(ego(j),po(j),dlo(j),k);
%     c = max(cl,cr);
%     pll = psip(vlo(j-1),c,evo(j));
%     plr = psim(vlo(j),c,evo(j-1));
%     pgl = psip(vgo(j-1),c,ego(j));
%     pgr = psim(vgo(j),c,ego(j-1));
%     vmixr = vlo(j)*evo(j)+vgo(j)*ego(j);
%     vmixl = vlo(j-1)*evo(j-1)+vgo(j-1)*ego(j-1);
%
%     pl = pp(vmixl,c);
%     pr = pm(vmixr,c);
%     mll= evo(j-1)*dlo(j-1);
%     mlr= evo(j)*dlo(j);
%     mgl= ego(j-1)*dgo(j-1);
%     mgr= ego(j)*dgo(j);
%
%     flc(j,1)= mll*pll+mlr*plr;
%     flc(j,2)= 0.0;
%     flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);
%
%     fgc(j,1)=0.0;
%     fgc(j,2)= mgl*pgl+mgr*pgr;
%     fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);
%
%     fp(j,1)= 0.0;
%     fp(j,2)= 0.0;
%     fp(j,3)= pl*po(j-1)+pr*po(j);

% First order methods ends here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Second order method starts here:
% Here slopelimiter is used on all variables except phase velocities

psll = po(j-1)+dx/2*s12(j-1);
pslr = po(j)-dx/2*s12(j);
dsl1 = dlo(j-1)+dx/2*s11(j-1);
dslr = dlo(j)-dx/2*s11(j);
dgl1 = dgo(j-1)+dx/2*s16(j-1);
dglr = dgo(j)-dx/2*s16(j);

vlv = vlo(j-1)+dx/2*s13(j-1);

```

```

vlh = vlo(j)-dx/2*s13(j);
vgv = vgo(j-1)+dx/2*s14(j-1);
vgh = vgo(j)-dx/2*s14(j);

gvv = ego(j-1)+dx/2*s15(j-1);
gvh = ego(j)-dx/2*s15(j);
lvv = 1-gvv;
lvh = 1-gvh;

cl = csound(gvv,psll,dsll,k);
cr = csound(gvh,pslr,dslr,k);
c = max(cl,cr);

pll = psip(vlo(j-1),c,lvh);
plr = psim(vlo(j),c,lvv);
pgl = psip(vgo(j-1),c,gvh);
pgr = psim(vgo(j),c,gvv);
vmixr = vlo(j)*lvh+vgo(j)*gvh;
vmixl = vlo(j-1)*lvv+vgo(j-1)*gvv;

pl = pp(vmixl,c);
pr = pm(vmixr,c);

mll= lvv*dsll;
mlr= lvh*dslr;
mgl= gvv*dgll;
mgr= gvh*dglr;

flc(j,1)= mll*pll+mlr*plr;
flc(j,2)= 0.0;
flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);

fgc(j,1)=0.0;
fgc(j,2)= mgl*pgl+mgr*pgr;
fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);

fp(j,1)= 0.0;
fp(j,2)= 0.0;
fp(j,3)= pl*psll+pr*pslr;

%%% Second order method ends here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Here sloplimiters is used on all variables. This
% has not worked so well yet. Therefore it is commented away.

%     psll = po(j-1)+dx/2*s12(j-1);
%     pslr = po(j)-dx/2*s12(j);

```

```

%      dsll = dlo(j-1)+dx/2*s11(j-1);
%      dslr = dlo(j)-dx/2*s11(j);
%      dgll = dgo(j-1)+dx/2*s16(j-1);
%      dglr = dgo(j)-dx/2*s16(j);
%
%      vlv = vlo(j-1)+dx/2*s13(j-1);
%      vlh = vlo(j)-dx/2*s13(j);
%      vgv = vgo(j-1)+dx/2*s14(j-1);
%      vgh = vgo(j)-dx/2*s14(j);
%
%      gvv = ego(j-1)+dx/2*s15(j-1);
%      gvh = ego(j)-dx/2*s15(j);
%      lvv = 1-gvv;
%      lvh = 1-gvh;
%
%      cl = csound(gvv,psll,dsll,k);
%      cr = csound(gvh,pslr,dslr,k);
%      c = max(cl,cr);
%
%      pll = psip(vlv,c,lvh);
%      plr = psim(vlh,c,lvv);
%      pgl = psip(vgv,c,gvh);
%      pgr = psim(vgh,c,gvv);
%      vmixr = vlh*lvh+vgh*gvh;
%      vmixl = vlv*lvv+vgv*gvv;
%
%      pl = pp(vmixl,c);
%      pr = pm(vmixr,c);
%      mll= lvv*dsll;
%      mlr= lvh*dslr;
%      mgl= gvv*dgll;
%      mgr= gvh*dglr;
%
%      flc(j,1)= mll*pll+mlr*plr;
%      flc(j,2)= 0.0;
%      flc(j,3)= mll*pll*vlv+mlr*plr*vlh;
%
%
%      fgc(j,1)=0.0;
%      fgc(j,2)= mgl*pgl+mgr*pgr;
%      fgc(j,3)= mgl*pgl*vgv+mgr*pgr*vgh;
%
%      fp(j,1)= 0.0;
%      fp(j,2)= 0.0;
%      fp(j,3)= pl*psll+pr*pslr;

```

end

```

% Fluxes have now been calculated. We will now update the conservative
% variables in each of the numerical cells.

```

```

% The source terms can be calculated by using a
% for loop.

```

```

% Note that the model is sensitive to how we treat the model
% for low Reynolds numbers (possible discontinuity in the model)
    for j=1:nobox
        fricgrad(j)=dpfric(vlo(j),vgo(j),evo(j),ego(j),dlo(j),dgo(j), ...
            po(j),do(j),di(j),viscl,viscg);
        hydgrad(j)=g*(dlo(j)*evo(j)+dgo(j)*ego(j));
    end

    sumfric = 0;
    sumhyd= 0;

    for j=1:nobox

% Here we solve the three conservation laws for each cell and update
% the conservative variables qv

        ar = area(j);

% Liquid mass conservation
        qv(j,1)=qvo(j,1)-dtdx*((ar*f1c(j+1,1)-ar*f1c(j,1)) ...
            +(ar*f1g(j+1,1)-ar*f1g(j,1)) ...
            +(ar*f1p(j+1,1)-ar*f1p(j,1)));

% Gas mass conservation:

        qv(j,2)=qvo(j,2)-dtdx*((ar*f2c(j+1,2)-ar*f2c(j,2)) ...
            +(ar*f2g(j+1,2)-ar*f2g(j,2)) ...
            +(ar*f2p(j+1,2)-ar*f2p(j,2)));

% Mixture momentum conservation:

        qv(j,3)=qvo(j,3)-dtdx*((ar*f3c(j+1,3)-ar*f3c(j,3)) ...
            +(ar*f3g(j+1,3)-ar*f3g(j,3)) ...
            +(ar*f3p(j+1,3)-ar*f3p(j,3)) ...
            -dt*ar*(fricgrad(j)+hydgrad(j));

% Add up the hydrostatic pressure and friction in the whole well.
        sumfric=sumfric+fricgrad(j)*dx;
        sumhyd=sumhyd+hydgrad(j)*dx;

    end

% Section where we find the physical variables (pressures, densities etc)
% from the conservative variables. Some trickes to ensure stability. These
% are induced to avoid negative masses.

    gasmass=0;
    liqmass=0;

```

```

for j=1:nobox

% Remove the area from the conservative variables to find the
% the primitive variables from the conservative ones.

    qv(j,1)= qv(j,1)/area(j);
    qv(j,2)= qv(j,2)/area(j);

%     if (qv(j,1)<0.00000001)
%         qv(j,1)=0.00000001;
%     end

    if (qv(j,2)< 0.000000000001)
        qv(j,2)=0.000000000001;
    end

%
% Here we summarize the mass of gas and liquid in the well respectively.
    gasmass = gasmass+qv(j,2)*area(j)*dx;
    liqmass = liqmass+qv(j,1)*area(j)*dx;

% Below, we find the primitive variables pressure and densities based on
% the conservative variables q1,q2. One can choose between getting them by
% analytical or numerical solution approach specified in the beginning of
% the program. Ps. For more advanced density models, this must be changed.

    if (analytical == 1)
%         % Analytical solution:

        t1=rho0-P0/al^2;

% Coefficients:
        a = 1/(al*al);
        b = t1-qv(j,1)-rt*qv(j,2)/(al*al);
        c = -1.0*t1*rt*qv(j,2);
%
%     Note here we use the very simple models from the PET510 course
        p(j)=(-b+sqrt(b*b-4*a*c))/(2*a); % Pressure
        dl(j)=rholiq(p(j),temp(j)); % Density of liquid
        dg(j)=rogas(p(j),temp(j)); % Density of gas

% The code below can be activated if we want to switch to the other set
% of density models. Also then remember to do the changes inside
% functions rogas og rholiq.

%         x1=rho0-P0*rho0/Bheta-rho0*Alpha*(temp(j)-T0);
%         x2=rho0/Bheta;
%         x3=-qv(j,2)*R*temp(j);

%         a = x2;

```

```

%         b = x1+x2*x3-qv(j,1);
%         c = x1*x3;

%         p(j)=(-b+sqrt(b*b-4*a*c))/(2*a); % Pressure
%         dl(j)=rholiq(p(j),temp(j));
%         dg(j)=rogas(p(j),temp(j));
else

%Numerical Solution: This might be used if we use more complex
%density models

    [p(j),error]=itsolver(po(j),qv(j,1),qv(j,2)); % Pressure
    dl(j)=rholiq(p(j),temp(j)); % Density of liquid
    dg(j)=rogas(p(j)); % Density of gas

    % Incase a numerical solution is not found, the program will write out
"error":
    if error > 0
        error
    end
end

% Find phase volume fractions
    eg(j)= qv(j,2)/dg(j);
    ev(j)=1-eg(j);

% Reset average conservative variables in cells with area included in the
variables.

    qv(j,1)=qv(j,1)*area(j);
    qv(j,2)=qv(j,2)*area(j);

end % end of loop

% Below we find the phase velocities by combining the
% conservative variable defined by the mixture momentum equation
% with the gas slip relation.
% At the same time we try to summarize the gas volume in the well. This
% also measure the size of the kick.

gasvol=0;

for j=1:nobox

% The interpolations introduced below are included

```



```

% to omit a singularity in the slip relation when the gas volume
% fraction becomes equal to 1/K. In addition, S is interpolated to
% zero when approaching one phase gas flow. In the transition to
% one phase gas flow, we have no slip conditions (K=1, S=0)

ktemp=k;
stemp=s;

k0(j) = ktemp;
s0(j) = stemp;

% Interpolation to handle that (1-Kxgasvolumefraction) does not become
zero
if ((eg(j)>=0.6) & (eg(j)<=0.8))
    xint = (eg(j)-0.6)/0.2;
    k0(j) = 1.0*xint+k*(1-xint);
elseif(eg(j)>0.8)
    k0(j)=1.0;
end

% Interpolate S to zero in transition to pure gas phase
if ((eg(j)>=0.6) & (eg(j)<=1.0))
    xint = (eg(j)-0.6)/0.4;
    s0(j) = 0.0*xint+s*(1-xint);
end

%
if (eg(j)>=0.9)
    % Pure gas
    k1(j) = 1.0;
    s1(j) = 0.0;
else
    %Two phase flow
    k1(j) = (1-k0(j)*eg(j))/(1-eg(j));
    s1(j) = -1.0*s0(j)*eg(j)/(1-eg(j));
end

help1 = dl(j)*ev(j)*k1+dg(j)*eg(j)*k0;
help2 = dl(j)*ev(j)*s1+dg(j)*eg(j)*s0;

vmixhelp1 = (qv(j,3)/area(j)-help2)/help1;
vg(j)=k0(j)*vmixhelp1+s0(j);
vl(j)=k1(j)*vmixhelp1+s1(j);

```

```

    % Variable for summarizing the gas volume content in the well.
    gasvol=gasvol+eg(j)*area(j)*dx;

end

% Old values are now set equal to new values in order to prepare
% computation of next time level.

po=p;
dlo=dl;
dgo=dg;
vlo=vl;
vgo=vg;
ego=eg;
evo=ev;
qvo=qv;
dlandinold = dlandinnew;           %for inlet
prandinold = prandinnew;
vlrandinold = vlrandinnew;

dgrandinold = dgrandinnew;
vgrandinold = vgrandinnew;
egrandinold = egrandinnew;

dlandoutold = dlandoutnew;         %for outlet
prandoutold = prandoutnew;
vlrandoutold = vlrandoutnew;
egrandoutold = egrandoutnew;
dgrandoutold = dgrandoutnew;
vgrandoutold = vgrandoutnew;

% % Here we calculate the increase in pitgain. This has been deactivated
% since it is sufficient to use the volgas variable.
%   pitrate = (dl(nobox)*ev(nobox)*vl(nobox)*area(nobox))/rho0-...
%               inletligmassrate/rho0; %m3/s %NNNNNNNNNNBBBBBBBBBBBBBBBB
%   pitvolume=pitvolume+pitrate*dt; %m3

% Section where we save some timedependent variables in arrays.
% e.g. the bottomhole pressure. They will be saved for certain
% timeintervalls defined in the start of the program in order to ensure
% that the arrays do not get too long!

if (counter>=nostepsbeforesavingtimedata)
    printcounter=printcounter+1;
    time % Write time to screen.

```

```

% Outlet massrates (kg/s) vs time
liquidmassrateout(printcounter)=dl(nobox)*ev(nobox)*vl(nobox)*area(nobox);
gasmassrateout(printcounter)=dg(nobox)*eg(nobox)*vg(nobox)*area(nobox);

% Outlet flowrates (lpm) vs time
liquidflowrateout(printcounter)=liquidmassrateout(printcounter)/...
    rho(liq(P0,T0))*1000*60;
gasflowrateout(printcounter)=gasmassrateout(printcounter)/...
    rho(gas(P0,T0))*1000*60;

% Hydrostatic and friction pressure (bar) in well vs time
hyd(printcounter)=sumhyd/100000;
fric(printcounter)=sumfric/100000;

% Volume of gas in well vs time (m3). Also used for indicating kick
% size

volgas(printcounter)=gasvol;

% Total phase masses (kg) in the well vs time
massgas(printcounter)=gasmass;
massliq(printcounter)=liqmass;

% pout defines the exact pressure at the outletboundary!
pout(printcounter)=(p(nobox)-0.5*dx*...
    (dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g-
dx*0.5*fricgrad(nobox))/100000;

% pin (bar) defines the exact pressure at the bottom boundary
pin(printcounter)=
(p(1)+0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g+0.5*dx*fricgrad(1))/100000;

% Pressure in the middle of top box (bar).
pnobox(printcounter)=p(nobox)/100000; %
tempbott(printcounter)=temp(1)-273; % temperature at bottom - fixed
% pitgain(printcounter)=pitvolume; Use volgas for kick size.

% Time variable
timeplot(printcounter)=time;

counter = 0;

end
end

% end of stepping forward in time.
% Printing of resultssection

countsteps % Marks number of simulation steps.

```

```

% Plot commands for variables vs time. The commands can also
% be copied to command screen where program is run for plotting other
% variables.

```

```

toc,
e = cputime-t

```

```

% Plot bottomhole pressure
plot(timeplot,pin)

```

```

% Show cfl number used.
disp('cfl')
cfl = al*dt/dx

```

```

figure(1)
plot(timeplot,pin);
grid on
title('pin')
%plot(timeplot,hyd)
%plot(timeplot,fric)
%plot(timeplot,liquidmassrateout)
%plot(timeplot,gasmassrateout)
figure(2)
plot(timeplot,volgas);
grid on
title('kick volume')
% plot(timeplot,liquidflowrateout)
%plot(timeplot,gasflowrateout)
figure(3)
plot(timeplot,massgas)
grid on
title('Mass gas')
% axis([0 1000 750 900])
% plot(timeplot,massliq)
%plot(timeplot,pout)
%plot(timeplot,pnobox)

```

```

%Plot commands for variables vs depth/Only the last simulated
%values at endtime is visualised

```

```

figure(4)
plot(vl,x);
title('Liquid velocity')
figure(5)
plot(vg,x);,
title('Gas velocity')
figure(6)
plot(eg,x);
grid on
title('Lambda')
% plot(x,p/100000);
%plot(dl,x);
%plot(dg,x);

```