Hydrodynamic modelling of free-surface flows with application to leaky barriers

Shannon Charis Leakey

School of Engineering

Newcastle University

A thesis submitted for the degree of Doctor of Philosophy

December 2022

Abstract

Leaky barriers are styled after natural accumulations of large woody debris. Communities and consortia have been installing them in watercourses with the intention of decreasing flood risk downstream. However, there is a lack of quantitative evidence on their impact at both small and large scales. To address the evidence gap, this thesis uses a fluid dynamics approach, combining hydraulic modelling, Computational Fluid Dynamics (CFD) and physical modelling in a flume.

First, leaky barriers are inserted into a 1D hydraulic model for large-scale modelling by modifying a Godunov-type scheme. At the location of a leaky barrier, fluxes from the barrier discharge equation replace the Riemann solver fluxes. Results compare well with 55 flume experiments, but this method only works when the barrier resembles a well-understood hydraulic structure, such as the weir and gate combination considered here. Different types of barriers require more fundamental investigations.

The majority of the thesis is about such small-scale modelling with CFD, with a focus on extending the CFD software OpenFOAM. A new boundary condition and post-processing tool allow the volume-of-fluid solver `interFoam` to model open-channel flows more effectively, and this solver is applied to recreate four flume experiments. Finally, a new CFD solver is developed, using a Godunov-type scheme and artificial compressibility. The free surface is captured automatically by the Riemann solver, and the pressure gradient discontinuity at the free surface is captured automatically by a novel reconstruction method. While the results compare well with analytical and experimental benchmark tests, the solver does not reliably converge on 3D unstructured meshes.

Once the convergence problem has been overcome, future research will apply the new CFD solver to model different designs of leaky barriers, and the resulting new knowledge can then be used in 1D hydraulic models to model networks of barriers across a catchment.

Acknowledgements

A big thank you to my supervisory team for consistently encouraging, supporting, and guiding me throughout this project. I would specifically like to thank Caspar Hewett for providing detailed feedback on my writing, Vassilis Glenis for being unwaveringly optimistic about my new solver, and Paul Quinn for being the nature-based solutions guru. Caspar and Vas, I will miss our Tuesday morning meetings! I also appreciate that you cared about my wellbeing, especially during the pandemic, which wasn't an easy time for anyone.

Many thanks to Sven Berendsen for the limitless supply of both programming advice and cake, James Mckenna for the Riemann solver chats and proofreading, and Oscar Ryans for helping with the flume experiments.

Finally, thank you to my family for supporting me and my church family for making Newcastle my home. I couldn't have done this without you.

Publications

The majority of the text in this thesis is taken verbatim from the following articles and proceedings papers. Note that the first five paragraphs of the *Water* paper were written by C.J.M.H. so these are not included in the thesis.

Chapters 1–2

- Leakey, S., Hewett, C.J.M., Glenis, V. and Quinn, P.F. (2020). Modelling the impact of leaky barriers with a 1D Godunov-type scheme for the shallow water equations. *Water*, 12(2):371. doi:10.3390/w12020371

Chapters 3–4

- Leakey, S. (2019). Inlets, outlets, and post-processing for modelling open-channel flow with the volume of fluid method. In H. Nilsson (ed.), *Proceedings of CFD with OpenSource Software*. doi:10.17196/OS_CFD

- Leakey, S., Hewett, C.J.M., Glenis, V. and Quinn, P.F. (2020). Hydrodynamic modelling of leaky barriers with OpenFOAM. In Uijttewaal, W., Franca, M.J., Valero, D., Chavarrias, V., Arbós, C.Y., Schielen, R., and Crosato, A. (eds.), *River Flow 2020: Proceedings of the 10th Conference on Fluvial Hydraulics*, pages 1319–1323. CRC Press. doi:10.1201/b22619

- Leakey, S., Hewett, C.J.M., Glenis, V. and Quinn, P.F. (2022). Investigating the behaviour of leaky barriers with flume experiments and 3D modelling. In Gourbesville, P. and Caigneart, G. (eds.), *Advances in Hydroinformatics*. Springer. doi:10.1007/978-981-19-1600-7_60

Chapter 5

- Leakey, S., Glenis, V. and Hewett, C.J.M. (2022). A novel Godunov-type scheme for free-surface flows with artificial compressibility. *Computer Methods in Applied Mechanics and Engineering*, 393:114763. doi:10.1016/j.cma.2022.114763

Chapter 6

- Leakey, S., Glenis, V. and Hewett, C.J.M. (2022). Artificial compressibility with Riemann solvers: Convergence of limiters on unstructured meshes. *OpenFOAM Journal*, 2:31–47. doi:10.51560/ofj.v2.49

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

BDF Backward Differencing.

CFD Computational Fluid Dynamics.

CFL Courant-Friedrichs-Lewy.

CSF Continuum Surface Force.

DIC Diagonal Incomplete Cholesky.

DILU Diagonal Incomplete Lower-Upper.

DNS Direct Numerical Simulation.

FCT Flux Corrected Transport.

FDIC Fast Diagonal Incomplete Cholesky.

GAMG Geometric-Algebraic Multi-Grid.

GIS Geographical Information Systems.

GPU Graphics Processing Unit.

HLL Harten-Lax-van Leer.

HLLC Harten-Lax-van Leer Contact.

HPC High-Performance Computing.

HRU Hydrological Response Unit.

LES	Large Eddy Simulation.
LWD	Large Woody Debris.
MAC	Marker and Cell.
MPI	Message Passing Interface.
MULES	Multidimensional Universal Limiter for Explicit Solution.
MUSCL	Monotonic Upstream-centered Scheme for Conservation Laws.
NFM	Natural Flood Management.
PBiCG	Preconditioned Bi-Conjugate Gradient.
PCG	Preconditioned Conjugate Gradient.
PISO	Pressure-Implicit with Splitting of Operators.
PLIC	Piecewise Linear Interface Calculation.
RAF	Runoff Attenuation Feature.
RAS	Reynolds-Averaged Simulation.
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations.
SLIC	Simple Line Interface Calculation.
SPH	Smooth Particle Hydrodynamics.
SST	Shear Stress Transport.
TVD	Total Variation Diminishing.
VOF	Volume of Fluid.
WENO	Weighted Essentially Non-Oscillatory.

Chapter 1. Introduction

This thesis is about leaky barriers—small, nature-based flood defences—and ways to model them at small and large scales. In particular, this thesis proposes using high-resolution shock-capturing Godunov-type schemes at both scales. The mathematics will appear from Chapter 2 onwards. First, let us consider why leaky barriers should be modelled at all, let alone hydrodynamically.

1.1 Leaky barriers as nature-based flood defences

1.1.1 Nature-based solutions

Traditionally, floods have been kept at bay using large engineered structures such as dams, diversion canals, and concrete flood walls as in Figure 1.1. However, a new class of flood management is emerging, called nature-based solutions (Nesshöver et al., 2017), working with natural processes (Environment Agency, 2012), catchment systems engineering (Hewett et al., 2020), or Natural Flood Management (NFM) (Forbes et al., 2015). Inspired by nature, although perhaps not natural (Lane, 2017; Nesshöver et al., 2017), these

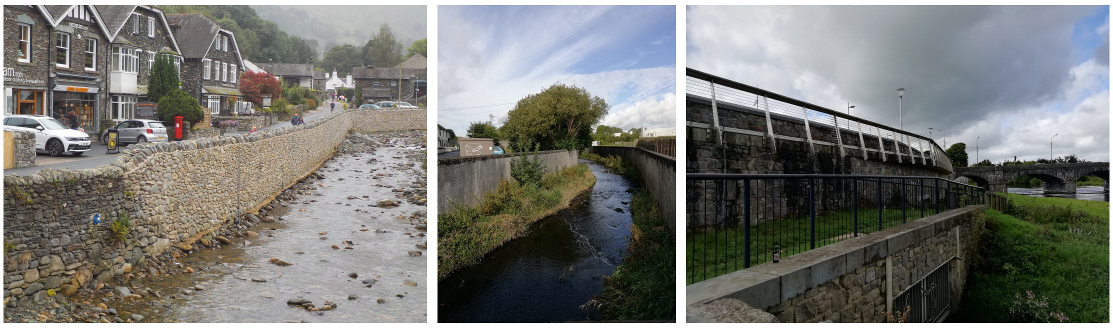

Figure 1.1: Flood walls at Glenridding in Cumbria, Ballymena in Co. Antrim, and Mallow in Co. Cork.

measures provide an alternative to holding water behind large engineered structures. From temporary storage ponds to riparian planting, from re-meandering of rivers to grip blocking, the idea is to slow runoff and store water upstream, thereby reducing the likelihood of flooding downstream. While such schemes generally need to be used in conjunction with traditional defences to protect from extreme events (Environment Agency, 2012), they are much cheaper than building ever-higher flood defences. They also provide wider benefits for society and the environment (Defra, 2005) such as enhancing biodiversity, water supply, and carbon sequestration, although these wider benefits are difficult to quantify (Iacob et al., 2014). Unsurprisingly, following Sir Michael Pitt's recommendation "to achieve greater working with natural processes" (Pitt, 2008, p.415), there has been increased interest around these nature-based flood defences, with at least 136 schemes having been installed in the UK, comprising 56 different types of feature (Nicholls and Hankin, 2015). They are popular, even praised by the former Secretary of State for the Environment as being "pioneering" (Gove, 2018), but the science is lagging behind public perception.

In particular, most of the existing evidence concerns small events in small catchments (Dadson et al., 2017; Ngai et al., 2017). Of course, minor floods are more frequent than extreme ones, and small catchments are easier to instrument than large ones. This means that any statements about the effect of nature-based solutions on large floods are based on largely untested assumptions. It may not be a simple case of scaling up from small catchments to large catchments. There are suggestions that this could cause peaks from different sub-catchments to synchronise, worsening the overall peak (Pattison et al., 2014; Dadson et al., 2017). Moreover, there are few baseline data to compare with post-intervention data (Dadson et al., 2017; Ngai et al., 2017). This makes it difficult to determine the effect of interventions on flood severity. These problems are at least partly due to the relatively recent nature of the increased interest in nature-based solutions (Addy and Wilkinson, 2019). However, a surge of evidence is expected to emerge over the coming years. For example, three NERC-funded research projects looking specifically at NFM are currently taking place in the UK and the Environment Agency is currently updating its comprehensive evidence base (Burgess-Gamble, 2020).

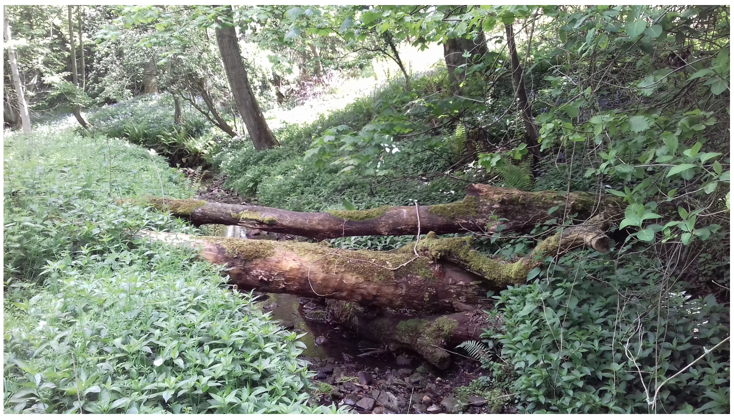

Figure 1.2: LWD in the Belford catchment in Northumberland. Credit: Caspar Hewett.

1.1.2 Runoff attenuation features

The Runoff Attenuation Feature (RAF) is one category of nature-based flood defence, providing opportunities to slow down flows and create temporary storage across a catchment, as well as filter runoff to improve water quality (Nicholson et al., 2012; Quinn et al., 2013). Online RAFs include bunds built across overland flow pathways, barriers installed in ditches, and Large Woody Debris (LWD) placed within streams, while offline RAFs are structures such as temporary storage ponds filled by flow diverted from streams (Quinn et al., 2013). This thesis focuses on one class of RAF: leaky barriers in channels. Leaky barriers mimic natural LWD (Figure 1.2) or beaver dams found in rivers, and indeed they can be built to resemble these features. However, they can also be more engineered. This variety in form is also reflected in the variety of terms used to refer to leaky barriers: woody debris, leaky dams, engineered log jams.

One leaky barrier by itself will have limited impact on a flood event; hundreds are needed across a catchment to make a practical difference. The problem is that we do not know what designs, quantities or locations are optimal. Nonetheless, while there is a distinct lack of understanding in how leaky barriers behave, both individually and in aggregate (Ngai et al., 2017), this has not stopped thousands being enthusiastically installed across the UK and globally. For example, since 2016, the local community of the Calder Valley, UK has created 619 leaky barriers in its river catchment (Slow the Flow Calderdale, 2022).

1.2 Observing leaky barriers in action

Most of the literature on leaky barriers is about LWD, and little of this relates to flooding (Ngai et al., 2017). Instead, it focuses on what would be considered wider benefits of nature-based solutions, like ecology and geomorphology, as well as formation, stability, and transport. Much of it is based in North America. A rich history of research in the Pacific Northwest stemmed from the work of Frederick J. Swanson in the late 1970s (Ruiz-Villanueva and Stoffel, 2017) and, in a comprehensive review of the last 50 years of literature on LWD, Wohl (2017) cited 361 studies, mostly on perennial rivers in Pacific Northwest conifer forests. In another review, Lo et al. (2021) note that LWD installed primarily to reduce flooding should have the same wider benefits as LWD installed in other contexts, and Ngai et al. (2017) remark that the literature rates LWD as generally having a positive impact in this wider setting. However, it is the hydraulic and hydrological effects that matter for flooding and, as outlined below, these are poorly understood. Specifically, there is limited research on how leaky barriers work in isolation from riparian planting, and very few different types of barrier have been tested (Barlow et al., 2014; Ngai et al., 2017).

1.2.1 Natural LWD dams

This section focuses on the measured hydraulic and hydrological impacts of natural LWD dams specifically. Beaver dams, engineered barriers, and flume studies are discussed in Sections 1.2.2, 1.2.3, and 1.4. The impact of LWD will depend on the form of the debris accumulation, as well as its level of debris entrapment (Dudley et al., 1998). While the "large" in LWD is generally understood to be at least "0.1m diameter and 1.0m length" (Linstead and Gurnell, 1999, p.ii), it is the size of the debris relative to the channel that determines its form. This was conceptualised by Wallerstein et al. (2001), whose classification system is illustrated in Figure 1.3.[1] As the relative width of the debris increases, there is more of an obstruction to the flow, until the debris spans the whole channel width and can no longer sit directly on the river bed. Kitts (2010, pp.132) notes how this overlaps with the classification of Gregory et al. (1985), with the exception of

[1]Throughout this thesis, figures use the "Dark2" colour palette by Brewer (2022).

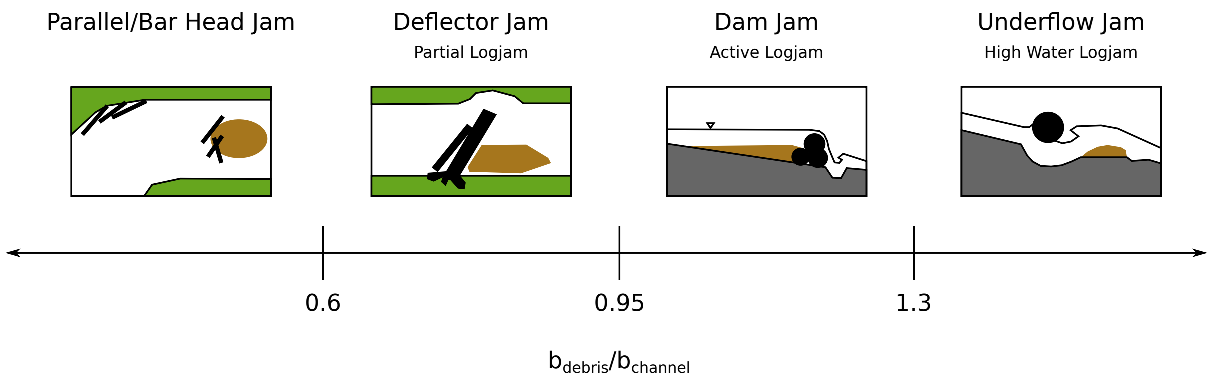

Figure 1.3: Wallerstein et al. (2001) jam classification overlapping with Gregory et al. (1985) logjam classification. Flow is from left to right. Based on Wallerstein et al. (2001, pp.1266).

complete logjams (not pictured) that span the whole channel width but do not induce a change in water level due to their porosity.

A popular way to study the hydraulic impact of LWD in the field and in models is with Manning's n. Manning's n is the roughness coefficient in the formula

$$Q = \frac{1}{n} A R^{2/3} S_f^{1/2} \tag{1.1}$$

where Q is the discharge, A the cross-sectional area, R the hydraulic radius, and S_f the friction slope (Akan, 2006). It can be back-calculated by measuring Q, A, R, and S_f in the field, and is often a critical parameter in hydrological and hydraulic models.

In many studies, the effect of LWD has been quantified by calculating the value of Manning's n at the reach level. This can be compared to reaches without LWD and reaches with different types of LWD. For example, at a test channel in Oklahoma, Dudley et al. (1998) found that the average Manning's n decreased from 0.127 to 0.093 with the removal of LWD. In the New Forest, Gregory et al. (1985) found that clearing two active logjams reduced Manning's n from 0.516 to 0.292. Also in the New Forest, Linstead and Gurnell (1999) found 10 reaches with active/complete logjams had a value of 0.963, the six with partial logjams 0.634, and the nine with no LWD 0.286. Again in the New Forest, Kitts (2010) found dam jams were the most rough at 1.42, deflector jams in the middle at 0.32, and underflow jams the least rough at 0.27. Therefore, the effect depends on the type of debris accumulation within the reach. However, these values will not be generalisable to all LWD accumulations under all flow rates (Addy and Wilkinson, 2019).

Indeed, there is evidence that the roughness effect decreases with increasing flow rate. Again in the New Forest, Gregory et al. (1985) found that Manning's *n* at one active logjam decreased from 1.0155 to 0.3072 as the flow rate increased and, in the Forest of Dean, Linstead and Gurnell (1999) found that the effect of LWD on Manning's *n* decreased as the flow rate increased at two reaches. This suggests that quantifying the effect of LWD with Manning's *n* is fundamentally flawed, based on data not physics. However, while Dudley et al. (1998) admit that its use is indeed theoretically questionable when most of the resistance comes from form drag instead of bed roughness—as is the case with LWD accumulations—they justify it based on "its familiarity and the lack of practical alternative" (p.1190). Therefore it is no surprise that representing LWD like this is widely used when including LWD in hydrological and hydraulic models, even decades later. This is discussed in more detail in Section 1.3.3.

Other hydraulic methods are more theoretically sound. For instance, Shields and Gippel (1995) partitioned the flow resistance at streams in Tennessee and New South Wales into bed, bend and LWD components. Similarly, Manga and Kirchner (2000) partitioned the shear stress at a stream in Oregon into bed and LWD components, finding that LWD accounted for half the total roughness despite covering "less than 2% of the surface area of the stream" (p.2378). In Illinois, Daniels and Rhoads (2003) found that LWD at the outside of a meander bend deflects the flow to the inside of the bend, making the helicoidal motion more intense. They then went on to show that, unsurprisingly, the impact of LWD on the flow around a meander depends on the form of the LWD (Daniels and Rhoads, 2004), and so it is difficult to generalise. In New York state, Manners et al. (2007) found that the number of wood pieces making up a deflector jam affected its porosity and, in turn, its impact on velocities around the jam. Indeed, if porosity is ignored, then the drag force is overestimated by 10–20%. However, it is not clear how these methods could be applied in hydraulic or hydrological models, unlike Manning's *n*.

Research on the direct impact of natural LWD dams on hydrology and flooding is similarly limited. In the New Forest, Gregory et al. (1985) estimated that LWD slowed down peak flows of 1.0m^{3}s^{-1} by 10 minutes, 0.3m^{3}s^{-1} by 35 minutes, and 0.1m^{3}s^{-1} by over 100 minutes. However, it is unclear where they got these numbers from. They had done a regression of travel time and discharge along a stretch of channel containing 93 logjams that were not removed. The regression showed that higher flows had shorter travel times than lower flows, but there were no baseline data, and so there is no way of knowing if the travel time would have been any different without LWD. Therefore, the numbers seem to have no basis. Despite this, they are often referred to in the literature, for example, in the Environment Agency evidence base (Ngai et al., 2017, p.34). Also in the New Forest, Sear et al. (2010) focused on how LWD causes an interaction between the channel and the floodplain. They found that high bedloads encourage aggradation upstream of LWD, resulting in a raised bed and thus raised water levels. Meanwhile, if there are low bedloads, then the raised water levels come from blocking and resistance at the LWD. Again, this detailed knowledge about the New Forest is site-specific, and thus is unlikely to be applicable to other locations within the New Forest and elsewhere.

Clearly, the impact of LWD on hydraulics and hydrology is far from a settled matter. Moreover, while Manning's *n* is the most popular way to quantify the hydraulic impact of LWD, it is theoretically questionable, and other approaches are limited by the difficulty of measuring in the field. This points us towards a hydrodynamic modelling approach, but first we must consider other forms of leaky barriers: beaver dams and engineered barriers.

1.2.2 Natural beaver dams

Although beaver dams have similarities with LWD accumulations, they have been investigated as their own category of obstruction. In Belgium, Nyssen et al. (2011) found that high flows decreased and low flows increased with the reintroduction of beavers. They standardised the discharge with precipitation, but the time series data were only available for six years after the dams were established, which is not long enough to make statements about extreme events. In Devon, Puttock et al. (2017) found that flood waves were attenuated through a dammed reach, but there were no baseline data, and so it is unclear how much attenuation would have been expected without the beaver dams. In contrast, Puttock et al. (2020) did use baseline data for four sites across the UK and came to the same conclusion. Westbrook et al. (2020) found that beaver dams even attenuated a large

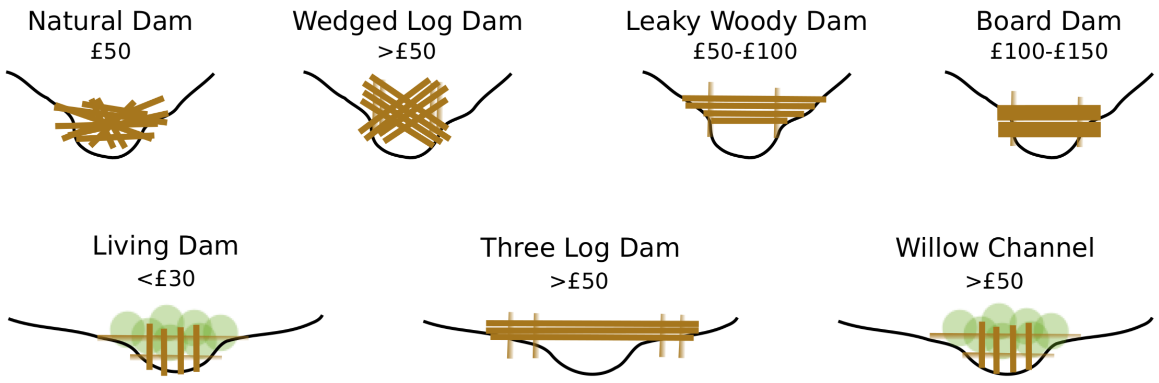

Figure 1.4: Engineered barriers and their approximate costs, based on Thames21 (2021).

flood event in the Canadian Rocky Mountains. Beavers have recently been reintroduced to Essex as part of a flood management scheme (Environment Agency, 2018). It would be interesting to see how this project develops as there is potential for beaver dams to attenuate floods, but it is clear that the strength of any results will be limited by the length of the observed record, which is a wider problem with the research on nature-based solutions (Dadson et al., 2017; Ngai et al., 2017).

1.2.3 Engineered barriers

Leaky barriers intentionally installed for flood attenuation purposes can have different forms and materials, as illustrated by Figure 1.4. They can directly mimic natural LWD dams or appear more engineered. They can even include living vegetation. As with natural LWD dams, different barriers will have different impacts on the flow. For instance, a wedged log dam will likely cause less temporary storage than a board dam due to its increased leakiness, and a three log dam stretching out onto the floodplain will only impact overbank flows due to its height (Thames21, 2021).

An early case study for assessing the impact of leaky barriers is the Belford catchment in Northumberland (Figure 1.2). The community at risk of flooding was too small to qualify for expensive flood walls, and so a group of organisations came together to explore alternative options. They built 30 RAFs, including LWD barriers and online ditch barriers (Wilkinson et al., 2010; Nicholson et al., 2012; Quinn et al., 2013). The results were more qualitative than quantitative, especially for the online features; however, Quinn et al. (2013) do provide some useful practical guidelines. For example, to comply with Environment Agency regulations, it is best to put online features in headwater streams where the upstream contributing area is smaller than 2km^2. Moreover, they found that it is difficult to estimate the temporary storage created by the barriers, which makes it difficult to understand how the barriers behave in transient flood scenarios.

While leaky barriers are popular, they were not commonly monitored as recently as 2017 (Dadson et al., 2017; Ngai et al., 2017). This is unfortunate because there is no way of evaluating their performance in the field without extensive monitoring. In a relatively early study in Germany, Wenzel et al. (2014) took a controlled, scientific approach to assessing the performance of leaky barriers in flood events. They installed entire trees lengthways into a creek and released water from a rafting pond upstream, simulating a 3.5-year flood event many times. With these barriers, the average hydrograph peak was delayed and slightly flatter but not attenuated.

Fortunately, much more research has emerged in the last few years, at least in the UK. In the South Pennines, Shuttleworth et al. (2019) installed and monitored the effect of blocking moorland gullies with stone and timber dams, undertaking a before-after-control-intervention study. While peat re-vegetation increased the lag time by 106% and decreased the peak flow by 27% on its own, combining peat re-vegetation with gully blocking increased the lag time by a further 94% and decreased the peak flow by a further 24%. This demonstrates the great potential of leaky barriers to slow down flows in combination with other measures, albeit at the micro-catchment scale. In Gloucestershire, Taylor and Clarke (2021) investigated two leaky barrier sites and a control over five flow rates. The barriers had little impact on low flows but induced a backwater effect at high flows. In Bolton, Norbury et al. (2021) found that, following the installation of five willow leaky barriers on a 130m stretch of floodplain, peak flows were reduced by an average of 27.3%, but only by 4.3% during storm events.

Moreover, there is ongoing research into this topic. For instance, monitoring is currently being undertaken by Powell et al. (2021) at a stretch of river in Berkshire with over 30 leaky barriers and Muhawenimana et al. (2021) at a stretch of river in Shropshire with

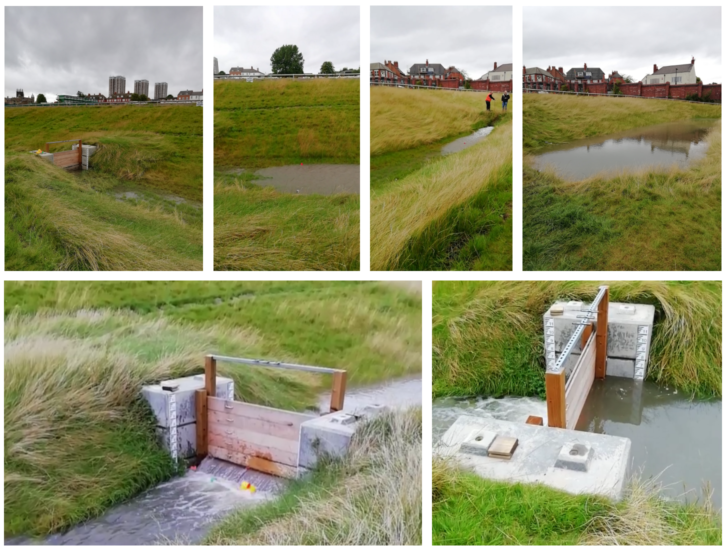

Figure 1.5: Leaky barrier experiment in the swale at Newcastle University's National Green Infrastructure Facility. Flow from right to left.

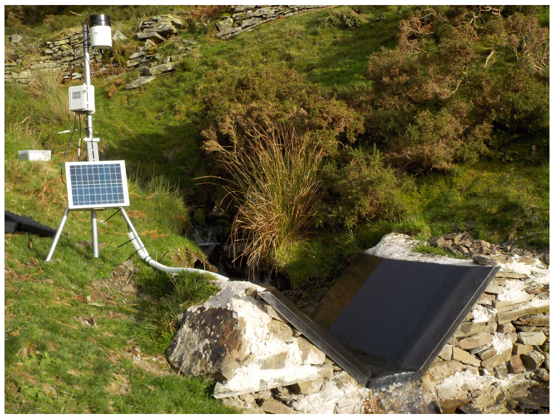

Figure 1.6: Measurement flume for Q-NFM project at Lorton, Cumbria.

105 leaky barriers. Meanwhile, at Newcastle University's National Green Infrastructure Facility, Starkey et al. (2020) are thoroughly monitoring the flow along a functional swale with two leaky barriers, as pictured in Figure 1.5. In Cumbria, the Q-NFM project is taking a slightly different approach by gauging 18 micro-catchments with flumes as in Figure 1.6 (Hankin et al., 2021). They are comparing the flow upstream and downstream of interventions, which makes it easier to isolate the effects of individual features, and investigating temporary storage by comparing the breakthrough curves of a tracer (Chappell, 2018). Explicitly, temporary storage can be calculated from the difference in tracer concentration (and thus mass) between the upstream and downstream breakthrough curves (Woessner,

2020).

Despite this recent surge of monitoring documented in the literature, results are limited to small events in small streams, a recurring problem with field studies of nature-based solutions (Dadson et al., 2017; Ngai et al., 2017). We need to know how the interventions perform in large flood events at the catchment scale. This takes time as we have to wait until a large event happens. Moreover, if we rely only on monitoring alone, then the barriers have to be installed before they can be understood. This has the potential to be a poor use of time and money, but at least it is better than installing barriers without monitoring them.

1.3 Modelling leaky barriers in aggregate

If a community or consortium is going to spend their time and money installing hundreds of leaky barriers across a catchment, they will want to put the right number of barriers in the right places. In fact, if they choose incorrectly, they might even make floods worse by synchronising the peaks from different parts of the river network (Pattison et al., 2014). Modelling is a good way to explore these issues before the barriers are installed, ideally allowing us to optimise placement and avoid synchronisation of flood peaks. However, current modelling methods fall short of this goal. Hankin et al. (2016) set out the options for modelling nature-based flood attenuation features, and these are very limited. Indeed, there is no "specially designed tool" for modelling leaky barriers (Ngai et al., 2017, p.46) and developing such a tool "is a key area for development" (Pinto et al., 2019, p.362). Consequently, researchers have had to guess how the barriers behave so that they can be included in models. Figure 1.7 shows the approaches taken in order of increasing complexity. Note that this is an overlapping but different categorisation to the review paper of Addy and Wilkinson (2019), who did not consider indirect comparison or the storage volume method.

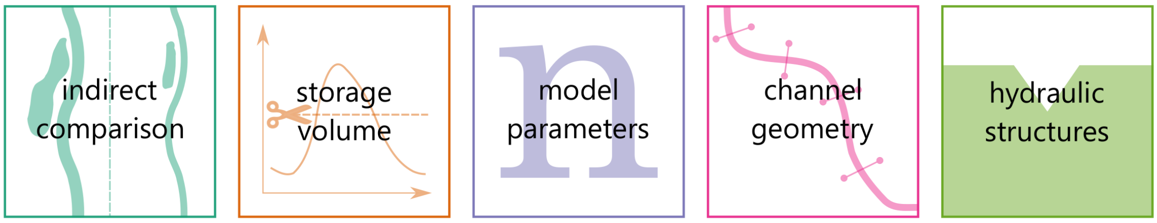

Figure 1.7: Current methods for including leaky barriers in hydrological and hydraulic models.

1.3.1 Indirect comparison

The simplest method is to circumvent the modelling problem by not modelling the barriers themselves, but rather using their measured effect to drive another model. The Slowing the Flow Partnership (2016) did this to find out how Pickering's leaky barriers slowed the flow during a storm on Boxing Day 2015. There were 167 wooden dams, 187 heather dams, 44ha of woodland, and a flood storage area with a capacity of 120,000 m^3. They used the measured peak flows from the 2015 event and similar rainfall events in 2008 and 2009 to drive a 2D hydraulic model, which gave simulated pre- and post-intervention inundation areas. As was expected, the inundation area was smaller with leaky barriers upstream. However, their rainfall total came from one rain gauge, which ignores spatial variability. Moreover, they attributed 50% of the attenuation to the flood storage area and 50% to the other measures (including the leaky barriers), which could only ever be a rough approximation. A more direct modelling approach would be more reliable and more useful for planning.

1.3.2 Storage volume

The flood-excess volume method (Bokhove et al., 2018) is the simplest way to model leaky barriers directly. Here, the volume of temporary storage provided by the barriers is estimated, and then it is removed from the flood hydrograph. For example, the storage volumes behind online and offline barriers have been estimated with 1D hydraulic models (Nisbet et al., 2015) and hydrological GIS tools (Norbury et al., 2018). This approach assumes the barrier is empty before the flood event, and so would not be effective at modelling multi-peak events. Also, it neglects any losses through the barrier walls, thus modelling only "static storage" instead of "kinematic storage" (Goudarzi et al., 2021), and it does not consider the location of the barriers. These problems mean that the modelled effects of the interventions are likely to be overestimates. Ideally, there would be a way to include leaky barriers in the hydrological and hydraulic models themselves.

1.3.3 Model parameters

A popular method for including leaky barriers in a model is to change parameters at the location of the barriers. Recall from (1.1) that the parameter Manning's n is used as a measure of roughness, and the impact of leaky barriers is often quantified in the field by back-calculating the value of Manning's n. Consequently, leaky barriers can be modelled by changing the value of Manning's n at the location of the barrier in a hydrological or hydraulic model.

Hydrological models are based on the conservation of mass, and do not include the conservation of momentum. Odini and Lane (2010) and Dixon et al. (2016) represented online barriers as increased values of Manning's n in a hydrological model. To model whole catchments quickly and in an exploratory way, they looked only at overland flow, effectively assuming that the catchment is already saturated before the flood event. However, the efficiency of the model meant that many model runs could be completed with different configurations of barrier location, and so they could find out where was best to locate the features: not in the downstream reaches, lest they fall foul of the synchronisation problem. These results were then used by Nisbet et al. (2015) to place Pickering's leaky barriers. While this modelling approach is better than not doing any modelling, it would be more accurate to route the flow through channels using a hydraulic model as the conservation of momentum would also be included.

Indeed, a similar approach of changing the roughness parameter has been undertaken for 1D and 2D hydraulic models by Kitts (2010), Pinto et al. (2019), Rasche et al. (2019), Ferguson and Fenner (2020a,b), and Macchione and Lombardo (2021). However, even though there is better representation of the open-channel flow in hydraulic models than hydrological models, the representation of the leaky barriers is just as basic. Consider the study by Rasche et al. (2019), who compared changing the Strickler roughness coefficient ($1/n$) across the whole reach and only at the locations of the leaky barriers. Testing the results against experimental data from Wenzel et al. (2014), they found that the reach-scale modification gave better results than the more localised modification. This suggests that the roughness parameter method is fundamentally flawed, not capturing the behaviour of the leaky barriers as it should.

While changing Manning's n has been used successfully to model other nature-based solutions, such as floodplain planting (Thomas and Nisbet, 2007, 2016; Nisbet and Thomas, 2008; Metcalfe et al., 2016), it seems less suitable for leaky barriers. First, even if the parameter values come from the field as in Kitts (2010), it is inappropriate to apply them to other sites or flow conditions (Addy and Wilkinson, 2019). Grabowski et al. (2019) note that more research is needed to find out what roughness values are suitable. Moreover, an increased Manning's n will impact low flows as well as high flows, and so this would not be a good model of barriers that let low flows through unimpeded. This could be overcome by only activating the increased Manning's n value at higher flows, which seems to be the method Wolstenholme et al. (2021) are currently researching within a 2D morphodynamic model. However, an increased Manning's n will have little effect on steeper slopes (Sholtes and Doyle, 2011; Dixon et al., 2016), but a leaky barrier is going to provide an obstruction either way. Consequently, while changing model parameters is a quick and easy solution, it is not a robust one.

1.3.4 Channel geometry

A more physically-based way to model online barriers is by altering the channel geometry. There are three main ways to do this: blockages, cross-sections, and topography (Figure 1.8). The blockage method involves using functionality built into standard 1D hydraulic models, where the head loss across a blockage is calculated with the Bernoulli equation to account for contraction and expansion (Innovyze, 2021; Jacobs, 2021). This was used to model leaky barriers by Forest Research (2007), Thomas and Nisbet (2012), and Pinto et al. (2019). The second method involves modifying the cross-section in the standard 1D

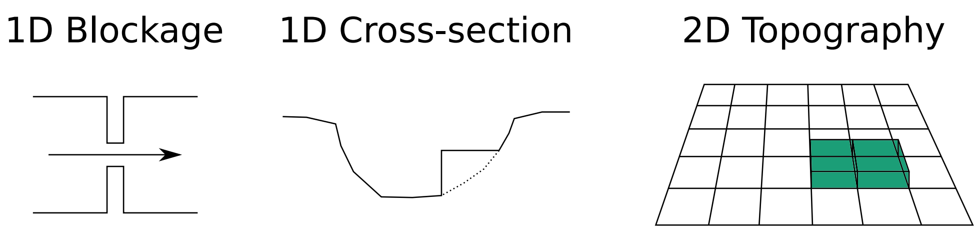

Figure 1.8: Methods for modelling leaky barriers by changing the channel geometry.

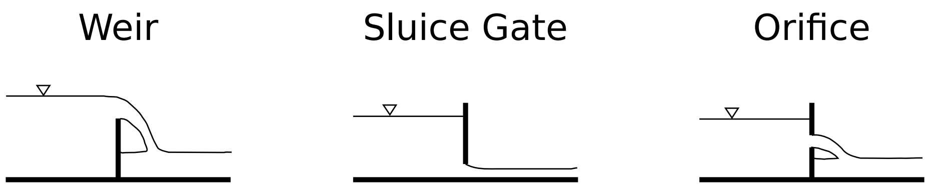

Figure 1.9: Hydraulic structures with well-known steady-state discharge equations.

hydraulic model to represent the size and location of the leaky barrier, as done by Valverde (2013). The third method is to represent leaky barriers as individual cells in a 2D hydraulic or morphodynamic model, either as solid raised topography (Hafs et al., 2014; Rasche et al., 2019; L'Hommedieu et al., 2020; Pearson, 2020) or a porous zone (Samra, 2017; Wilhelmsen et al., 2021). Representing leaky barriers this way has a different effect to changing the roughness. Indeed, while increasing the roughness reduces velocities at the location of the barrier, decreasing the cross-sectional area increases velocities (Valverde, 2013). Moreover, any leakiness through the structure itself is ignored in the non-porous methods. Again, this assumes that the barriers behave a certain way to model their impact.

1.3.5 Hydraulic structures

Finally, leaky barriers can be modelled by comparing them to well-understood hydraulic structures such as weirs, sluice gates, or orifices (Figure 1.9). This is because the Q-h relationship at these structures—an empirical equation for the discharge Q in terms of the depth h—is already known, and so can be taken to be an approximation of the Q-h relationship at a leaky barrier.

For example, Milledge et al. (2015) used weir equations to study the behaviour of different barrier designs—full brow, rectangular slot, V-notch, inverted V-notch, letterbox slot—as part of a hydrological model describing the ponded water behind each barrier.

They found that the letterbox slot performed best, and, for each type of barrier, the narrowest and deepest notches attenuated most. This strange result is because attenuation needs more water coming into the pond than leaving the pond, but the more this happens, the higher the head over the weir crest, and so the higher the discharge.

Weir equations can be combined with other hydraulic structure equations to model more complicated barriers. Keys et al. (2018) combined weir equations with orifice equations in a hydraulic model and Pearson (2020) combined weir equations with culvert equations in another hydraulic model, but the most developed approach is to combine weir equations with sluice gate equations, as pioneered by Metcalfe et al. (2015, 2017, 2018) in the catchment-scale hydrological model Dynamic-Topmodel. Dynamic-Topmodel (Metcalfe et al., 2015) lumps together similar areas into a Hydrological Response Unit (HRU) to get the overall catchment runoff. Metcalfe et al. (2017) routed this runoff into a simple diffusive wave hydraulic model, which worked by calculating the fluxes between sub-reaches using Manning's equation. Leaky barriers were modelled by changing from Manning's equation to the weir and sluice gate equations at the location of each barrier. They found that adding the barriers resulted in a delay of the flood peaks of Storm Desmond. However, the storage volume provided by the barriers could be underutilised, depending on the exact configurations of the barriers and the type of storm (single- or double-peaked). This example also demonstrates why the storage volume method, neglecting transient storage and only including static storage, is too simple. The behaviour of leaky barriers is inherently transient and needs to be modelled as such.

An identical representation of leaky barriers was used by Cabaneros et al. (2018) in their network model. However, while Metcalfe et al. (2017) considered the submergence of gates and the impacts from hillslope runoff, Cabaneros et al. (2018) did not, but they did consider other aspects of the model. For example, they found that barriers placed in the tributaries of a river pose less of a cascade failure risk than those in the main trunk. This also agrees with the advice from Quinn et al. (2013) that barriers should be placed in small headwater streams for compliance with Environment Agency regulations. Leaky barriers can also be included in hydraulic models using the same technique (Hankin et al., 2019). However, Hankin (2020) suggested that the small cell sizes required put such a constraint on the Courant number that it is not a practical method for large-scale 2D modelling.

As well as online RAFs, Metcalfe et al. (2018) introduced offline hillslope RAFs to Dynamic-Topmodel by lumping them all into one HRU. These were not online leaky barriers and the fact that every RAF was part of the same HRU meant that the synchronisation problem could not be investigated, but the study raised an important question: what is the optimal leakiness of an RAF? Features that drained in 1 hour drained too quickly to be useful in the largest storm events, while features that drained in 100 hours filled too early in the smaller events. They found that a 10-hour drainage time was best. This ties in with the findings of Milledge et al. (2015), that is, the storage ponds behind barriers should not fill up in low flows, but should quickly increase in attenuation once a certain threshold is crossed.

The hydraulic structure method has more potential to replicate the real-life behaviour of online and offline barriers than the other methods. However, we are constrained by the theoretical and empirical equations that already exist. For barriers that do not look like combinations of well-understand traditional hydraulic structures, we need more data. This could come from flumes, the field or, as Metcalfe et al. (2018, p.2602) suggest, "hydrodynamic analysis of simulated individual structures situated in realistic topographic representations."

In a recent review paper, Addy and Wilkinson (2019) recommended that the hydraulic structure method be investigated in more detail. Consequently, Chapter 2 of this thesis is about this method. It outlines a new 1D model where leaky barriers are inserted into a 1D Godunov-type scheme for the shallow water equations using internal boundary conditions. Importantly, the chapter also compares the results with the new hydraulic flume data presented in Appendix A.

1.4 Towards a fluid dynamics approach

There are limited detailed measurements of what happens at leaky barriers, and so there is no agreed-upon way to represent leaky barriers in models. The key problem with the existing modelling approaches is that they make untested assumptions about how leaky barriers behave individually to deduce how leaky barriers behave in a network. Consequently, no matter how well the background model is calibrated and validated, the most important parts of these studies—the leaky barriers themselves—are not. We must first understand how leaky barriers behave individually, and Computational Fluid Dynamics (CFD) provides an effective tool for this.

While open-channel flow is usually modelled using the shallow water equations, the complex flow around the barrier is highly three dimensional, involving both water and air. Thus, both the Navier-Stokes equations and a method for dealing with the air-water interface are required to model even the simplest leaky barriers adequately. There is some such CFD modelling of leaky barriers in the literature, specifically two main papers with associated flume and meshing studies. First, Allen et al. (2008) and Lai and Bandrowski (2014) developed methods for meshing in-channel interventions, and Allen and Smith (2012) then investigated the effect of geometrically simplifying a piece of LWD in a CFD simulation. As the geometry was simplified, the kinetic energy became more turbulent and the velocity magnitude increased. Second, Xu and Liu (2017) compared three ways to incorporate engineered logjams into a CFD model—fully resolved, porosity block, and a solid barrier—comparing the results with flume experiments by Gallisdorfer et al. (2014) and Bennett et al. (2015). They found that all three approaches gave fairly similar results downstream from the barrier, but different results at the barrier itself. Both these CFD studies used the fixed-lid approach to dealing with the free surface, where the free surface location is an input to the model, rather than an output (Bates et al., 2005). This is a major limitation as it means that the method could not be used to investigate the backwater effect of different leaky barrier designs. Moreover, they focused on LWD-style barriers rather than more engineered barriers with larger gaps.

It is important to note that CFD must be used in combination with theory and experiment, not in isolation (Anderson, 1995). While the field is not controlled enough for this, hydraulic flumes offer a suitable alternative (Friedrich et al., 2021). As mentioned above, Xu and Liu (2017) compared their results with flume experiments by Gallisdorfer et al. (2014) and Bennett et al. (2015). Other researchers have conducted flume studies without CFD modelling. For instance, Schalko et al. (2018, 2021), Okamoto et al. (2019), Follett et al. (2020, 2021), Maricar and Maricar (2020) and Piton et al. (2020) investigated the backwater effects caused by LWD, Muhawenimana et al. (2020) investigated the backwater effects caused by more structured leaky barriers, Müller et al. (2021a,b) investigated the effect of such structured leaky barriers on fish movements, Hart et al. (2020) investigated the wider hydraulic effects of porous barriers, and Spreitzer et al. (2020) used structure from motion to assess LWD accumulations. It is popular to construct the LWD models from wooden dowels; however, Perry et al. (2018) note that they behave significantly differently than LWD models constructed from natural materials. Friedrich et al. (2021) provide a full review of LWD flume studies, citing 50 studies from 1957–2021, 12 of which were published in 2020 alone. Clearly, there is currently much interest in this area of research. However, using theory and experiment in isolation from CFD means that detail and flexibility are lost.

A fluid dynamics approach combining CFD experiments and flume studies is ideal for isolating how different types of leaky barrier behave. The majority of the thesis is about this kind of detailed modelling with the Navier-Stokes equations. Chapter 3 outlines the theoretical background for CFD, Chapter 4 applies OpenFOAM (Weller et al., 1998) to model three-dimensional non-fixed-lid flow around leaky barriers, comparing with new hydraulic flume data presented in Appendix A, and Chapters 5–6 develop a new free-surface solver and work towards using it to model leaky barriers. As in Chapter 2, the numerical methods developed in Chapters 5–6 use high-resolution shock-capturing Godunov-type schemes with Riemann solvers, a focus of this research.

The ultimate goal this project works towards is to extract information from CFD modelling and insert it into hydraulic models, which could then be used to model networks of leaky barriers across a catchment accurately. This requires a combination of different types of modelling—simplified modelling with the shallow water equations, detailed modelling with the Navier-Stokes equations, and physical modelling in a flume. The combination of all three is a unique contribution of this thesis.

Chapter 2. Leaky barriers in a 1D hydraulic model

In this chapter, we investigate the hydraulic structure method for modelling leaky barriers. Away from a leaky barrier, the movement of water can be modelled by the shallow water equations. However, the shallow water equations are no longer valid in the vicinity of a leaky barrier, so we need a method for inserting the behaviour of leaky barriers at their location. We do this here using internal boundary conditions. The work in this chapter has been published in Leakey et al. (2020). Before describing the model and its application, this chapter introduces Godunov-type schemes, the numerical method used both for the shallow water equations here and for the Navier-Stokes equations in Chapters 5–6.

2.1 Introduction to Godunov-type schemes

2.1.1 A dam break

Consider water at rest behind a dam (not a leaky dam). The dam is instantaneously removed. What happens next? Properties of the 1D shallow water equations say that a smooth wave called a rarefaction will form, with its two ends propagating in opposite directions at constant speeds away from where the dam was, as illustrated in Figure 2.1. If we generalise this idea to arbitrary depths and velocities on either side of the dam, we have what is called a Riemann problem. Formally, let **U** be the vector of conserved variables which, for the shallow water equations, is

$$\mathbf{U} = \begin{bmatrix} h \\ hu \end{bmatrix} \tag{2.1}$$

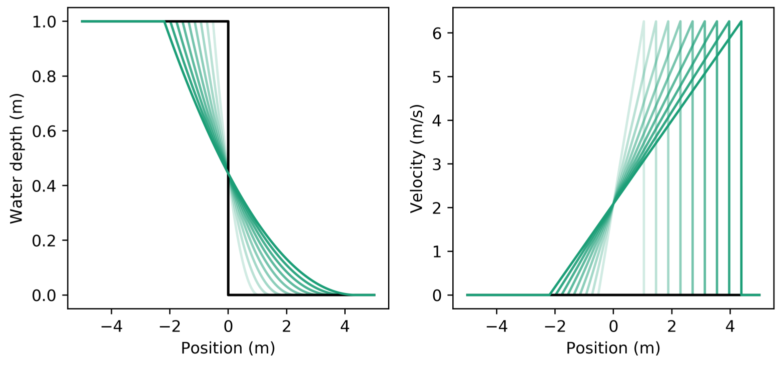

Figure 2.1: Exact solution to dam break with 1D shallow water equations. Initial conditions given by black line. Green lines are equally-spaced snapshots between $t = 0.1$s and $t = 0.7$s. Based on Toro (2001, p.114).

where h is depth and u is average velocity. The initial conditions of the Riemann problem are

$$\mathbf{U}(x,0) = \begin{cases} \mathbf{U}_L & \text{if } x < 0 \\ \mathbf{U}_R & \text{if } x \geq 0 \end{cases} \tag{2.2}$$

where $\mathbf{U}_L$ and $\mathbf{U}_R$ are constants and we seek to find $\mathbf{U}(x,t)$.

Different wave patterns can form depending on the initial conditions (Toro, 2001). First note that there are two types of nonlinear waves: rarefactions and shocks. Rarefaction waves are marked by diverging characteristics and result in smooth variations of all the variables across the wave. Shock waves are marked by converging characteristics and result in discontinuous variations of all the variables across the wave. If both sides of a Riemann problem have non-zero depth, then the resulting wave pattern will be a rarefaction wave and a shock wave, two rarefactions, or two shocks. For the 2D shallow water equations, there will also be a contact wave between the two nonlinear waves, which is marked by parallel characteristics across the wave and results in a discontinuity in the tangential velocity.

2.1.2 Riemann solvers

Riemann problems are the building blocks of a Godunov-type scheme (Godunov, 1959). Godunov-type schemes are based on the finite volume method, where variables are stored at the cell centres and advanced in time by calculating the fluxes at cell interfaces. The novelty of a Godunov-type scheme specifically is that the fluxes are calculated using a Riemann solver, the values at either side of the cell interface giving the initial conditions for the Riemann problem. The Riemann solver can be exact or approximate; popular approximate Riemann solvers include HLL (Harten et al., 1983), HLLC (Toro et al., 1994), Roe (Roe, 1981), and Osher (Osher and Solomon, 1982).

2.1.3 Capturing shocks

Since Riemann solvers include the wave structure of Riemann problems either directly or indirectly, Godunov-type schemes are shock-capturing, that is, they can deal with discontinuous solutions of the governing equations without generating spurious oscillations or excessive smearing (Toro, 2001). For the 1D shallow water equations, this means that Godunov-type schemes can capture rapidly varying flows, transcritical flows, bores, and hydraulic jumps. We only consider the 1D shallow water equations in this chapter, but we develop a Godunov-type scheme for different governing equations in Chapters 5–6, also taking advantage of the shock-capturing capabilities.

2.2 Numerical scheme

The 1D shallow water equations are a set of nonlinear, hyperbolic partial differential equations. They are derived from the Navier-Stokes equations by assuming that vertical acceleration is negligible (Toro, 2001; LeVeque, 2004). They can be written in conservative form as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{S}, \tag{2.3}$$

where the vector of conserved variables, flux vector, and source term vector are

$$\mathbf{U} = \begin{bmatrix} h \\ hu \end{bmatrix}, \mathbf{F} = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix} \text{ and } \mathbf{S} = \begin{bmatrix} 0 \\ gh(S_b - S_f) \end{bmatrix}, \tag{2.4}$$

respectively, and h is the water depth, u the depth-averaged velocity, g the gravitational acceleration, S_f the friction slope, and S_b the bottom slope.

We use a finite volume Godunov-type scheme. One splitting scheme for the inhomogeneous equations (2.3) is the explicit time-marching formula

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}(\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n) + \Delta t \mathbf{S}_i^n, \tag{2.5}$$

where $\mathbf{U}_i^n$ is the average value of $\mathbf{U}$ in cell i at time t_n (Toro, 2001).

Alternatively, we can obtain a second-order scheme using the semi-discrete method, which treats the space and time discretisations separately (Toro, 2009). We discretise in space but not time to get the ordinary differential equation

$$\frac{d\mathbf{U}_i}{dt} = -\frac{1}{\Delta x}(\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}) + \mathbf{S}_i. \tag{2.6}$$

Then we can use a second-order Runge-Kutta method to solve (2.6). Following Liang and Marche (2009), let

$$\mathbf{U}_i^* = \mathbf{U}_i^n + \Delta t K_i(\mathbf{U}_i^n) \tag{2.7}$$

where

$$K(\mathbf{U}_i^n) = -\frac{1}{\Delta x}(\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n) + \mathbf{S}_i^n. \tag{2.8}$$

This is equivalent to (2.5), but second-order accuracy in time is obtained using

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{1}{2}\Delta t(K(\mathbf{U}_i^n) + K(\mathbf{U}_i^*)). \tag{2.9}$$

To use equation (2.9), the time step Δt, interface fluxes $\mathbf{F}_{i+1/2}^n$ and $\mathbf{F}_{i-1/2}^n$, and source term $\mathbf{S}_i^n$ need to be found.

For stability, each time step Δt is obtained using the Courant-Friedrichs-Lewy condition

$$\Delta t = \text{CFL}\frac{\Delta x}{\max(|u| + a)} \tag{2.10}$$

where $a = \sqrt{gh}$ and $\text{CFL} \in (0, 1]$ is the Courant number (Toro, 2001). The value of CFL is fixed at the start of the simulation, although the first time step is set to be $\Delta t = 10^{-5}$s. This is because the particle velocity u does not contribute to the wave speed at the start of a simulation when its initial condition is zero. However, the velocity will instantly increase drastically in simulations such as dam breaks, meaning that a first time step based on a particle velocity of zero could be too large for stability (Toro, 2009).

The HLL Riemann solver (Harten et al., 1983) is used to estimate the flux through the cell interface. Suppose the conserved variables to the left of a given interface are $\mathbf{U}_L$ and those to the right are $\mathbf{U}_R$. Then the flux through the interface is approximated by

$$\mathbf{F} = \begin{cases} \mathbf{F}_L & \text{if } 0 \leq S_L \\ \frac{S_R \mathbf{F}_L - S_L \mathbf{F}_R + S_L S_R (\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} & \text{if } S_L \leq 0 \leq S_R \\ \mathbf{F}_R & \text{if } S_R \leq 0 \end{cases}, \tag{2.11}$$

where S_L and S_R are approximate wave speeds in the Riemann problem, estimated here using the adaptive method of Glenis et al. (2018). In this method, an initial guess is given for the depth in the star region using a two-rarefaction approximation:

$$h_0 = \frac{1}{g}\left(\frac{1}{2}(a_L + a_R) + \frac{1}{4}(u_L - u_R)\right)^2 \tag{2.12}$$

If this depth h_0 is less than h_L and h_R, then the two-rarefaction approximation is kept to get $h_* = h_0$. Otherwise, a two-shock approximation is used to get

$$h_* = \frac{p_L h_L + p_R h_R + u_L - u_R}{p_L + p_R} \tag{2.13}$$

where

$$p_K = \sqrt{\frac{g(h_0 + h_K)}{2 h_0 h_K}} \tag{2.14}$$

for $K = L, R$. Now, if $h_* > h_L$, then the left wave is a shock and

$$S_L = u_L - a_L \frac{\sqrt{0.5(h_* + h_L)h_*}}{h_L}, \tag{2.15}$$

otherwise

$$S_L = u_L - a_L. \tag{2.16}$$

Similarly, if $h_* > h_R$, then the right wave is a shock and

$$S_R = u_R + a_R \frac{\sqrt{0.5(h_* + h_R)h_*}}{h_R}, \tag{2.17}$$

otherwise

$$S_R = u_R + a_R. \tag{2.18}$$

For second-order accuracy in space, the left and right Riemann states $\mathbf{U}_L$ and $\mathbf{U}_R$ are found using MUSCL-type extrapolation (van Leer, 1976) with a minmod slope limiter as in Liang and Marche (2009). At the interface between the i^{th} and $(i+1)^{th}$ cells, which is indexed by $i+1/2$, the left and right Riemann states are calculated as

$$\mathbf{U}^L_{i+1/2} = \mathbf{U}_i + \frac{\Delta x}{2}\nabla\mathbf{U}_i, \tag{2.19}$$

$$\mathbf{U}^R_{i+1/2} = \mathbf{U}_{i+1} - \frac{\Delta x}{2}\nabla\mathbf{U}_{i+1} \tag{2.20}$$

with the gradient $\nabla\mathbf{U}_i$ limited by

$$\nabla\mathbf{U}_i = \text{minmod}\left(\frac{\mathbf{U}_i - \mathbf{U}_{i-1}}{\Delta x}, \frac{\mathbf{U}_{i+1} - \mathbf{U}_i}{\Delta x}\right) \tag{2.21}$$

where the minmod slope limiter is defined as

$$\text{minmod}(a,b) = \begin{cases} \min(a,b) & \text{if } a,b \geq 0 \\ \max(a,b) & \text{if } a,b \leq 0 \\ 0 & \text{otherwise} \end{cases} \cdot \tag{2.22}$$

This requires two ghost cells at each boundary.

Despite the governing equations (2.3) containing source terms, only the homogeneous case was considered when finding the flux. The bed slope source term

$$ghS_b = -gh\frac{\partial z_b}{\partial x} \tag{2.23}$$

is incorporated into the time-marching formula (2.9) via equation (2.8) at the end of each time step. A central difference approximation gives

$$\mathbf{S}^n_i = \begin{bmatrix} 0 \\ -gh^n_i \frac{z_{b,(i+1/2)} - z_{b,(i-1/2)}}{\Delta x} \end{bmatrix} \tag{2.24}$$

as in Liang and Marche (2009). In contrast, the friction source term S_f is split from the homogeneous part (Toro, 2009) into an ordinary differential equation

$$\frac{dq}{dx} = -ghS_f \tag{2.25}$$

solved implicitly at the start of each time step with

$$q^{n+1} = q^n - \Delta t gh\frac{u|u|n^2}{R^{4/3}} \Big/ \left(1 + 2\Delta t\, g\frac{|u|n^2}{R^{4/3}}\right) \tag{2.26}$$

as in Liang and Marche (2009), where R is the wetted perimeter. This is stable for high values of Manning's n, which may be useful when comparing different ways to model leaky barriers.

2.3 Internal boundary condition

Suppose there is a leaky barrier located exactly at the interface between the cells i and $i+1$, that is, at the interface $i+1/2$. The interface flux $\mathbf{F}^n_{i+1/2}$ needed for the time-marching formula (2.9) has two components: the mass flux $hu = q$ and the momentum flux

$$hu^2 + \frac{1}{2}gh^2 = \frac{q^2}{h} + \frac{1}{2}gh^2. \tag{2.27}$$

Unless the water is shallow enough to flow unimpeded below the barrier, these fluxes cannot be found using the Riemann solver (2.11)—there is a leaky barrier obstructing the flow and so the shallow water equations (2.3) are no longer valid.

There are different ways to solve these Riemann problems at structures. In the analytical option, the equations of discharge and energy conservation at the structure are solved simultaneously to the wave structures in the rest of the domain. For instance, broad-crested weirs (Cozzolino et al., 2014a,b) and constrictions and obstructions (Cozzolino et al., 2017; Pepe et al., 2019) can be represented by a non-conservative product (Dal Maso et al., 1995) in the shallow water equations, and broad-crested weirs (Cozzolino et al., 2014a,b) and sluice gates (Cozzolino et al., 2015) can be represented by the steady-state discharge equations of the hydraulic structure. In the numerical option, the most widespread method is to calculate the numerical mass and momentum fluxes using the steady-state equations that describe the flow through the hydraulic structure in question, which seems to have originated with Zhao et al. (1994). This was used by Morales-Hernández et al. (2013), Cozzolino et al. (2015) and Cui et al. (2019) to model sluice gates, Bogoni et al. (2015) to model sharp-crested weirs, and Echeverribar et al. (2019) to model levees. Cui et al. (2019) found this method performed better than simply changing the mass source term, and Echeverribar et al. (2019) found that, even though it ignores the effect of tangential flow in 2D simulations, the effect is minimal. Another method is that of Jaafar and Merkley (2010), who solved the sluice gate equations simultaneously with the characteristic equations. No matter the method, it is important to recognise that the hydraulic structure will often have multiple operating conditions, and thus the algorithm should reflect this. For example, both Bogoni et al. (2015) and

Cozzolino et al. (2015) used the method of Lin et al. (2002) to distinguish between free and submerged flow conditions. This then allowed them to set suitable values of h and u at the internal boundary, according to whether there was subcritical or supercritical flow at either side.

We follow the method used by Cozzolino et al. (2015) for modelling sluice gates. There are two steps. First, find the mass flux q through the hydraulic structure using a well-understood steady-state discharge equation. Second, find some depth h to substitute into (2.27) to get the momentum flux. This general method will now be applied to a simple horizontal leaky barrier, using the data presented in Appendix A.1.

2.3.1 Conceptual model

As outlined in Appendix A.1, we model a simple horizontal barrier. There will always be flow under the barrier, and there may be flow over the barrier during high flows. In the Wallerstein et al. (2001) classification system, it is an underflow jam (Figure 1.3). Such barriers have been mathematically modelled by adding together the sluice gate and rectangular weir equations (Metcalfe et al., 2017; Cabaneros et al., 2018).

The physically-based models of Samani and Mazaheri (2009) suggest that there are five possible operating stages for a combined gate and weir structure, as illustrated in Figure 2.2. These are:

1. Gate operating freely, no weir overflow
2. Gate submerged by tailwater, no weir overflow
3. Gate and weir operating freely
4. Gate submerged by tailwater, weir operating freely
5. Gate and weir submerged by tailwater

In stages 1-2, the barrier acts as a sluice gate, while in stages 3-5, the barrier acts as a combined sluice gate and rectangular weir. The differences within these two groups depend on the depth of the tailwater and the location of the associated hydraulic jump.

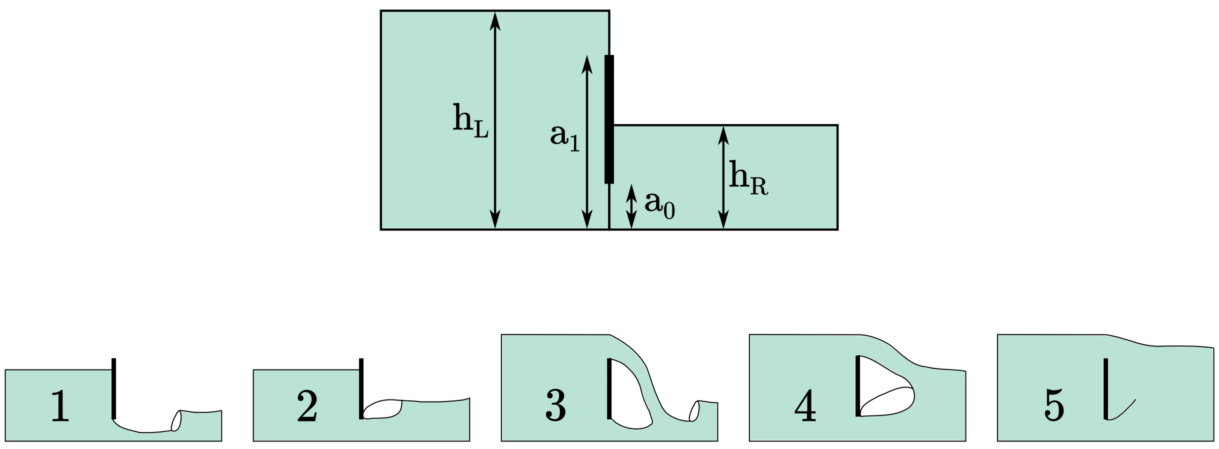

Figure 2.2: The leaky barrier model: definition sketch and five conceptual operating stages, based partially on Samani and Mazaheri (2009). White areas depict circulating water in a hydraulic jump.

Different discharge equations are used if a gate is operating freely, as in stages 1 and 3, or with its *vena contracta* submerged by a hydraulic jump, as in stages 2, 4, and 5 (Swamee, 1992). Similarly, different discharge equations are used if a weir is operating freely, as in stages 3 and 4, or if it is submerged, as in stage 5. Although in reality the gate and weir flows will interact (Negm et al., 2002, 2004; Duru, 2014), the interaction is ignored in this chapter, and the gate and weir discharges are simply added together, as in Metcalfe et al. (2017) and Cabaneros et al. (2018).

2.3.2 Numerical model

Let the depth of the base of the leaky barrier be a_0 and the top a_1, and assume that the flow is from left to right (Figure 2.2). First, assume that the gate and weir are operating freely. The gate flow is given by the classical equation

$$q_g = C_g a_0 \sqrt{2gh_L}. \tag{2.28}$$

Here, the discharge coefficient C_g is a function of the upstream depth h_L,

$$C_g = C_c \sqrt{\frac{1}{1 + C_c a_0/h_L}} \tag{2.29}$$

where C_c is the contraction coefficient, as described by Cozzolino et al. (2015). The weir flow is given by

$$q_w = \frac{2}{3}\sqrt{2g}C_w(h_L - a_1)^{3/2}, \tag{2.30}$$

where C_w is a constant. Ignoring any interactions between the gate and weir, the combined flow is given by

$$q = q_g + q_w. \tag{2.31}$$

If the gate is operating freely, then there is supercritical flow downstream of the barrier, as illustrated in stages 1 and 3 of Figure 2.2. The depth of this supercritical flow, denoted h_{sup}, is calculated iteratively using an energy balance either side of the barrier, as in Bogoni et al. (2015) and described in the following section. Following Lin et al. (2002) and Cozzolino et al. (2015), if h_R is less than the conjugate depth of h_{sup}, then there is sufficient momentum to push the hydraulic jump downstream and the gate is operating freely. Otherwise the gate is submerged (stages 2, 4, and 5). In these cases, the submerged flow equation of Henry (1950) is used to find a reduced value of C_g,

$$C_g = KC_c\sqrt{1 - \frac{h'}{h_L}} \tag{2.32}$$

where

$$\frac{h'}{h_L} = \frac{K^2 \pm \sqrt{K^4 + 4A\left(A\left(\frac{h_R}{a_0}\right)^2 - \left(\frac{h_L}{a_0}\right)K^2\right)}}{2A} \tag{2.33}$$

and

$$K = \frac{1}{\sqrt{1 - \left(\frac{C_c a_0}{h_L}\right)^2}}, A = \frac{1}{4C_c\left(1 - \frac{C_c a_0}{h_R}\right)}. \tag{2.34}$$

Here, the discharge coefficient C_g is a function of h_R as well as h_L. This is the most suitable submerged discharge equation when there are no calibration data (Sepulveda et al., 2009; Cozzolino et al., 2015); however, it does add a degree of uncertainty due to its purely theoretical nature. Additionally, if h_R is deeper than the critical depth over the weir, then the weir is also submerged (stage 5). In this case, the submerged flow equation of Villemonte (1947) is used to modify q_w, multiplying it by

$$\left(1 - \left(\frac{h_R - a_1}{h_L - a_1}\right)^n\right)^m, \tag{2.35}$$

where $m = 0.185$ and $n = 1.5$, to make q_w also a function of h_R as well as h_L.

The discharge equation (2.31) provides the mass flux $hu = q$, which is the same leaving the left-hand cell as entering the right-hand cell. The momentum flux (2.27) is more difficult as it needs a value for h as well as q, and h is different in each cell due to the discontinuity created by the leaky barrier. For the momentum flux from the left-hand cell, the depth h in (2.27) is h_L. For the flux into the right-hand cell, the depth h in (2.27) is h_{sup} for stages 1 and 3, and h_R for stages 2, 4, and 5.

2.3.3 Establishing discharge coefficients and energy losses

The internal boundary condition used in the 1D model is based on equations with empirical parameters. These parameters were found for the flume experiments presented in Appendix A.1 by analysing the measured depths immediately upstream and downstream of the barrier. First, as the slots either side of the barrier reduced the barrier width to 0.264m, the discharge equation (2.31) was multiplied by the ratio $0.246/0.294 \approx 0.84$. The discharge equation requires two more coefficients: the gate contraction coefficient C_c and the weir discharge coefficient C_w. For each of the 52 free-flow experiments, the depth measurements at $x = 11$m were plugged into the discharge equation, with this predicted discharge then compared with the measured discharge as in Figure 2.3. The coefficients $C_c = 0.70$ and $C_w = 0.70$ were found to give the best fit with the measurements, and so they were used in the numerical model.

Second, in free-flow conditions, an energy balance is required to find the depth of the supercritical flow downstream of the leaky barrier, h_{sup}. Bernoulli's equation gives

$$h_L + \frac{(q/h_L)^2}{2g} = h_{sup} + \frac{(q/h_{sup})^2}{2g} + \Delta E. \tag{2.36}$$

Note that the change in the channel bed elevation Δz_b is not included in (2.36) since we are considering values at the cell interface $i + 1/2$. The effect of the bed slope is taken into account in the source term.

If we ignore energy loss (i.e. assume $\Delta E = 0$) and solve (2.36) iteratively for h_{sup}, the water exits the leaky barrier too quickly in the 1D model, resulting in the hydraulic jump occurring further downstream than observed. However, the upstream backwater effect is

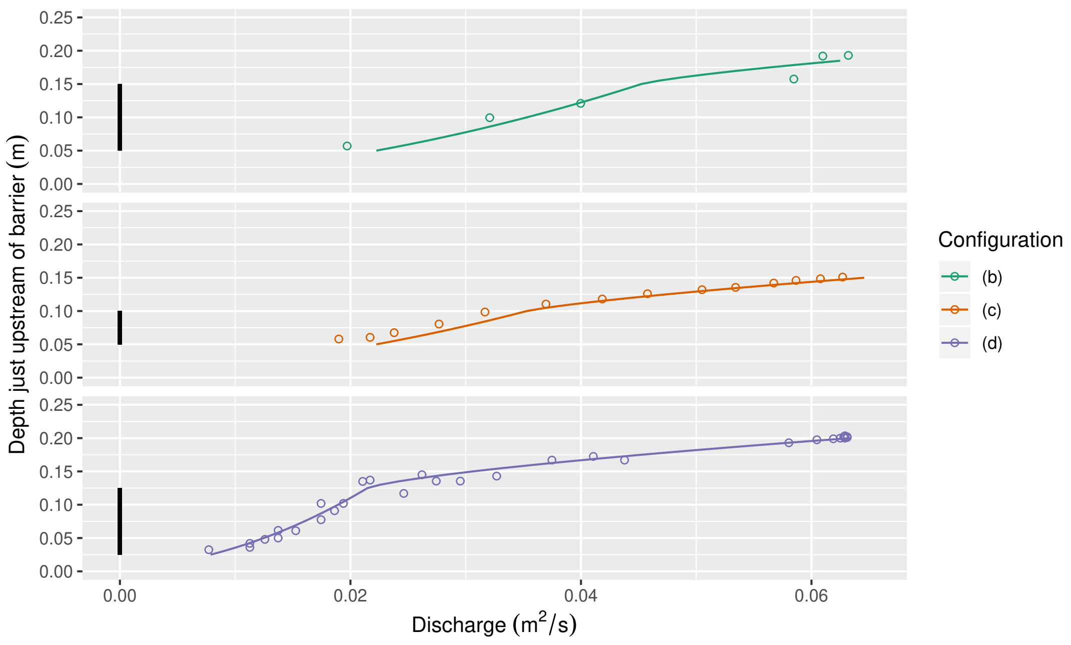

Figure 2.3: Measured depth-discharge relationship (circles) and the empirical discharge equation with $C_c = 0.70$ and $C_w = 0.70$ (lines) for each barrier configuration. Only includes free-flow scenarios where the depth is greater than the gap underneath the barrier.

captured well, indicating that the problem is not in the value of q, which determines the mass flux. This is demonstrated in Figures 2.9–2.16. Therefore, a method for estimating ΔE in (2.36) was required. Incorporating energy losses into the momentum flux like this is reported elsewhere in the literature, for example, for 1D flow around islands (Franzini et al., 2018).

To find a function for these energy losses, they were first approximated for each of the 52 free-flow experiments using Bernoulli's equation:

$$\Delta E = h_{us} + \frac{(q/h_{us})^2}{2g} - h_{ds} - \frac{(q/h_{ds})^2}{2g} - \Delta z_b, \tag{2.37}$$

where q is the measured discharge, h_{us} is the depth measurement at $x = 11$m, and h_{ds} is a depth measurement downstream of the leaky barrier, generally at $x = 12$m. In some cases h_{ds} was taken to be at $x = 11.15$m or $x = 11.30$m instead, depending on the position of the hydraulic jump and weir nappe. Note that Δz_b is included in (2.37) because the depth measurements were taken up to 0.3m away from each other. This analysis resulted in the

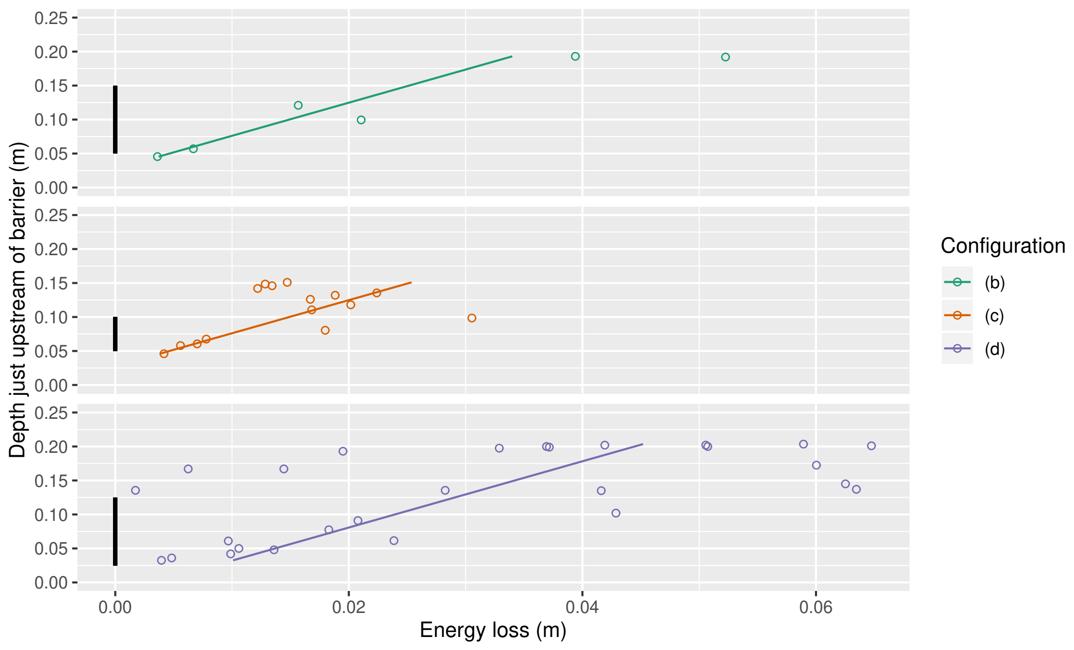

Figure 2.4: Estimated measured energy losses (circles) and the regression equation (2.38) (lines) for each barrier configuration. Only includes free-flow scenarios where the depth is greater than the gap underneath the barrier.

regression equation

$$\Delta E = 0.012 - 0.362 \cdot a_0 + 0.205 \cdot h_{us} \tag{2.38}$$

which gave $R^2 = 0.48$, plotted in Figure 2.4. This is included in the 1D model by replacing h_{us} with h_L in (2.38), and then plugging ΔE into (2.36) when finding the supercritical depth h_{sup}.

2.4 Benchmarking

The 1D scheme without a leaky barrier was tested to check if it correctly solved the shallow water equations (2.3). These benchmarks included two steady-state and one transient case. Then the 1D scheme with a leaky barrier operating as a sluice gate (stages 1 and 2) was tested against three further benchmarks. Unless otherwise noted, a Courant number of $C = 1.0$ was used.

2.4.1 Subcritical flow

To verify that the scheme can deal with topographical variations, the first benchmark was steady-state subcritical flow over a bump (Delestre et al., 2013). The domain has length $l = 25$m and the bed is flat apart from the small bump:

$$z_b(x) = \begin{cases} 0.2 - 0.05(x-10)^2 & \text{if } 8 < x < 12 \\ 0 & \text{otherwise} \end{cases}. \tag{2.39}$$

The upstream boundary condition is $q(0) = 4.42\text{m}^2/\text{s}$, the downstream boundary condition is $h(25) = 2$m, and the initial conditions are $q(x) = 4.42\text{m}^2/\text{s}$ and $h(x) = 2$m. Figure 2.5 shows that the results settle to the correct steady-state solution which, by Bernoulli's principle, is $h(x)$ where

$$h(x)^3 + \left(z_b(x) - \frac{q^2}{2gh(0)^2} - h(0)\right) h(x)^2 + \frac{q^2}{2g} = 0. \tag{2.40}$$

2.4.2 Transcritical flow with Manning's *n*

To verify that the scheme can deal with transcritical flows and friction represented by Manning's n, the second benchmark was a steady-state transcritical flow case from Delestre et al. (2013). Full derivations of the benchmark can be found in Tuoi (2008) and a programmed example in Roberts et al. (2019). The idea behind the derivation is backwards: set the location of the hydraulic jump, here $x_s = 200/3$m for a domain of $l = 100$m, form the depth $h(x)$ as a polynomial of

$$r(x) = x - x_s = \frac{x}{100} - \frac{2}{3}, \tag{2.41}$$

then do some integration to get z_b. So, for the polynomial, we have

$$h(x) = \begin{cases} \left(\frac{4}{g}\right)^{1/3} \left(\frac{4}{3} - \frac{x}{100}\right) - \frac{9x}{1000} r(x) & \text{if } x \le x_s \\ \left(\frac{4}{g}\right)^{1/3} \left(a_1 r(x)^4 + a_1 r(x)^3 - a_2 r(x)^2 + a_3 r(x) + a_4\right) & \text{otherwise} \end{cases} \tag{2.42}$$

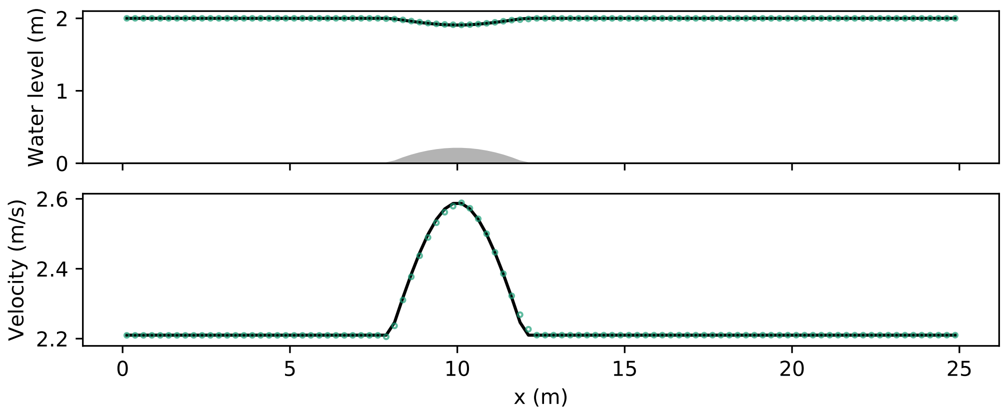

Figure 2.5: Exact (black lines) and numerical (green circles) solution for subcritical flow.

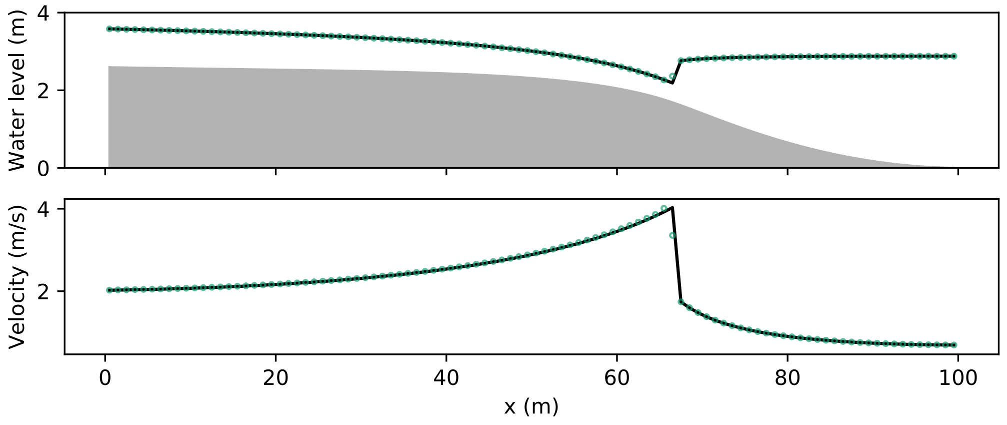

Figure 2.6: Exact (black lines) and numerical (green circles) solution for transcritical flow.

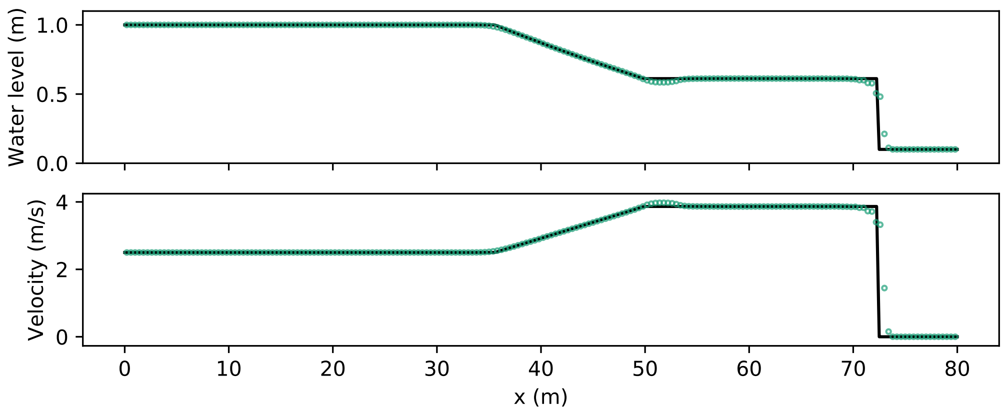

Figure 2.7: Exact (black lines) and numerical (green circles) solution for Riemann problem.

where $a_1 = 0.674202$, $a_2 = 21.7112$, $a_3 = 14.492$, and $a_4 = 1.4305$. Now note that, for steady-state cases with wide channels, the shallow water equations (2.3) reduce to

$$\begin{aligned}\frac{\partial z_b}{\partial x} &= \left(\frac{q^2}{gh(x)^3} - 1\right)\frac{\partial h}{\partial x} - S_f \qquad (2.43)\\ &= \left(\frac{q^2}{gh(x)^3} - 1\right)\frac{\partial h}{\partial x} - \frac{n^2 q^2}{h(x)^{10/3}} \qquad (2.44)\end{aligned}$$

Setting the steady-state discharge to be $q = 2\text{m}^2/\text{s}$ and Manning's $n = 0.0328$, integrate (2.44) numerically to get the value of $z_b(x)$ across the domain. This z_b was used to drive the model, with the upstream boundary condition $q(0) = 2\text{m}^2/\text{s}$ and downstream boundary condition $h(100)$ coming from (2.42). Figure 2.6 shows that, when the simulation settles to a steady state, it matches up well with the exact solution from (2.42) with only a small error in the location of the hydraulic jump.

2.4.3 Transcritical rarefaction Riemann problem

To verify that the scheme can capture shocks, the third benchmark was a Riemann problem from Toro (2001). This Riemann problem is designed to test how well a numerical method resolves a transcritical rarefaction, that is, a rarefaction that passes through a point where $u = a$. The initial conditions are

$$h(x) = \begin{cases} 1.0 & \text{if } x < 40 \\ 0.1 & \text{otherwise} \end{cases} \quad \text{and } q(x) = \begin{cases} 2.5 & \text{if } x < 40 \\ 0.0 & \text{otherwise} \end{cases} \qquad (2.45)$$

across a domain of length $l = 80$m. The exact solution can be found using the rarefaction and shock wave relations (Toro, 2001). First, solve the depth functions for h_* and u_*, determine the left and right waves (in this case a left rarefaction and right shock), then find the rarefaction wave fan. Figure 2.7 shows that the numerical method performs well. In particular, the shock speed and size is correct, there are no spurious oscillations, and there is no "rarefaction shock" (Toro, 2001, p.121) caused by the critical point.

2.4.4 Sluice gate Riemann problems

The internal boundary condition generalises the numerical method of Cozzolino et al. (2015). In the same paper, exact solutions to Riemann problems where a sluice gate separates the two Riemann states were also derived. As the leaky barrier can operate as a sluice gate (stage 1 and 2), these benchmarks were used to test the 1D model. In all three cases, the gap under the barrier was set to be $a_0 = 0.2$m. The initial conditions for discharge were set to be $q(x) = 0$ everywhere and the depth set to be

$$h(x) = \begin{cases} 1.0 & \text{if } x < 50 \\ h_R & \text{otherwise} \end{cases} \tag{2.46}$$

where $h_R = 0.002, 0.2, 0.6$m for each case respectively. Deriving the exact solution is similar to the exact Riemann solver (Toro, 2001), but complicated by the presence of the sluice gate. The approach for each of the three cases was, respectively:

- (Left rarefaction $\leftrightarrow$ free gate) $\rightarrow$ (left rarefaction $\leftrightarrow$ right shock). Here, a left rarefaction is solved simultaneously with the free gate equation (2.28) to obtain the state just upstream of the gate. Continuity gives the state just downstream of the gate, and then a left rarefaction is solved simultaneously with a right shock.

- (Left rarefaction $\leftrightarrow$ free gate) $\rightarrow$ (left shock $\leftrightarrow$ right shock). Here, a left rarefaction is solved simultaneously with the free gate equation (2.28) to obtain the state just upstream of the gate. Continuity gives the state just downstream of the gate, and then a left shock is solved simultaneously with a right shock.

- Left rarefaction $\leftrightarrow$ submerged gate $\leftrightarrow$ right shock. This is more difficult as information travels in both directions across the barrier. First, the set of possible submerged states (2.28) that can be linked to a left rarefaction is defined. This is then solved simultaneously with a right shock to get the state just downstream of the gate. By continuity, this can then be intersected with a left rarefaction.

Figure 2.8 shows that the wave structure, speeds, and strengths are accurate in all three wet-bed cases, but the first test case requires a smaller Courant number to obtain accurate results. This demonstrates the shock-capturing capabilities of the scheme under transient conditions when leaky barriers are included.

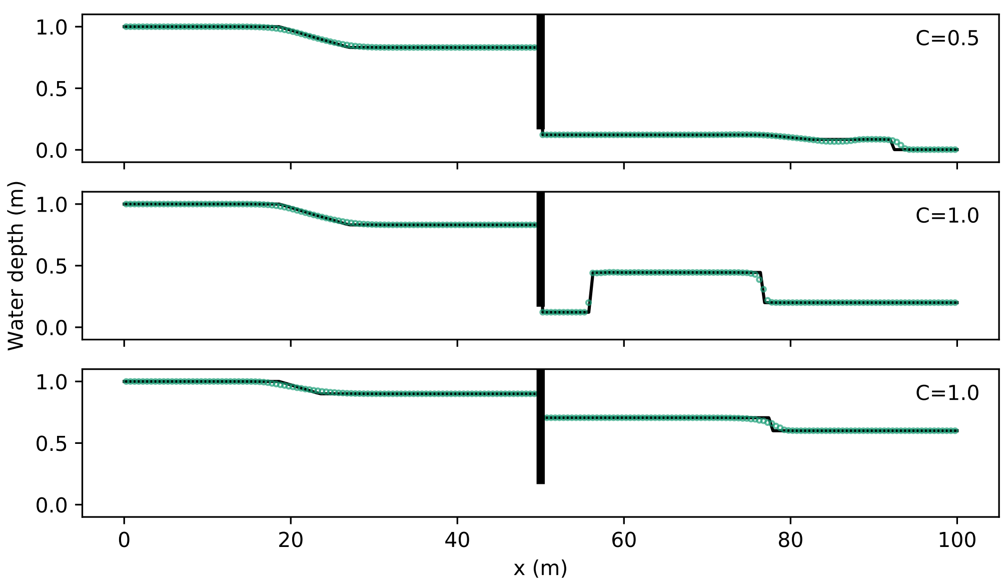

Figure 2.8: Exact (black lines) and numerical (green circles) solutions for Riemann problems at sluice gates, based on Tests 3-5 from Cozzolino et al. (2015) with $C_c = 0.61$ and $\Delta E = 0$.

2.5 Steady-state flume experiments

The 1D model should be able to recreate longitudinal profiles for each of the 55 steady-state flume scenarios from Appendix A.1. This was tested by using the parameters from Section 2.3.3 for the internal boundary condition, the measured discharge as an upstream boundary condition, and the depth measurement at $x = 17.8$m as a downstream boundary condition. As the grid size was set to be $\Delta x = 0.05$m, the leaky barrier was located at $x = 11.15$m, one centimetre away from its real placement. The Courant number was set to be $C = 1.0$ and the model run until it reached a steady state.

The water surface profiles of all 55 experiments are displayed in Figures 2.9–2.16. Note that the weir overflow is squeezed into the leaky barrier cell interface, so it is modelled even though it is not plotted. First recall that the internal boundary condition introduces two new pieces of information into the domain: the discharge through the barrier and the supercritical depth downstream of the barrier. The first piece of information travels both upstream and downstream from the barrier, whereas the second piece of information travels only downstream from the barrier. Generally, the profile upstream of the barrier

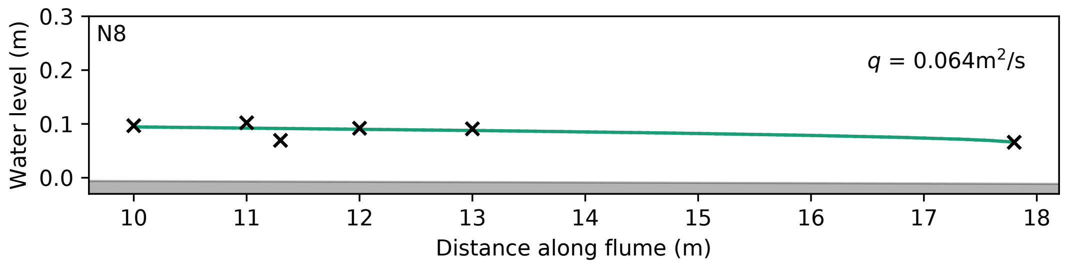

Figure 2.9: Results for configuration (a). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

is simulated well by the model, reflecting the lack of scatter in the discharge equation shown in Figure 2.3. The profile downstream of the barrier is subject to more error, reflecting the poor R^2 value for the ΔE equation (2.38) and the resulting scatter in Figure 2.4. In particular, cases where the depth is only slightly greater than the gap beneath the barrier are subject to more error, which may be attributable to how quickly the depth profile changes once the barrier is reached, and thus these cases are sometimes better represented by the model without energy losses. Despite this, the downstream profile is still captured accurately by the model in most cases. Moreover, as there were no velocity measurements across the flume, the location of the hydraulic jump is a proxy for the velocity. Its accurate placement thus suggests that the model accurately simulates the velocities.

It is interesting to note that the submerged states are recreated well, considering that the two parameters of (2.31) were calibrated for the free-flow conditions, and it is only the equations of Henry (1950) and Villemonte (1947) that were used to generalise these to submerged flow. This is important because leaky barriers are installed in series, and so the backwater effect from a downstream barrier may influence the hydraulics of an upstream barrier.

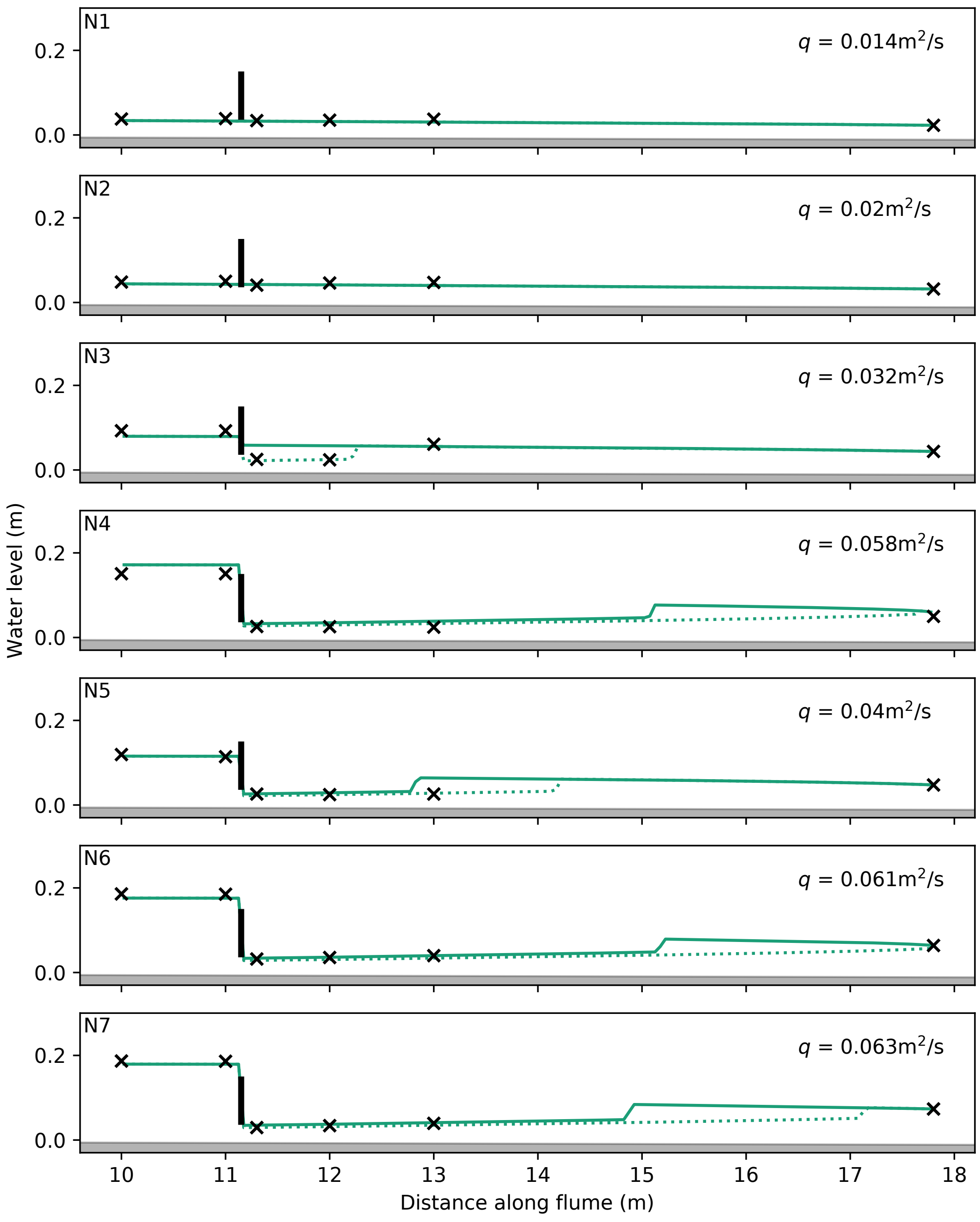

Figure 2.10: Results for configuration (b). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

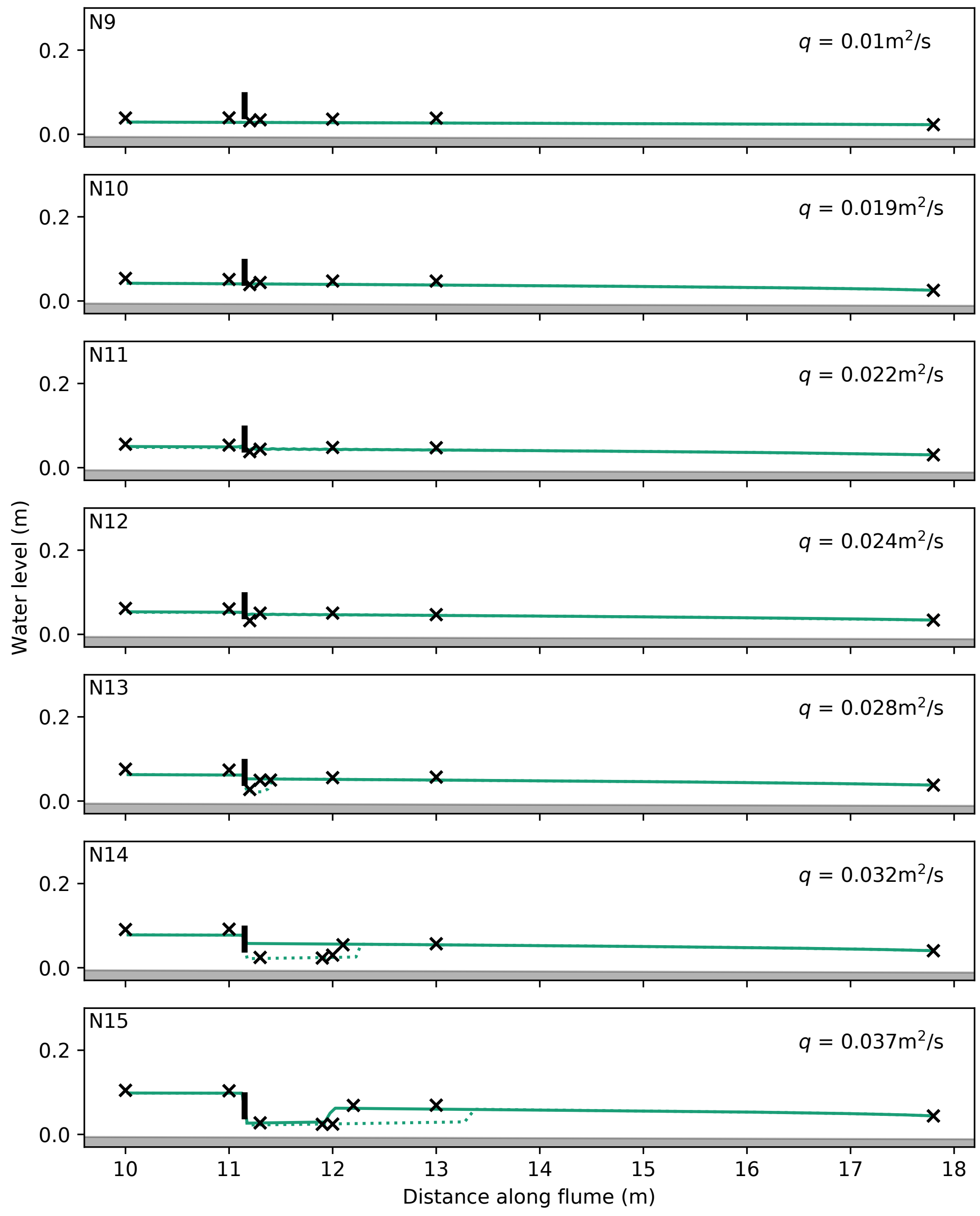

Figure 2.11: Results for configuration (c). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

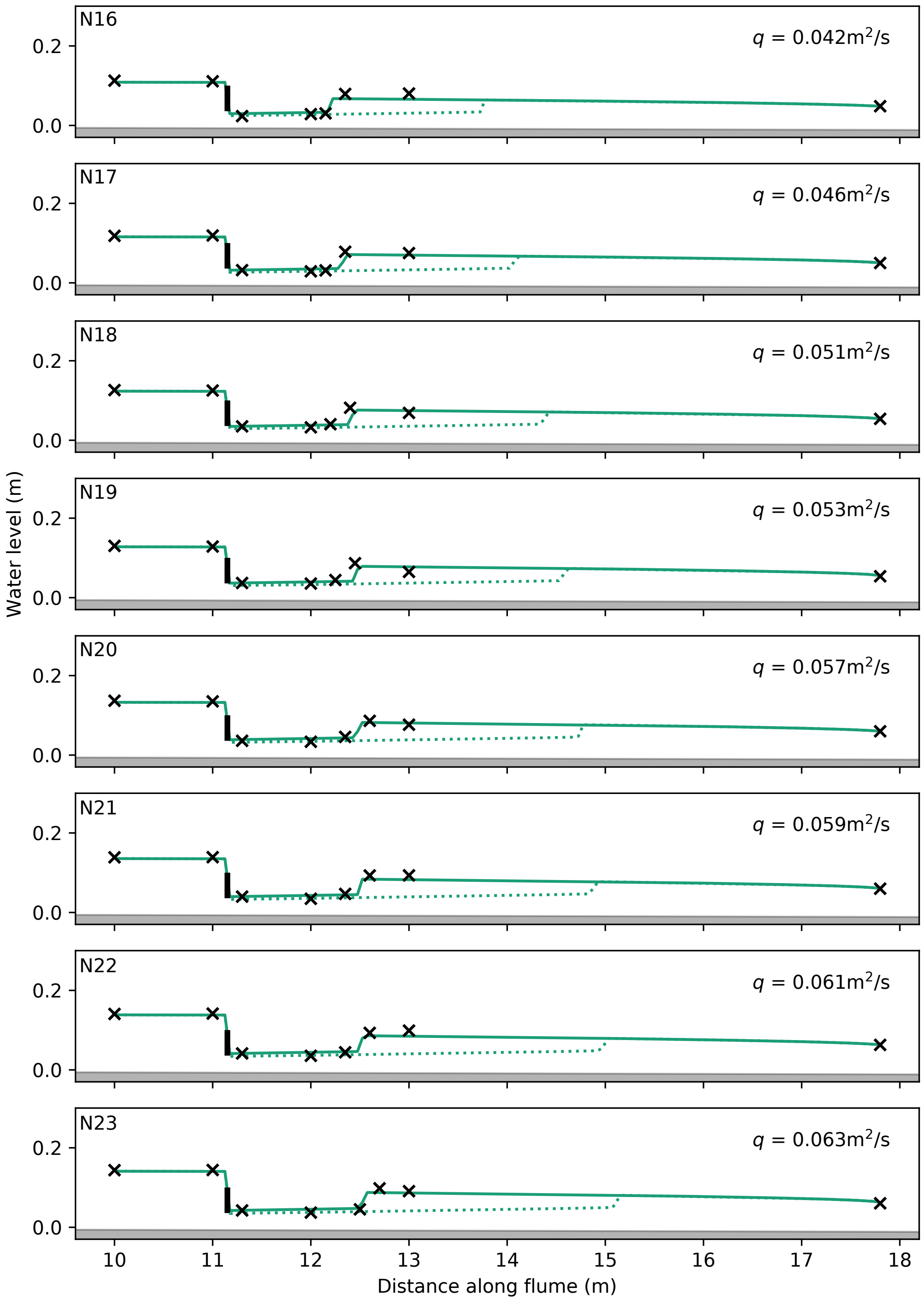

Figure 2.12: Results for configuration (c). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

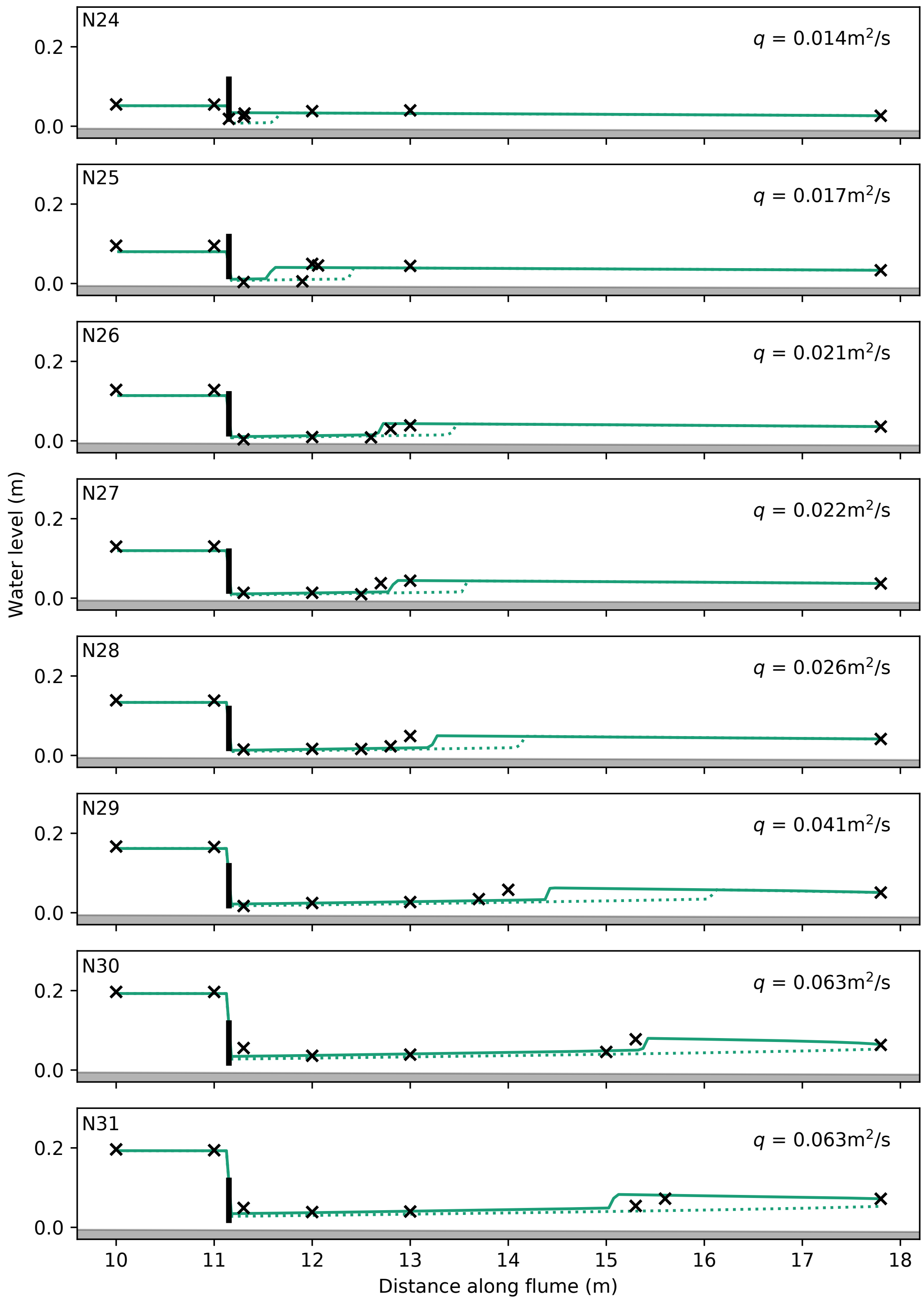

Figure 2.13: Results for configuration (d). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

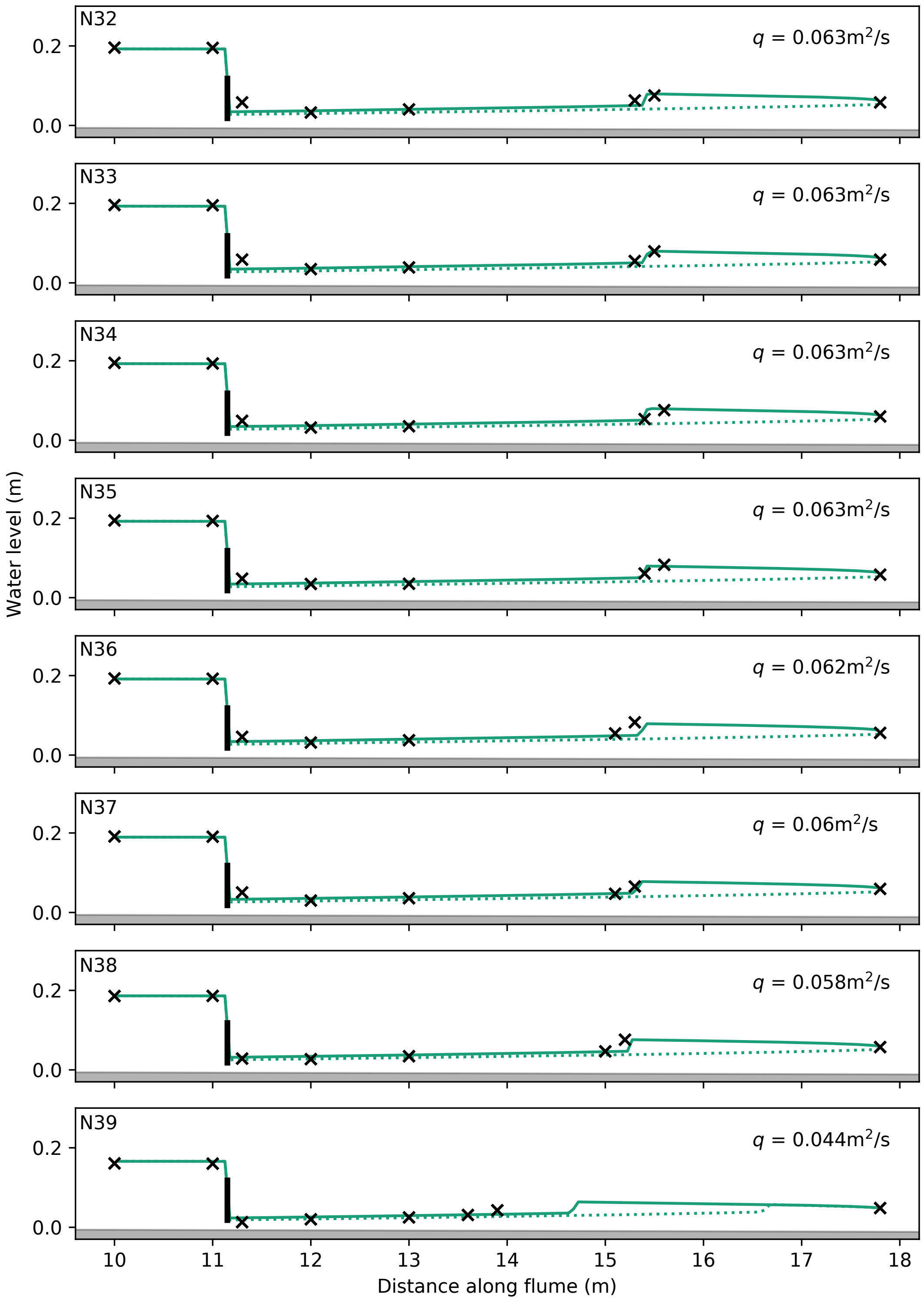

Figure 2.14: Results for configuration (d). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

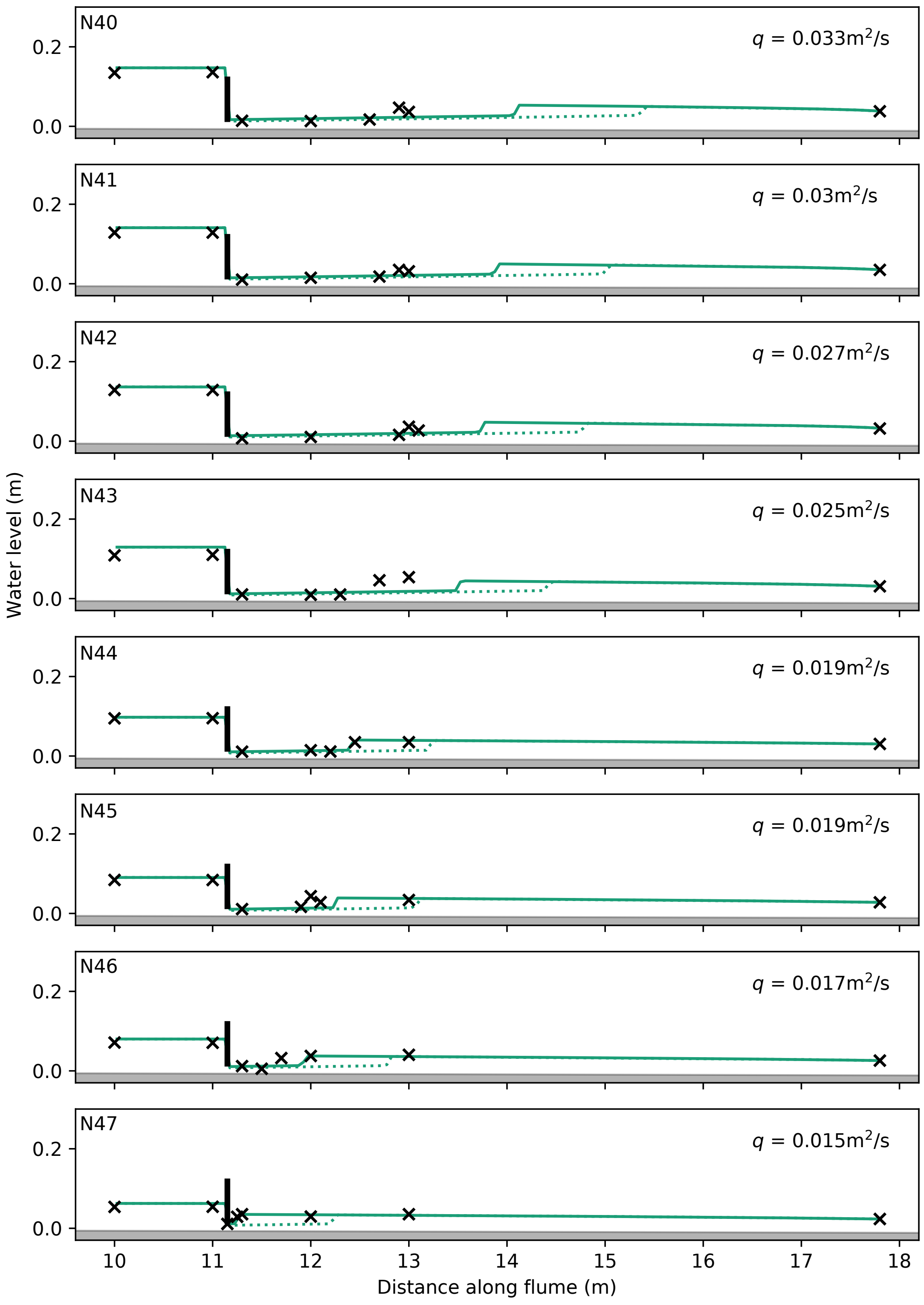

Figure 2.15: Results for configuration (d). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

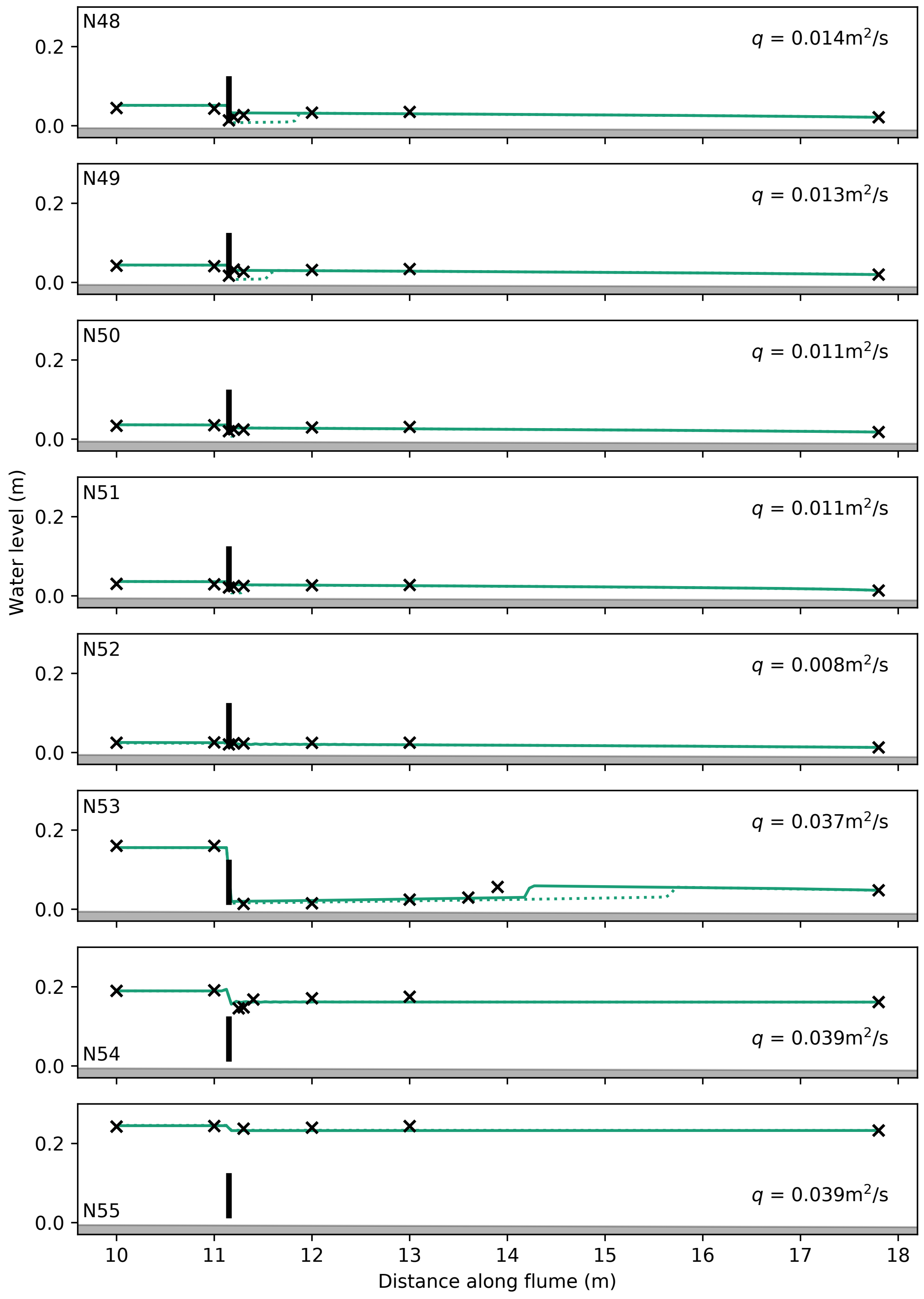

Figure 2.16: Results for configuration (d). Observations (black crosses), simulations without energy losses (dotted green line), and simulations with energy losses (solid green line).

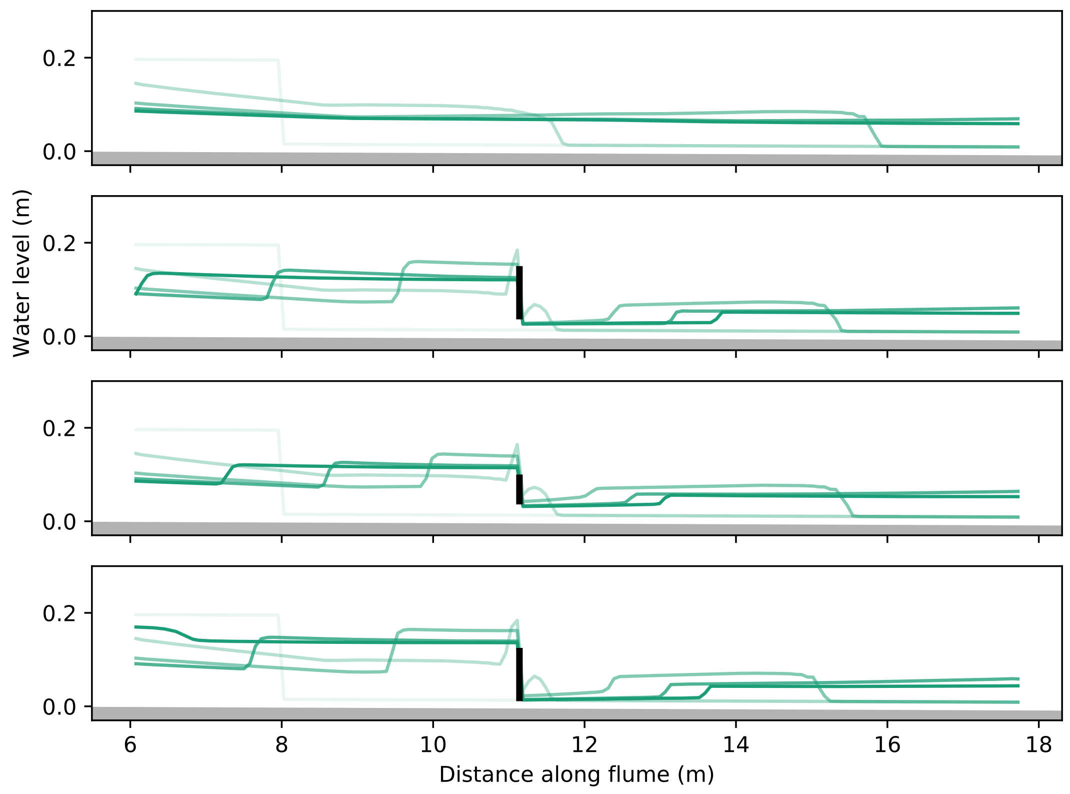

Figure 2.17: Longitudinal profiles at $t = 0.0, 1.4, 2.8, 4.2, 5.6, 7.0$s for the dam break simulations for configurations (a)-(d).

2.6 Transient experiments

Finally, to demonstrate the transient functionality of the 1D model, it was used to simulate a dam break in the flume for each configuration (a)-(d). The initial conditions were

$$h(x) = \begin{cases} 0.20 & \text{if } x < 8 \\ 0.02 & \text{otherwise} \end{cases} \tag{2.47}$$

with an upstream boundary condition of $q = 0.06\text{m}^2/s$. Again, the grid size was set to be $\Delta x = 0.05$m. The results, shown in Figure 2.17, are similar to video footage from the flume experiments. However, there is currently no quantitative data to compare them against. Nonetheless, the hydrographs in Figure 2.18 show that the bigger barriers provide more of an obstruction, which is to be expected.

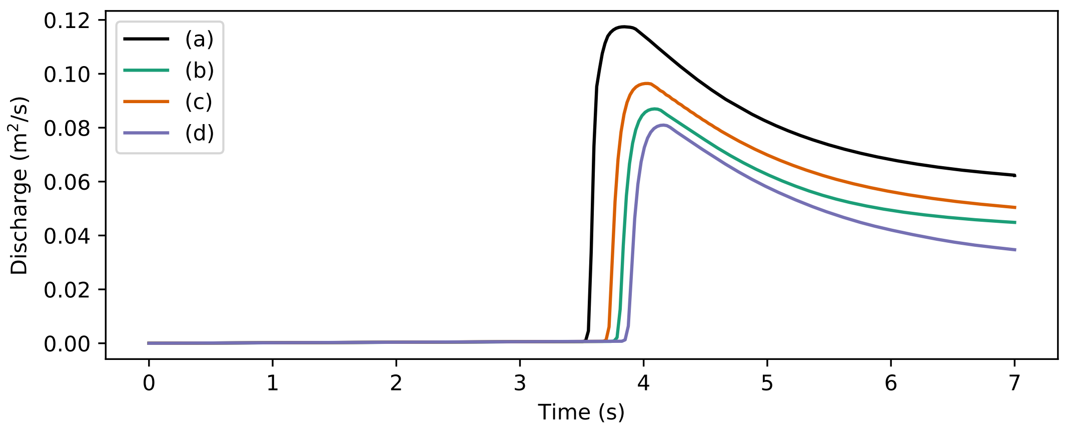

Figure 2.18: Flood hydrograph at $x = 16$m for the dam break simulations for configurations (a)-(d).

A mesh convergence study was conducted to show that $\Delta x = 0.05$m is fine enough for modelling the flow. This was done by replicating the above experiment for configuration (b) and $\Delta x = 0.2, 0.1, 0.05, 0.025, 0.0125$m. Figures 2.19 and 2.20 show that only small increases in accuracy are gained by increasing the mesh resolution beyond $\Delta x = 0.05$m.

2.7 Building upon MSc dissertation

Note that the type of model in this chapter was first applied to leaky barriers in my MSc dissertation (Leakey, 2018). This chapter builds substantially upon that work by conserving depth instead of water level in the governing equations (2.3), improving the wave speeds in the Riemann solver (2.11), incorporating energy losses (2.38) into the internal boundary condition, benchmarking against a transcritical rarefaction Riemann problem (Section 2.4.3) and three sluice gate Riemann problems (Section 2.4.4), and testing against new experimental data (Appendix A.1). Therefore, this chapter both improves the method and establishes it as robust and applicable.

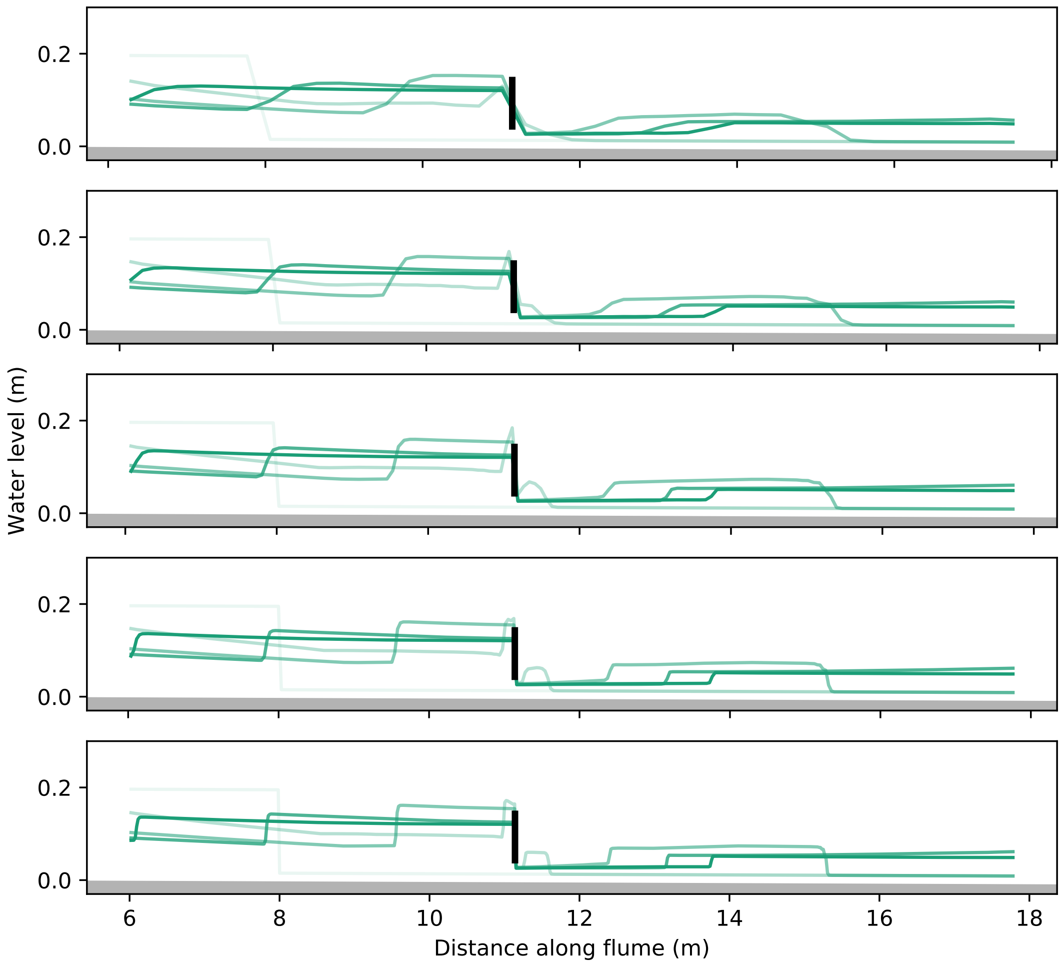

Figure 2.19: Longitudinal profiles at $t = 0.0, 1.4, 2.8, 4.2, 5.6, 7.0$s for the dam break simulations for $\Delta x = 0.2, 0.1, 0.05, 0.025, 0.0125$m.

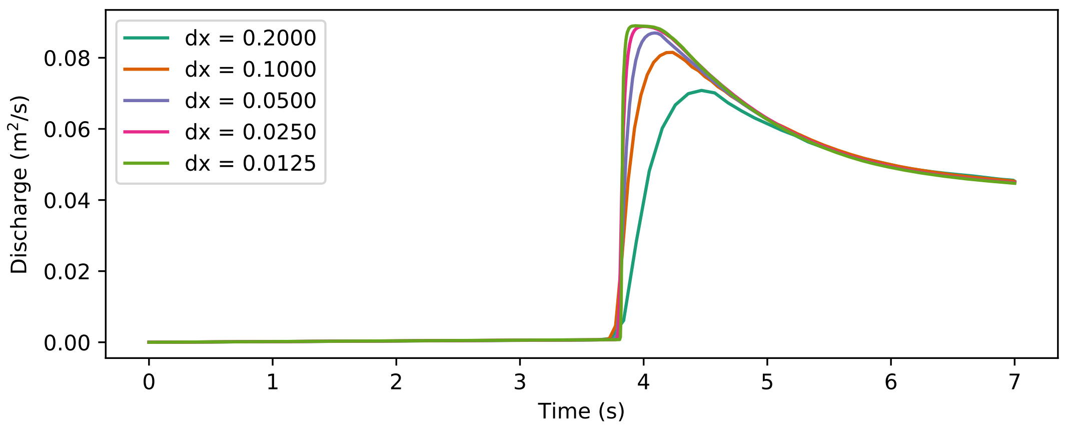

Figure 2.20: Flood hydrograph at $x = 16$m for the dam break simulations for mesh sizes $\Delta x = 0.2, 0.1, 0.05, 0.025, 0.0125$m.

2.8 Conclusion

This investigation of the hydraulic structure method addresses a specific research gap identified by Addy and Wilkinson (2019) in their review paper. The results presented show that representing leaky barriers as hydraulic structures is a valid method, provided that the barrier resembles the hydraulic structure, the energy losses are incorporated into the momentum part of the internal boundary condition, and possible submergence from downstream backwaters is considered. Given the current interest in installing multiple leaky barriers across catchments for flood prevention purposes, greater understanding of potential synchronisation issues is critical. The model presented here, which provides a first approximation of a range of natural and designed barrier types including log jams, woody debris and beaver dams, has the potential to address this issue by putting a large number of leaky barriers into a 1D hydrodynamic model of a channel network.

More generally, the approach taken in this chapter, using a combination of physical and mathematical modelling, has great potential to answer questions which urgently need addressing, as highlighted in Chapter 1. For leaky barrier designs where empirical equations already exist, such as the letterbox slot design (Milledge et al., 2015) which can be modelled as a large rectangular orifice with weir overflow, the approach presented here can be applied directly. The effect of a horizontal barrier being porous could even be incorporated, as suggested by Follett et al. (2021). However, some barrier designs do not resemble hydraulic structures with established equations (e.g. combinations of gates, weirs and orifices). A fluid dynamics approach is needed to establish discharge equations.

Chapter 3. Theory of 3D free-surface modelling

Chapter 2 introduced a method for including leaky barriers in a 1D hydraulic model. While the results compared well against flume data, the method was limited to structures for which the stage-discharge relationship is already known. For the rest of this thesis, we leave the shallow water equations behind and focus on the Navier-Stokes equations, working towards extracting information about the fundamental behaviour of leaky barriers using Computational Fluid Dynamics (CFD). This chapter covers the theoretical background required to understand the OpenFOAM modelling discussed in Chapter 4, and some of it is also relevant to the new CFD solver discussed in Chapters 5–6. First, we introduce the Navier-Stokes equations and their averaged form, before discussing turbulence modelling, discretisation methods, and—a particular focus of this work—methods for modelling free-surface flow.

3.1 Governing equations

3.1.1 Navier-Stokes equations

The Navier-Stokes equations describe how fluids flow in three dimensions. Conservation of mass is expressed by the continuity equation:

$$\underbrace{\frac{\partial \rho}{\partial t}}_{\text{rate of change of } \rho} + \underbrace{\nabla \cdot (\rho \mathbf{u})}_{\text{divergence}} = 0 \tag{3.1}$$

while Newton's second law gives the momentum equation:

$$\underbrace{\frac{\partial}{\partial t}(\rho \mathbf{u})}_{\text{rate of change of } \rho u} + \underbrace{\nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u})}_{\text{divergence}} = \underbrace{-\nabla p}_{\text{pressure}} + \underbrace{\nabla \cdot (\mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda(\nabla \cdot \mathbf{u})\mathbf{I})}_{\text{viscous stress}} + \underbrace{\rho \mathbf{g}}_{\text{body forces}} \tag{3.2}$$

where ρ is density, $\mathbf{u}$ velocity, p pressure, μ dynamic viscosity, $\mathbf{g}$ gravity, and λ second viscosity, which is not well understood but usually negligible (Versteeg and Malalasekera, 2007). Incompressible flows are governed by the constraint

$$\nabla \cdot \mathbf{u} = 0 \tag{3.3}$$

which, combined with the assumption of invariable density, renders (3.1) redundant. However, it is possible to have flows that can be modelled as incompressible with variable density e.g. free-surface flows of water and air, and so (3.1) is not redundant for all incompressible flows. Compressible flows require an energy equation and, to close the system, an equation of state. Generally, the Navier-Stokes equations have to be solved approximately using numerical methods.

An added complication is that fluid flow can exhibit turbulence, a property measured by the Reynolds number

$$\mathrm{Re} = \frac{\rho U L}{\mu} \tag{3.4}$$

where U is the characteristic velocity and L the characteristic length scale. Flows with a low Reynolds number (laminar flows) are characterised by layers of fluid that smoothly flow in almost parallel paths with minimal mixing. In contrast, in flows with a high Reynolds number (turbulent flows), the inertial forces dominate the viscous forces and there is unsteady, chaotic flow featuring an energy cascade (Versteeg and Malalasekera, 2007), so that

> Big whorls have little whorls,
> Which feed on their velocity;
> And little whorls have lesser whorls,
> And so on to viscosity.
> — Lewis Fry Richardson (Pope, 2000, p.183)

The scales of this turbulence mean that the flow described by the Navier-Stokes equations is often too detailed to be solved, even with numerical methods. This is significant because most practical applications feature turbulent flows (Versteeg and Malalasekera, 2007).

3.1.2 Reynolds averaging

The Navier-Stokes equations have to be simplified for practical applications. First, define the time average of any variable x to be

$$\overline{x} = \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} x(t)\, dt. \tag{3.5}$$

One way to simplify the equations is to split the velocity $\mathbf{u}$ into the mean $\overline{\mathbf{u}}$ and a fluctuation about the mean $\mathbf{u}'$ so that

$$\mathbf{u} = \overline{\mathbf{u}} + \mathbf{u}' \tag{3.6}$$

for statistically stationary flows. This is called the Reynolds decomposition, and it leads to the Reynolds-averaged Navier-Stokes equations.

Pope (2000) puts it this way. Consider incompressible flows with constant viscosity, and note that

$$\overline{\left(\frac{\partial u_j}{\partial t} + \sum_i \frac{\partial}{\partial x_i}(u_i u_j)\right)} = \frac{\partial \overline{u_j}}{\partial t} + \sum_i \frac{\partial}{\partial x_i}\overline{u_i u_j} \tag{3.7}$$

as averaging and differentiating can swap order. Then, using the definition of the Reynolds decomposition and the simple averaging rules given in Versteeg and Malalasekera (2007, p.62), we have

$$\begin{aligned}
\overline{u_i u_j} &= \overline{(\overline{u_i} + u_i')(\overline{u_j} + u_j')} && (3.8)\\
&= \overline{(\overline{u_i}\,\overline{u_j} + u_i'\overline{u_j} + u_j'\overline{u_i} + u_i' u_j')} && (3.9)\\
&= \overline{(\overline{u_i}\,\overline{u_j})} + \overline{(u_i'\overline{u_j})} + \overline{(u_j'\overline{u_i})} + \overline{(u_i' u_j')} && (3.10)\\
&= \overline{u_i}\,\overline{u_j} + \overline{u_i' u_j'} && (3.11)
\end{aligned}$$

and so

$$\begin{aligned}
\overline{\left(\frac{\partial u_j}{\partial t} + \sum_i \frac{\partial}{\partial x_i}(u_i u_j)\right)} &= \frac{\partial \overline{u_j}}{\partial t} + \sum_i \frac{\partial}{\partial x_i}(\overline{u_i}\,\overline{u_j} + \overline{u_i' u_j'}) && (3.12)\\
&= \frac{\partial \overline{u_j}}{\partial t} + \sum_i \overline{u_i}\frac{\partial \overline{u_j}}{\partial x_i} + \sum_i \frac{\partial}{\partial x_i}\overline{u_i' u_j'}. && (3.13)
\end{aligned}$$

since $\Sigma_i \partial\overline{u_i}/\partial x_i = 0$ because of incompressibility. Rearranging, and substituting in the momentum equation (3.2) for $\rho = \text{const}$, we get

$$\rho\left(\frac{\partial\overline{u_j}}{\partial t} + \sum_i \overline{u_i}\frac{\partial\overline{u_j}}{\partial x_i}\right) = \overline{\left(\sum_i \frac{\partial}{\partial x_i}\left(-p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) + \rho g_i\right)} - \rho\sum_i \frac{\partial}{\partial x_i}\overline{u_i' u_j'} \tag{3.14}$$

$$= \sum_i \frac{\partial}{\partial x_i}\left(-\overline{p}\delta_{ij} + \mu\left(\frac{\partial\overline{u_i}}{\partial x_j} + \frac{\partial\overline{u_j}}{\partial x_i}\right) - \rho\overline{u_i' u_j'}\right) + \rho\overline{g_i}, \tag{3.15}$$

which is the Reynolds-averaged momentum equation, where δ_{ij} is the Kronecker delta function. Therefore, the only difference to replacing the velocity in the momentum equation (3.2) by its time average is that we have this extra part

$$-\rho\overline{u_i' u_j'} \tag{3.16}$$

which looks like a stress term. This Reynolds stress term adds more unknowns but no more equations. To close the problem, we need more equations. The extra equations come from turbulence models. Note that, although this derivation of the Reynolds-averaged equations required the assumption of constant density, there are analogous equations for compressible flows called the Favre-averaged equations (Versteeg and Malalasekera, 2007).

3.2 Turbulence modelling

3.2.1 DNS, RAS, or LES?

While turbulent flow is too detailed to be fully resolved by numerical methods, this is a computational constraint, not a mathematical one. In fact, solving the non-averaged Navier-Stokes equations approximately is theoretically possible—it is called Direct Numerical Simulation (DNS)—but it is currently only used to research turbulence itself. This is because flow with $\text{Re} = 10^4$ requires 10^9 cells and at least 100 time steps (Versteeg and Malalasekera, 2007). Therefore, practical applications (with larger cells) need to incorporate the effect of at least some scales of the turbulence using empirical formulae. This is called turbulence modelling.

Using the Reynolds-averaged equations is called Reynolds-Averaged Simulation (RAS) and it is the opposite end of the spectrum from DNS. Every scale of turbulence is modelled, not resolved. That is, the Reynolds stress $-\rho\overline{u_i'u_j'}$ is calculated using empirical formulae, and we only have results for the temporal mean $\overline{\mathbf{u}}$, which does not include any turbulence-related fluctuations (Versteeg and Malalasekera, 2007). Incidentally, even though turbulence is inherently transient, modelling it using RAS means that we can conduct steady-state simulations.

Large Eddy Simulation (LES) lies between DNS and RAS. The large turbulent eddies are directly resolved, but the effect of smaller-scale turbulence is modelled using empirical formulae. This is done using spatial filtering which, similarly to Reynolds averaging, results in an additional stress term called the subgrid-scale stress. However, as some eddies are resolved directly, LES is not suitable for steady-state simulations, only transient (Versteeg and Malalasekera, 2007).

Chapter 4 uses RAS turbulence models due to the decreased computational cost and the fact that resolving any turbulence would give a level of detail not necessary for the intended application. In the next sections, we consider the key RAS turbulence models that are currently available.

3.2.2 *k*-ϵ model

The most widely used RAS turbulence model is the k-ϵ model of Launder and Spalding (1974), where $k = \langle \mathbf{u}' \cdot \mathbf{u}' \rangle/2$ is the turbulent kinetic energy and ϵ the dissipation of the turbulent kinetic energy. The idea of the model is to treat the Reynolds stress $-\rho\overline{u_i'u_j'}$ like the viscous stress, although this approach is based on intuition rather than theoretical insight. Boussinesq hypothesised that

$$-\rho\overline{u_i'u_j'} = \mu_t \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) - \frac{2}{3}\rho k \delta_{ij} \tag{3.17}$$

where $\mu_t = \rho\nu_t$ is the turbulent or eddy viscosity, ν_t the kinematic eddy viscosity, and δ_{ij} the Kronecker delta function. The first part of the right-hand side of (3.17) is analogous to viscous stresses but with turbulent viscosity instead of the usual viscosity. The second part ensures that the normal stresses sum to $-\rho\langle \mathbf{u}' \cdot \mathbf{u}' \rangle$, as required (Versteeg and Malalasekera, 2007).

Tennekes and Lumley (1972) showed that the turbulent kinetic energy k is transported by the equation

$$\underbrace{\frac{\partial}{\partial t}(\rho k)}_{\text{rate of change of } \rho k} + \underbrace{\nabla \cdot (\rho k \overline{\mathbf{u}})}_{\text{convection of } \rho k}$$

$$= \nabla \cdot \left(- \underbrace{\overline{p'\mathbf{u}'}}_{\text{pressure}} + \underbrace{2\mu \overline{\mathbf{u}' s'_{ij}}}_{\text{viscous stress}} - \underbrace{\rho \frac{1}{2}\overline{(u'_i \cdot u'_i u'_j)}}_{\text{Reynolds stress}} \right) - \underbrace{2\mu \overline{(s'_{ij} \cdot s'_{ij})}}_{\text{dissipation of } \rho k} - \underbrace{\rho \overline{u'_i u'_j} \cdot \overline{s_{ij}}}_{\text{production of } \rho k} , \quad (3.18)$$

where the rate of deformation s_{ij} can be decomposed into mean and fluctuating parts,

$$s_{ij} = \overline{s_{ij}} + s'_{ij} = \underbrace{\frac{1}{2}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)}_{\text{mean deformation}} + \underbrace{\frac{1}{2}\left(\frac{\partial u'_i}{\partial x_j} + \frac{\partial u'_j}{\partial x_i}\right)}_{\text{fluctuating deformation}} . \quad (3.19)$$

However, this is complicated, and the equation for ϵ is more problematic still (Versteeg and Malalasekera, 2007). To overcome this complexity, the k-ϵ model simplifies the transport equations of k and ϵ to

$$\underbrace{\frac{\partial(\rho k)}{\partial t}}_{\text{rate of change of } \rho k} + \underbrace{\nabla \cdot (\rho k \overline{\mathbf{u}})}_{\text{convection of } \rho k} = \underbrace{\nabla \cdot \left(\frac{\mu_t}{\sigma_k} \nabla k\right)}_{\text{diffusion of } \rho k} + \underbrace{2\mu_t \overline{s_{ij}} \cdot \overline{s_{ij}}}_{\text{production of } \rho k} - \underbrace{\rho \epsilon}_{\text{destruction of } \rho k} \quad (3.20)$$

$$\underbrace{\frac{\partial(\rho \epsilon)}{\partial t}}_{\text{rate of change of } \rho \epsilon} + \underbrace{\nabla \cdot (\rho \epsilon \overline{\mathbf{u}})}_{\text{convection of } \rho \epsilon} = \underbrace{\nabla \cdot \left(\frac{\mu_t}{\sigma_\epsilon} \nabla \epsilon\right)}_{\text{diffusion of } \rho \epsilon} + \underbrace{C_{1\epsilon}\frac{\epsilon}{k} 2\mu_t \overline{s_{ij}} \cdot \overline{s_{ij}}}_{\text{production of } \rho \epsilon} - \underbrace{C_{2\epsilon}\rho\frac{\epsilon^2}{k}}_{\text{destruction of } \rho \epsilon} \quad (3.21)$$

where

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} \quad (3.22)$$

and the empirical constants are $C_\mu = 0.09$, $\sigma_k = 1.00$, $\sigma_\epsilon = 1.30$, $_{1\epsilon} = 1.44$, and $C_{2\epsilon} = 1.92$, which come from "comprehensive data fitting for a wide range of turbulent flows" (Versteeg and Malalasekera, 2007, p.76).

Consider each of the terms in turn. The k and ϵ diffusion terms are modelled with gradient diffusion, while the k production term is identical to that in the exact equation (3.18) with the Boussinesq hypothesis (3.17) substituted in. Note how the ϵ production and destruction terms are equal to those for k multiplied by ϵ/k and a constant. This is so that k and ϵ increase and decrease with each other (Versteeg and Malalasekera, 2007). While these empirical equations are much simpler than the exact equations, Wilcox (2006, p.129) calls their derivation "drastic surgery" and, especially for the ϵ-equation, no better than dimensional analysis.

3.2.3 *k*-*ω* model

The k-ω model of Wilcox (1988) uses the turbulence frequency $\omega = \epsilon/k$ instead of the turbulent dissipation ϵ, with the eddy viscosity given by

$$\mu_t = \frac{\rho k}{\omega}. \tag{3.23}$$

The empirical k-equation is

$$\underbrace{\frac{\partial(\rho k)}{\partial t}}_{\text{rate of change of } \rho k} + \underbrace{\nabla \cdot (\rho k \overline{\mathbf{u}})}_{\text{convection of } \rho k} = \underbrace{\nabla \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right)}_{\text{turbulent diffusion of } \rho k} + \underbrace{2\mu_t \overline{s_{ij}} \cdot \overline{s_{ij}} - \frac{2}{3} \rho k \frac{\partial \overline{u_i}}{\partial x_j} \delta_{ij}}_{\text{production of } \rho k} - \underbrace{\beta^* \rho k \omega}_{\text{dissipation of } \rho k}, \tag{3.24}$$

the empirical ω-equation is

$$\underbrace{\frac{\partial(\rho \omega)}{\partial t}}_{\text{rate of change of } \rho\omega} + \underbrace{\nabla \cdot (\rho \omega \overline{\mathbf{u}})}_{\text{convection of } \rho\omega} = \underbrace{\nabla \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_\omega} \right) \nabla \omega \right)}_{\text{turbulent diffusion of } \rho\omega} + \underbrace{\gamma_1 \left(2\rho \overline{s_{ij}} \cdot \overline{s_{ij}} - \frac{2}{3} \rho \omega \frac{\partial \overline{u_i}}{\partial x_j} \delta_{ij} \right)}_{\text{production of } \rho\omega} - \underbrace{\beta_1 \rho \omega^2}_{\text{dissipation of } \rho\omega}, \tag{3.25}$$

and the empirical constants are $\sigma_k = 2.0$, $\sigma_\omega = 2.0$, $\gamma_1 = 0.553$, $\beta_1 = 0.075$, and $\beta^* = 0.09$. The ω-equation is not the same as substituting $\epsilon = k\omega$ into (3.21). It has the advantage over the k-ϵ model of working for low Reynolds number flows without wall-damping functions, but the disadvantage of sensitivity to ω in the free stream (Versteeg and Malalasekera, 2007).

3.2.4 *k*-*ω* SST model

To avail of the relative advantages of the k-ϵ and k-ω models, Menter (1994) blended the two in his Shear Stress Transport (SST) model, using k-ϵ away from the wall and k-ω near the wall. The only difference between the models presented above is that there is

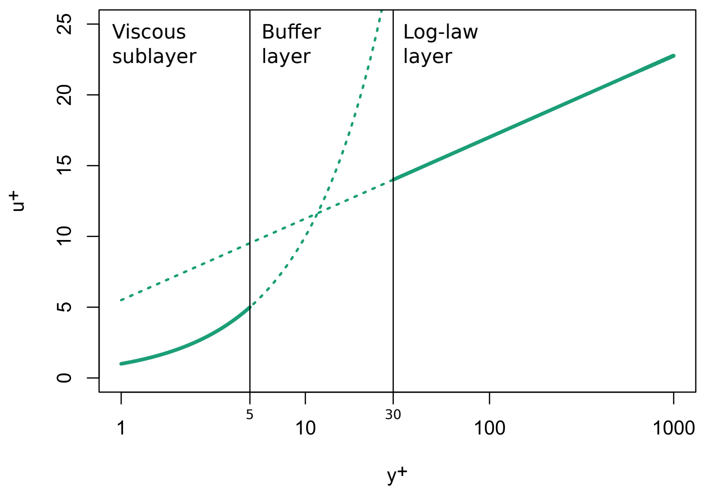

Figure 3.1: Velocity profile in the viscous sublayer and log-law region. The buffer region is not described well by either of these functions. Note that the log-law layer is a straight line because the y^+ axis has a logarithmic scale.

a different transport equation for ω, namely that resulting from substituting $\epsilon = k\omega$ into (3.21),

$$\underbrace{\frac{\partial(\rho\omega)}{\partial t}}_{\text{rate of change of } \rho\omega} + \underbrace{\nabla\cdot(\rho\omega\bar{\mathbf{u}})}_{\text{convection of } \rho\omega}$$
$$= \underbrace{\nabla\cdot\left(\left(\mu + \frac{\mu_t}{\sigma_{\omega,1}}\right)\nabla\omega\right)}_{\text{turbulent diffusion of } \rho\omega} + \underbrace{\gamma_2\left(2\rho\overline{s_{ij}}\,\overline{s_{ij}} - \frac{2}{3}\rho\omega\frac{\partial\overline{u_i}}{\partial x_j}\delta_{ij}\right)}_{\text{production of } \rho\omega} - \underbrace{\beta_2\rho\omega^2}_{\text{dissipation of } \rho\omega} + \underbrace{2\frac{\rho}{\sigma_{\omega,2}\omega}\frac{\partial k}{\partial x_k}\frac{\partial\omega}{\partial x_k}}_{\text{cross-diffusion}}, \tag{3.26}$$

where there is now a cross-diffusion term due to the difference between the k-ϵ and k-ω models (Versteeg and Malalasekera, 2007). Menter et al. (2003) suggested optimal empirical coefficients of $\sigma_k = 1.0$, $\sigma_{\omega,1} = 2.0$, $\sigma_{\omega,2} = 1.17$, $\gamma_2 = 0.44$, and $\beta_2 = 0.083$.

3.2.5 Wall functions

Walls impose a no-slip boundary condition, giving rise to the universal velocity profile shown in Figure 3.1. It might initially seem like very small cells are needed near walls so that the velocity profile can be captured in sufficient detail. However, wall functions provide a way to model near-wall flows with bigger cells, saving on computation time. First, note the three distinct layers in the velocity profile. There is a viscous or linear sublayer, a log-law layer, and a buffer layer separating the two, while the outer layer lies beyond the log-law layer. The values u^+ and y^+ are dimensionless groups, defined by the law of the wall:

$$u^+ = \frac{u}{u_\tau} \tag{3.27}$$

$$y^+ = \frac{\rho u_\tau y}{\mu} \tag{3.28}$$

$$u^+ = f(y^+) \tag{3.29}$$

where y is the distance from the wall, $u_\tau = \sqrt{\tau_w/\rho}$ the friction velocity, and τ_w the wall shear stress (Versteeg and Malalasekera, 2007).

The function f is defined piecewise. Viscous forces dominate in the viscous sublayer ($y^+ < 5$), leading to the linear relationship

$$u^+ = y^+. \tag{3.30}$$

Both viscous and turbulent forces are important in the log-law layer ($30 < y^+ < 500$), leading to the logarithmic relationship

$$u^+ = \frac{1}{\kappa}\ln(y^+) + B = \frac{1}{\kappa}\ln(Ey^+), \tag{3.31}$$

where $\kappa \approx 0.4$ is von Karman's constant and $B \approx 5.5$ or $E \approx 9.8$. However, the buffer layer ($5 < y^+ < 30$) is not described well by either the linear or logarithmic relationships. One option is to mark the transition at $y^+ = 11.63$, where the linear and logarithmic lines intersect, but the best option is to avoid the buffer layer altogether.

If the first cells adjacent to the wall have centres within the log-law region ($30 < y^+ < 500$), then the law of the wall (3.29) gives the friction velocity u_τ at the cell centre. This can then be substituted into the wall functions

$$k = \frac{u_\tau^2}{\sqrt{\beta^*}},\ \epsilon = \beta^{*3/4}\frac{k^{3/2}}{\kappa y},\ \omega = \frac{k^{1/2}}{\beta^{*1/4}\kappa y} \tag{3.32}$$

to find the turbulence parameters at the cell centre (Wilcox, 2006). The turbulent viscosity $\nu_t = \mu_t/\rho$ is found either by substituting

$$y_+ = \frac{\rho y u_\tau}{\mu} = \frac{\rho y\sqrt{\tau_w/\rho}}{\mu} \text{ or } y_* = \frac{\rho y u_*}{\mu} = \frac{\rho y\sqrt{C_\mu^{1/2}k}}{\mu} \tag{3.33}$$

into

$$\nu_t = \nu \left(\frac{y_+ \kappa}{\log(E y_+)} - 1 \right) \text{ or } \nu_t = \nu \left(\frac{y_* \kappa}{\log(E y_*)} - 1 \right), \tag{3.34}$$

as described by Moukalled et al. (2016). For non-equilibrium conditions, the value y_* is more suitable than y_+ (Launder and Spalding, 1974).

This requires some level of guessing; we need to know y^+ to determine the cell size but we are doing the simulation precisely because we want to find out what the flow field is like. If the cells are too big, then the centres will lie in the outer layer and the wall functions will no longer be valid. If the cells are too small, then the centres will lie in the viscous sublayer and the wall functions will no longer be valid. However, some level of automation can be implemented in the k-ω model so that, if the cell centres are in the viscous sublayer, then different wall functions (low Re) are used, and we just need to avoid the outer layer (Menter and Esch, 2001).

3.2.6 Inlets and initialisation

The inlets and initial conditions must set values for the turbulence parameters. These are based on the reference flow speed $|\mathbf{u}|$, the turbulence intensity I, and the characteristic length scale L. For the turbulent kinetic energy we have

$$k = \frac{3}{2}(|\mathbf{u}| I)^2, \tag{3.35}$$

the turbulent dissipation rate

$$\epsilon = \frac{1}{L} C_\mu^{0.75} k^{1.5}, \tag{3.36}$$

and the specific turbulent dissipation rate

$$\omega = \frac{1}{L} C_\mu^{-0.25} k^{0.5}. \tag{3.37}$$

(Versteeg and Malalasekera, 2007; OpenCFD, 2019b). The turbulent viscosity ν_t is already determined by these values so does not need to be specified explicitly. The coefficient C_μ is used in the calculation of μ_t in (3.22), set to be 0.09 by default (see Section 3.2.2 above), and the turbulent intensity I is generally given by a guess, usually somewhere in the range 1–20%. As long as the parameter values are generally sensible, they will settle to the right value away from the inlet and starting time.

3.3 Standard finite volume methods

The continuity and momentum equations (3.1)–(3.2) have to be solved simultaneously, which is computationally problematic as there is no equation for pressure. A popular way to address this problem is with a pressure-based method based on matrix inversion (Patankar and Spalding, 1972; Issa, 1986). As this is the method generally used in OpenFOAM, it is the focus of this section. We first outline how the governing equations are discretised and collected into a system of equations, then how matrix solvers work, and finally what the main pressure-based algorithms are.

3.3.1 Discretisation

First, note that the conservative Navier-Stokes momentum equation (3.2) can be written in the more general form

$$\underbrace{\frac{\partial}{\partial t}(\rho\theta)}_{\text{time}} + \underbrace{\nabla\cdot(\rho\mathbf{u}\theta)}_{\text{convection}} - \underbrace{\nabla\cdot(\Gamma\nabla\theta)}_{\text{diffusion}} = \underbrace{S(\theta)}_{\text{sources}} \tag{3.38}$$

where θ is an arbitrary scalar field and Γ an arbitrary diffusion coefficient. Integrating over a computational cell Ω (Figure 3.2) gives

$$\iiint_\Omega \frac{\partial\rho\theta}{\partial t}\,dV + \iiint_\Omega \nabla\cdot(\rho\mathbf{u}\theta)\,dV - \iiint_\Omega \nabla\cdot(\Gamma\nabla\theta)\,dV = \iiint_\Omega S(\theta)\,dV. \tag{3.39}$$

Let us consider each term in turn to derive the basis of the finite volume method, as in Guerrero (2019) and OpenCFD (2019a). Differentiation and integration can swap order for continuous functions, and so we have

$$\iiint_\Omega \frac{\partial\rho\theta}{\partial t}\,dV = \frac{\partial}{\partial t}\iiint_\Omega (\rho\theta)\,dV. \tag{3.40}$$

Meanwhile, we can use Gauss's theorem to get

$$\iiint_\Omega \nabla\cdot(\rho\mathbf{u}\theta)\,dV = \iint_{\partial\Omega}(\rho\mathbf{u}\theta)\cdot d\mathbf{S} \approx \sum_{f\in\partial\Omega}(\rho\mathbf{u}\theta)_f\cdot\mathbf{S}_f = \sum_{f\in\partial\Omega}\varphi_f\theta_f, \tag{3.41}$$

where $\varphi_f = (\rho\mathbf{u})_f\cdot\mathbf{S}_f$ is the flux, and

$$\iiint_\Omega \nabla\cdot(\Gamma\nabla\theta)\,dV = \iint_{\partial\Omega}(\Gamma\nabla\theta)\cdot d\mathbf{S} \approx \sum_{f\in\partial\Omega}(\Gamma\nabla\theta)_f\cdot\mathbf{S}_f \tag{3.42}$$

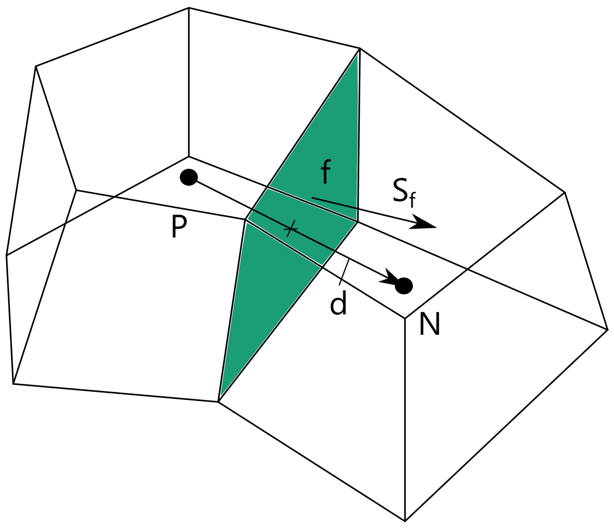

Figure 3.2: Two cells sharing a face. As in OpenFOAM, the owner cell is marked by P, the neighbour cell by N, the face by f, and the cell face area vector by $\mathbf{S}_f$. Based on OpenCFD (2019a, p.32).

where the gradient in the diffusion term also has to be approximated (more on that in Section 3.3.2). Note that (3.42) can be called the Laplacian term because Γ is constant (Greenshields and Weller, 2022).

Finally, a source term can be treated explicitly if it is constant or small, but this may lead to divergence if the source term is large. Therefore, the source term is linearised to get

$$S = S_C + S_P\theta_P \tag{3.43}$$

where S_P is treated implicitly and must be non-positive for diagonal dominance, while S_C is treated explicitly (Patankar, 1980; Moukalled et al., 2016). Note that θ_P is the future value of θ and S_P is the coefficient of θ_P, not S evaluated at P. One way to find S_C and S_P is with a Taylor series expansion. Integrating over Ω, we have

$$\iiint_\Omega S(\theta)\, dV \approx |\Omega|(S_C + S_P\theta_P). \tag{3.44}$$

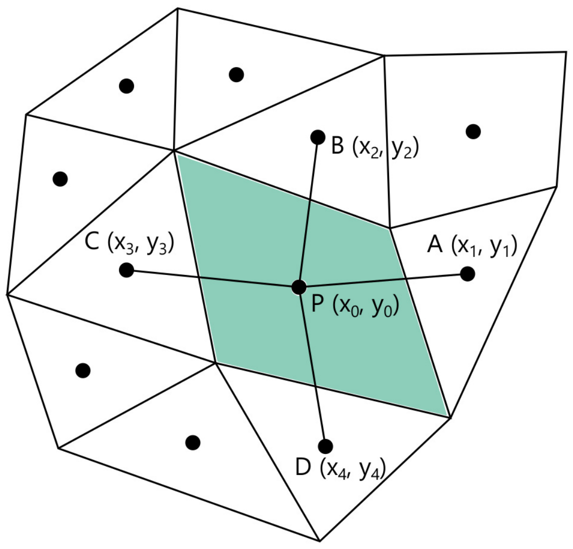

Figure 3.3: Parameters for 2D least-squares gradient calculation, based on Versteeg and Malalasekera (2007, p.322).

3.3.2 Gradient

There are two main ways to approximate the gradient: Green-Gauss and least-squares. The Green-Gauss method calculates the gradient of a scalar using

$$\iiint_{\Omega} (\nabla\theta)\, dV = \iint_{\delta\Omega} \theta\, d\mathbf{S} \approx \sum_{f\in\partial\Omega} \theta_f \mathbf{S}_f \tag{3.45}$$

where θ must be interpolated to the cell faces with a specified interpolation scheme (usually linear) unless it is a boundary face, where the value is set by the boundary conditions.

The least-squares method is more complicated. Consider the 2D stencil in Figure 3.3 and the subscripts $0, 1, \ldots, N$ defined there, as in Versteeg and Malalasekera (2007). Now, the cell-centre value of a scalar θ in cell i can be given by

$$\theta_i = \theta_0 + \left(\frac{\partial\theta}{\partial x}\right)\Big|_0 (x_i - x_0) + \left(\frac{\partial\theta}{\partial y}\right)\Big|_0 (y_i - y_0) \tag{3.46}$$

$$= \theta_0 + \left(\frac{\partial\theta}{\partial x}\right)\Big|_0 \Delta x_i + \left(\frac{\partial\theta}{\partial y}\right)\Big|_0 \Delta y_i. \tag{3.47}$$

Assemble the equations from each of the N cells adjacent to P into the system

$$\underbrace{\begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \Delta x_3 & \Delta y_3 \\ \dots & \dots \\ \Delta x_N & \Delta y_N \end{bmatrix}}_{\text{A}} \cdot \underbrace{\begin{bmatrix} \frac{\partial \theta}{\partial x}\Big|_0 \\ \frac{\partial \theta}{\partial y}\Big|_0 \end{bmatrix}}_{\text{x}} = \underbrace{\begin{bmatrix} \theta_1 - \theta_0 \\ \theta_2 - \theta_0 \\ \theta_3 - \theta_0 \\ \dots \\ \theta_N - \theta_0 \end{bmatrix}}_{\text{b}}, \tag{3.48}$$

which, as it is overdetermined, can be solved using the least-squares method:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{3.49}$$

Blazek (2005) notes that the Gram-Schmidt process should be used to decompose **A** on stretched grids.

3.3.3 Interpolation

We need to calculate $(\rho\mathbf{u})_f$ and θ_f for the divergence term (3.41) and Γ_f for the Laplacian term (3.42). The simple way to approximate these cell interface values is to interpolate them from the cell centres. For the convective term, this can be upwind, taking the value of the upwind cell centre. While this is very stable, it is diffusive and only first-order. Linearly interpolating between the two cells gives a second-order approximation. The value $(\rho\mathbf{u})_f$ is usually linearly interpolated, but this method can be oscillatory if used for the advected quantity θ_f. Therefore, some blend between upwind and linear is required to obtain a solution that is both stable and accurate, and possibly also bounded (Tabor, 2018; OpenCFD, 2019a; Greenshields and Weller, 2022).

This can be done using a Total Variation Diminishing (TVD) scheme to interpolate between the owner and neighbour cells in such a way that no extrema are introduced or exacerbated. We have

$$\theta_f = (1 - \psi(r))\theta_U + \psi(r)\theta_L \tag{3.50}$$

where θ_U is the face value of θ from the upwind scheme, θ_L is the face value of θ from the linear scheme, $\psi(r)$ is the TVD limiter, and r is given by

$$r = 2\frac{\mathbf{d}\cdot(\nabla\theta)_P}{\mathbf{d}\cdot(\nabla\theta)_f} - 1 = 2\frac{\mathbf{d}\cdot(\nabla\theta)_P}{\theta_N - \theta_P} - 1 \tag{3.51}$$

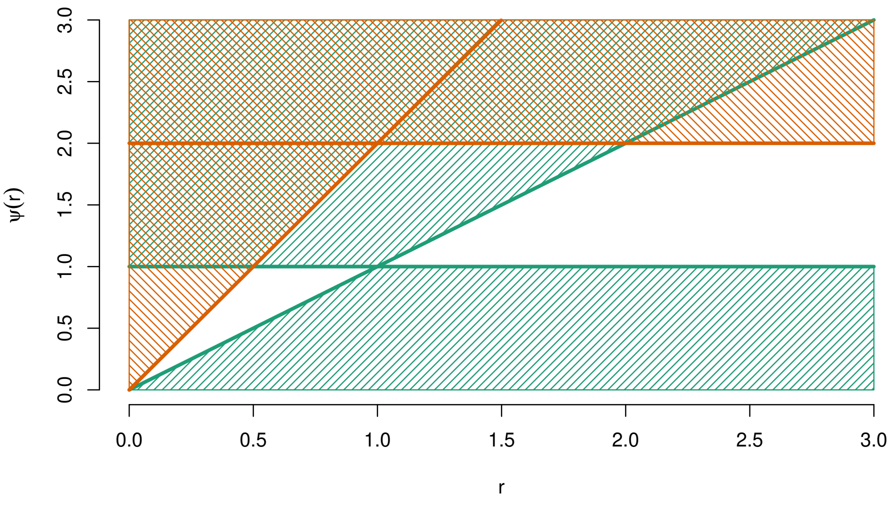

Figure 3.4: Second-order TVD region, based on Sweby (1984, p.1001). The orange lines correspond to TVD constraints while the green lines correspond to second-order constraints.

where $\mathbf{d} = \mathbf{C}_N - \mathbf{C}_P$ is the distance between the owner and neighbour cell centres (Greenshields and Weller, 2022). Darwish and Moukalled (2003) note that this definition of r comes from generalising the r from structured meshes—where it is simply "the upwind ratio of consecutive gradients of the solution" (p.600)—to unstructured meshes in such a way that it is possible to calculate it with the information provided by an unstructured mesh. The necessary and sufficient conditions of $\psi(r)$ to be both second-order and TVD can be plotted on a Sweby diagram, as shown in Figure 3.4. Note that interpolating like this is not the only way to approximate the cell interface values, but it is fairly standard. An alternative approach is used in Chapters 5–6.

3.3.4 Mesh quality

A slightly different method is required for the diffusion term. The problem is the dot product. If there is a face f lying between an owner cell P and a neighbour cell N, we might be tempted to calculate $\nabla\theta_f \cdot \mathbf{S}_f$ by

$$\nabla\theta_f \cdot \mathbf{S}_f \approx S_f \frac{\theta_N - \theta_P}{\mathbf{d}} \tag{3.52}$$

where $\mathbf{d} = \mathbf{x}_N - \mathbf{x}_P$ and $S_f = |\mathbf{S}_f|$. However, this only makes sense when $\mathbf{S}_f$ lies along $\mathbf{d}$, which is not the case in non-orthogonal meshes such as that shown in Figure 3.5.

Jasak (1996) proposed some alternatives that make more sense for non-orthogonal meshes. Before looking at these, split $\nabla\theta_f \cdot \mathbf{S}_f$ into an orthogonal part and a non-orthogonal part:

$$\nabla\theta_f \cdot \mathbf{S}_f = \underbrace{\nabla\theta_f \cdot \Delta}_{\text{orthogonal}} + \underbrace{\nabla\theta_f \cdot \mathbf{k}}_{\text{non-orthogonal}} \tag{3.53}$$

where $\mathbf{S}_f = \Delta + \mathbf{k}$ and Δ lies along $\mathbf{d}$. Then the orthogonal part with Δ can be calculated using (3.52), and the non-orthogonal part calculated as an explicit source term. Treating the non-orthogonal part explicitly makes the matrix easier to solve iteratively. There are an infinite number of possibilities for Δ, but we will consider the three in Figure 3.6. For conciseness, we now omit the subscript in $\mathbf{S}_f$. The first approach is minimum correction, where $\mathbf{k}$ is minimised. Then we have

$$|\Delta| = \frac{\mathbf{S}\cdot\mathbf{d}}{|\mathbf{d}|} \Rightarrow \Delta = \mathbf{d}\frac{\mathbf{S}\cdot\mathbf{d}}{|\mathbf{d}|^2}. \tag{3.54}$$

The second approach is orthogonal correction, where $\mathbf{S}$ and Δ have the same magnitude. Then we have

$$|\Delta| = |\mathbf{S}| \Rightarrow \Delta = |\mathbf{S}|\frac{\mathbf{d}}{|\mathbf{d}|}. \tag{3.55}$$

The third approach is over-relaxation, where $\mathbf{k}$ is made perpendicular to $\mathbf{S}$. Then we have

$$|\Delta| = |\mathbf{S}|\frac{|\mathbf{d}||\mathbf{S}|}{\mathbf{d}\cdot\mathbf{S}} \Rightarrow \Delta = |\mathbf{S}|^2\frac{\mathbf{d}}{\mathbf{d}\cdot\mathbf{S}}. \tag{3.56}$$

Note that k is smallest for the minimum correction approach and largest for the over-relaxation approach. The larger the value of k, the larger the error.

Jasak (1996) showed that, as the angle between Δ and $\mathbf{S}$ increases, the minimum correction approach diverges, and then so does the orthogonal correction approach. Meanwhile, the over-relaxed approach stays stable, even though it takes a while to converge for the most non-orthogonal meshes. This is because of the relative contributions of Δ and k as the non-orthogonality increases. Hence we have the competing demands of accuracy and stability.

Skewness causes greater numerical diffusion. We desire the value of a variable at the centre of the face, f, so interpolate between the cell centres P and N. However, if a mesh

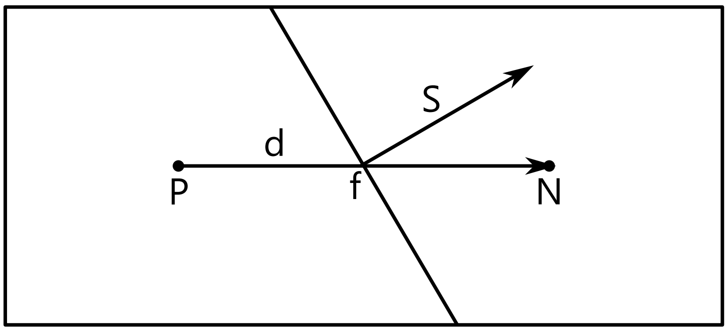

Figure 3.5: A non-orthogonal mesh, based on Jasak (1996, p.83).

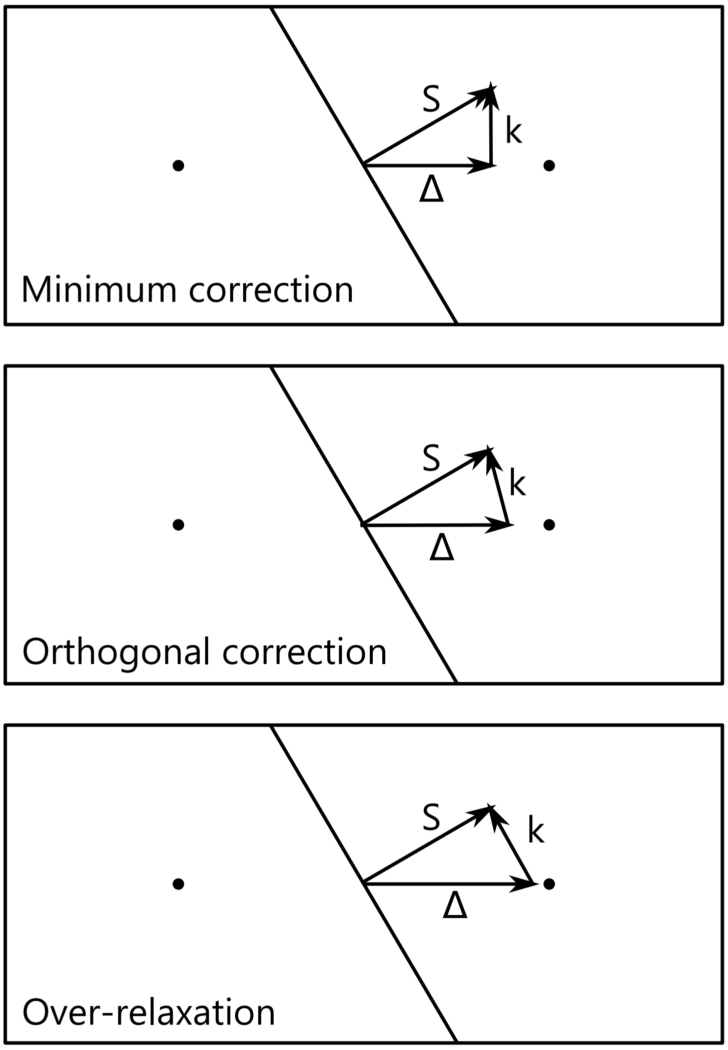

Figure 3.6: A non-orthogonal mesh, based on Jasak (1996, pp.84–85).

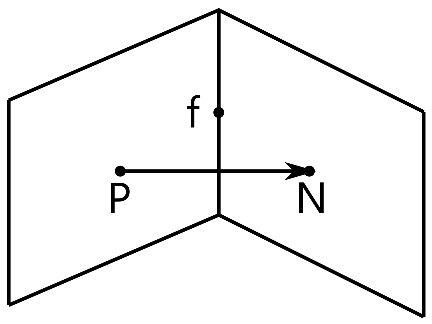

Figure 3.7: A skewed mesh, based on Jasak (1996, p.124).

is skewed as in Figure 3.7, this may not be in the centre of the face as desired, introducing "diffusion-like" error into the solution (Jasak, 1996, p.124). Note that a mesh can be skewed without being non-orthogonal, and vice versa.

3.3.5 Matrix solvers

Implicit schemes assemble the discretised equations into the system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.57}$$

which is solved for **x**, that is,

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{3.58}$$

This equation can be solved exactly with Gaussian elimination, but that is unrealistically expensive for practical applications. Iterative techniques, such as Jacobi or Gauss-Seidel iteration, are used instead, and there are ways to get the iteration to converge faster. One such method is preconditioning, where the system (3.57) is transformed into one that has the same solution but is easier to solve. This is done using a preconditioning matrix **P** so that we have

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b} \tag{3.59}$$

where $\mathbf{P}^{-1}\mathbf{A}$ has more useful spectral properties than **A** by itself (Moukalled et al., 2016).

Alternatively, gradient methods turn the matrix system (3.57) into a function to be minimised. The steepest descent method follows the path of steepest descent on this function iteratively, which always converges. However, as shown in Figure 3.8, the convergence can be slow due to oscillations around local minima like valleys. The conjugate gradient method

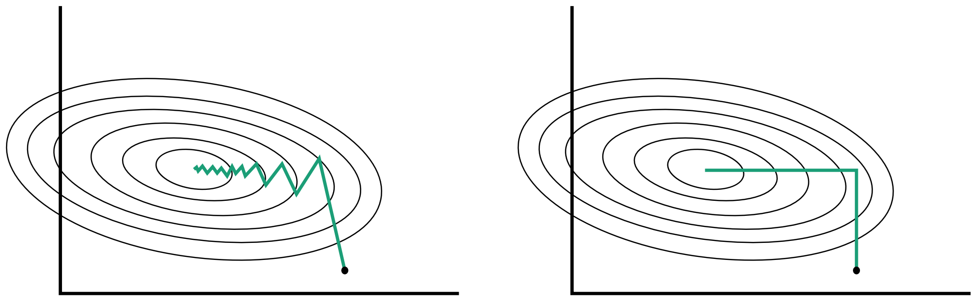

Figure 3.8: Steepest descent and conjugate gradient methods, based on Tabor (2018, p.28).

for symmetric matrices converges quicker than the steepest descent method. It works by minimising along two directions, conjugate with the respect to the matrix **A**. This can be generalised to the biconjugate gradient method for asymmetric matrices (Shewchuk, 1994; Ferzinger and Perić, 2002; Moukalled et al., 2016).

Now, while finer meshes give more accurate solutions, these iterative matrix solvers converge much more slowly on them. This is because the errors have components of different wavelengths, relative to the size of the mesh. Short-wavelength errors are reduced quickly, but long-wavelength errors are reduced more slowly. Again, these depend on the size of the mesh. The short-wavelength errors of a coarse mesh may be the same wavelength as the long-wavelength errors of a fine mesh. Multigrid methods use this feature to get iterative matrix solvers to converge more quickly by transferring between meshes of different resolutions (Versteeg and Malalasekera, 2007).

3.3.6 SIMPLE, PISO, and PIMPLE

There are two main classes of method to deal with the lack of pressure equation. Pressure-based solvers, historically used for low-speed incompressible flows, calculate the pressure from a correction equation. In contrast, density-based solvers, historically used for high-speed compressible flows, calculate the pressure from an equation of state (Ansys, 2009). As will be discussed in Chapters 5–6, there are ways to use pressure-based methods for compressible flows and density-based methods for incompressible flows, but we focus on pressure-based algorithms for incompressible flows in this section.

The following explanation is based on Pieri (2014), Greenshields and Weller (2022), Guerrero (2019), Holzmann (2019), and Wimshurst (2019). We have discretised equations collected in the system

$$\mathbf{Mu} = -\nabla p. \tag{3.60}$$

Extract from the matrix **M** the diagonal part **A** and define

$$\mathbf{H} = \mathbf{Au} - \mathbf{Mu}. \tag{3.61}$$

This is done because diagonal matrices are easy to invert, and we have the notation

$$\mathbf{A}^{-1} = \frac{1}{\mathbf{A}} \tag{3.62}$$

to highlight that each of the diagonal components a_i of **A** get replaced by $1/a_i$ for the inverse $\mathbf{A}^{-1}$. Now, combine (3.60) and (3.61) together to get

$$\mathbf{Au} - \mathbf{H} = -\nabla p \tag{3.63}$$

$$\mathbf{u} = \frac{1}{\mathbf{A}}\mathbf{H} - \frac{1}{\mathbf{A}}\nabla p. \tag{3.64}$$

The incompressible continuity equation (3.3) says that the divergence of the velocity is zero, so take the divergence of (3.64) to get

$$\nabla \cdot \left(\frac{1}{\mathbf{A}}\nabla p\right) = \nabla \cdot \left(\frac{1}{\mathbf{A}}\mathbf{H}\right). \tag{3.65}$$

Figure 3.9 shows the steps taken in the pressure-velocity coupling algorithms. First, the momentum predictor (3.60) is used to obtain **Mu**. Then this is substituted into (3.61) to be decomposed into a diagonal part **A** and **H**. Then **H** becomes part of a source term on the right-hand side of the pressure equation (3.65), which gives us a value of ∇p that obeys the continuity equation. The pressure can then be substituted into the momentum corrector (3.64) so that it obeys the momentum equation. However, now it may no longer obey the continuity equation; iteration is required. Different algorithms use different types of iteration. In the SIMPLE algorithm of Patankar and Spalding (1972), the new velocity value from the momentum corrector is looped back to the momentum predictor. In the PISO algorithm of Issa (1986), the new velocity value from the momentum corrector is looped back to the matrix decomposition, which is less computationally expensive. The

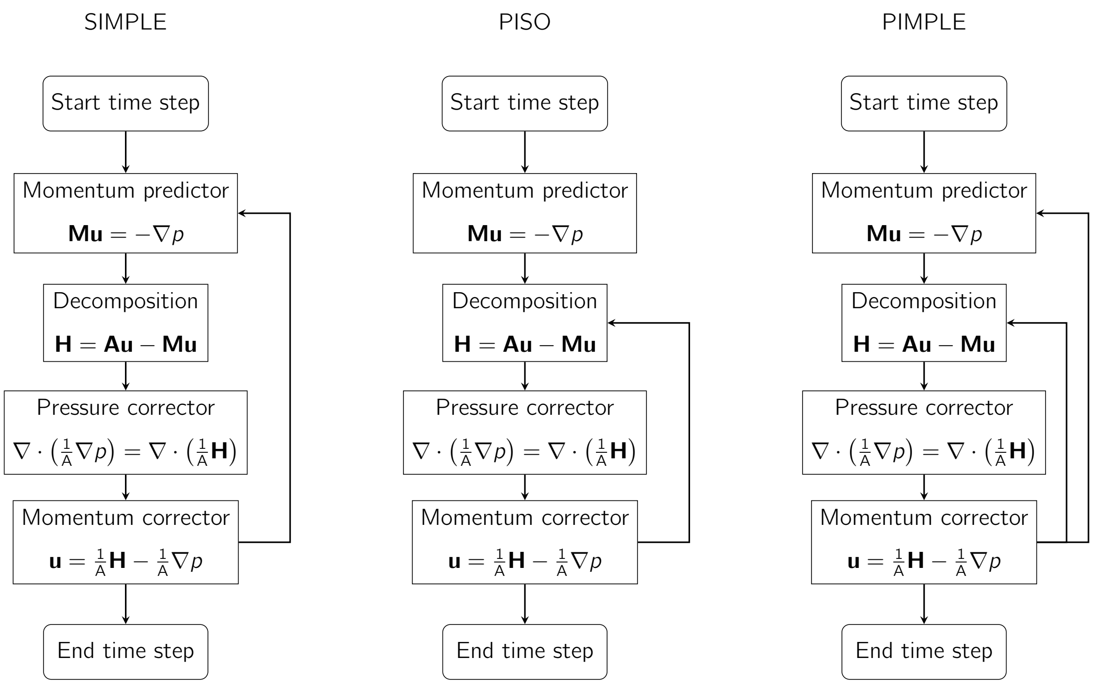
SIMPLE
PISO
PIMPLE
Start time step
Momentum predictor
$\mathbf{Mu} = -\nabla p$
Decomposition
$\mathbf{H} = \mathbf{Au} - \mathbf{Mu}$
Pressure corrector
$\nabla \cdot \left(\frac{1}{A}\nabla p\right) = \nabla \cdot \left(\frac{1}{A}\mathbf{H}\right)$
Momentum corrector
$\mathbf{u} = \frac{1}{A}\mathbf{H} - \frac{1}{A}\nabla p$
End time step

Figure 3.9: Pressure-velocity coupling algorithms.

SIMPLE algorithm is used for steady-state simulations and, as it is not numerically consistent, it includes under-relaxation. In contrast, the PISO algorithm is suitable for transient simulations but it has a stability constraint of CFL < 1. The seemingly OpenFOAM-exclusive PIMPLE algorithm is a combination of the outer correctors in SIMPLE and the inner correctors in PISO. It can be used for transient cases with much larger time steps. Unfortunately, while these methods are popular, they are difficult and inefficient to parallelise due to the requirement for matrix inversion (Nourgaliev et al., 2004; Loppi, 2019; Hodges, 2020).

3.4 Volume of fluid method

Free-surface flows of water and air are of much interest in engineering, and indeed this thesis. They come under the umbrella of multiphase methods, which also includes flows of two or more liquids with an interface. There are different methods to model the interfaces between different fluids. First note that there are two main approaches to CFD: Lagrangian and Eulerian. Modelling interfaces is easy in a Lagrangian approach, where we consider the movement of parcels of fluid. If we start by looking at a parcel of water, it is going to

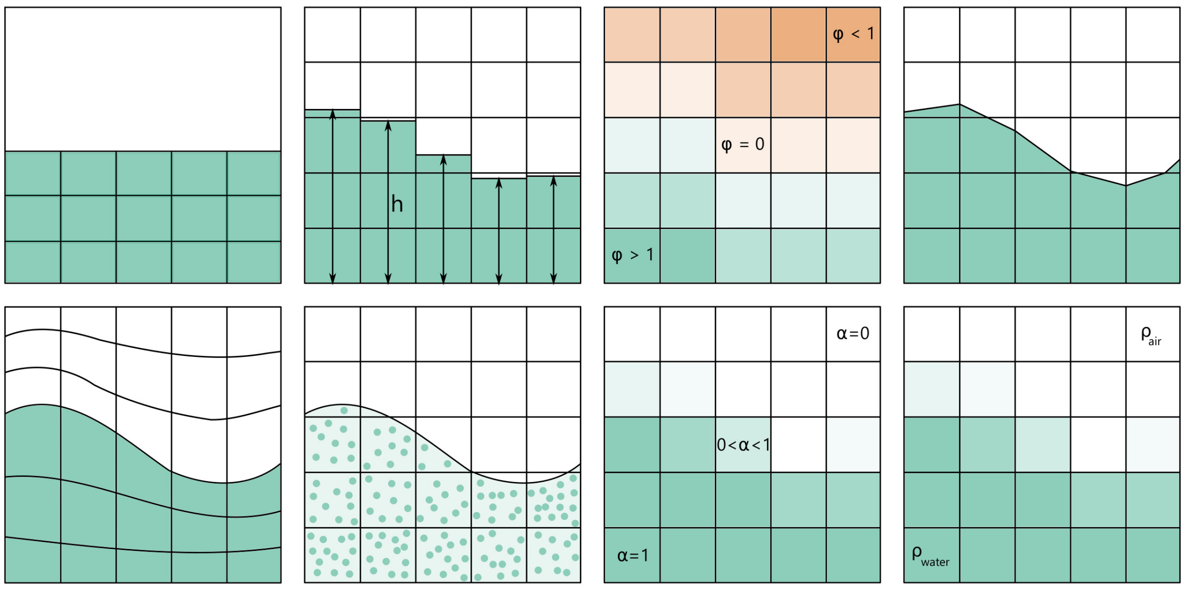

Figure 3.10: Eulerian methods for modelling free-surface flows. From left to right: fixed lid, height functions, level set, front tracking, surface fitted, marker and cell, volume of fluid, Riemann solvers.

stay a parcel of water. It is more difficult to model interfaces with an Eulerian approach, where we consider movement through a fixed region, because the air-water interface has to be dealt with separately. Broadly, there are two different ways to model interfaces using an Eulerian approach: either track the interface sharply using a surface-based method or use a volume-based method that automatically captures the interface, but smeared over a few cells. Figure 3.10 illustrates the following Eulerian multiphase methods:

- Fixed lid. The simplest surface-based method is to model the air-water interface as a fixed lid to the domain (Bates et al., 2005). As this requires the location of the interface, it is not appropriate for many situations, for example, the common scenario when the water depth is required as an output of the model. Note that this method has been used in all the CFD studies of leaky barriers so far (Allen and Smith, 2012; Xu and Liu, 2017), and so they could not be used to quantify any backwater effect. This is a limitation that must be overcome for detailed modelling of different leaky barrier designs.

- Height functions. Height functions are similar to the shallow water equations in that they consider the height of the air-water interface relative to some specified datum, but this time transporting it using a kinematic equation. However, Hirt and Nichols (1981) note that height functions do not work when there might be multiple heights at a given point.

- Level set. The level-set method models the interface as the zero set of a function that is positive in one phase (say, water) and negative in the other (air). This function could be the signed distance from the interface. Sethian and Smereka (2003) note that the level-set method is good for dealing with topological changes and higher dimensions. However, it does need frequent re-initialisation, which is computationally expensive.

- Front tracking. Another way to keep the interface sharp is to track it with line or polynomial segments. Each point is then moved with the local velocity. This method can cope with formation of droplets or self-intersections. However, Hirt and Nichols (1981) note that such topological changes require reordering of the segments, which is especially problematic for 3D cases.

- Surface fitted. In the surface-fitted method, the mesh always lines up with the air-water interface, and so it deforms as the simulation progresses. Hyman (1984) notes that there are two options: either snap the mesh (or its diagonals) to the interface, or allow the mesh to distort completely in a Lagrangian manner. This method keeps the interface sharp, but expensive remeshing is required if the interface distorts too much (Weston, 2000).

- Marker and Cell (MAC). The MAC method is the first volume-based method. It involves locating marker particles within the water. These move with the water, and then the interface is the boundary between the region with markers and the region without. Hirt and Nichols (1981) note that this method can automatically deal with topological changes, but it does require more computational power and storage.

- Volume of Fluid (VOF). Hirt and Nichols (1981) used MAC to motivate VOF, where only one extra value is needed per cell, but it can still deal with topological changes.

This value is the volume fraction, the fraction of the cell that is filled by water, which gets advected with the velocity of the water. The advection can be done algebraically or geometrically (Mirjalili et al., 2017). While this method deals automatically with topological changes, it does not keep the interface as sharp as a surface-based method.

- Riemann solvers. Within the context of a finite volume scheme, Kelecy and Pletcher (1997) modelled the air-water interface using a contact wave in a Riemann solver, marking a discontinuity between the density or the volume fraction. This has the advantage of automatically capturing the location of the surface. However, it would require that the background scheme uses Riemann solvers, which makes it less widely applicable than the other methods, and thus there is less evidence on its effectiveness.

Ultimately, it is easier to deal with topological changes of the free surface using the volume-based methods (MAC, VOF, Riemann solvers) than the surface-based methods (fixed lid, height functions, level set, front tracking, surface fitted). This is a significant advantage for the types of flow considered in this thesis: complex 3D flows around in-stream obstacles where we do not know what the free surface will look like in advance. However, MAC requires significant computational resources to deal with the marker particles defining the fluid region. VOF and Riemann solvers bypass this requirement while retaining the flexibility of a volume-based method. Since VOF is used by OpenFOAM, which we use in Chapter 4 to model leaky barriers, let us now outline the method in more detail. The Riemann solver method will be considered in more detail in Chapters 5–6, where it is used to develop a new CFD solver.

3.4.1 Governing equations

The idea of VOF is to use a characteristic (or indicator, or colour) function χ, defined as

$$\chi(\mathbf{x}, t) = \begin{cases} 1 & \text{if there is water at point } \mathbf{x} \text{ at time } t \\ 0 & \text{otherwise.} \end{cases} \tag{3.66}$$

If a computational cell Ω contains both air and water, then it will have an average value of χ somewhere between 0 and 1. Formally, define the phase fraction α to be the integral average of χ, that is,

$$\alpha(\Omega, t) = \overline{\chi} = \frac{1}{|\Omega|} \iiint_\Omega \chi \, dV \tag{3.67}$$

and then $\alpha = 0$ for cells filled with air, $\alpha = 1$ for cells filled with water, and $0 < \alpha < 1$ for cells containing a free surface.

The velocity vector field is defined over the whole domain, not just in the parts that are water. If we want the velocity of the water in a particular computational cell, we have to consider the phasic average $\overline{\mathbf{u}}_{\text{water}}$, defined by Drew (1983) to be

$$\overline{\mathbf{u}}_{\text{water}} = \frac{\overline{\mathbf{u}\chi}}{\alpha} = \frac{1}{|\Omega|\alpha} \iiint_\Omega \mathbf{u}\chi \, dV. \tag{3.68}$$

and likewise for air:

$$\overline{\mathbf{u}}_{\text{air}} = \frac{\overline{\mathbf{u}(1-\chi)}}{1-\alpha} = \frac{1}{|\Omega|(1-\alpha)} \iiint_\Omega \mathbf{u}(1-\chi) \, dV. \tag{3.69}$$

Then, for a particular computational cell,

$$\overline{\mathbf{u}} = \alpha\overline{\mathbf{u}}_{\text{water}} + (1-\alpha)\overline{\mathbf{u}}_{\text{air}}. \tag{3.70}$$

From a simple examination of the VOF literature, it is unclear what the governing equation for VOF is. Indeed, for each of the four following equations, there is at least one source in the literature that claims it to be true:

$$\frac{\partial \chi}{\partial t} + \nabla \cdot (\mathbf{u}\chi) = 0 \tag{3.71}$$

(Cifani et al., 2016; Marek et al., 2008; Mirjalili et al., 2017; Rider and Kothe, 1998; Scardovelli and Zaleski, 1999, 2003; Sethian and Smereka, 2003; Ubbink, 1997; Yeoh and Tu, 2010),

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\alpha) = 0 \tag{3.72}$$

(Chen and Hagen, 2011; Gopala and van Wachem, 2008; Ketabdari, 2016; Marić et al., 2013; Olsen, 2015; Rudman, 1997),

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\alpha) + \nabla \cdot ((\mathbf{u}_{\text{water}} - \mathbf{u}_{\text{air}})\alpha(1-\alpha)) = 0 \tag{3.73}$$

(Berberović et al., 2009; Cifani et al., 2016), and

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\overline{\mathbf{u}}\alpha) + \nabla \cdot ((\overline{\mathbf{u}}_{\text{water}} - \overline{\mathbf{u}}_{\text{air}})\alpha(1-\alpha)) = 0 \tag{3.74}$$

(Rusche, 2002). Let us investigate these equations further.

If we follow an individual parcel of water or air over time and space, its value of χ will not change, and so

$$\frac{D\chi}{Dt} = 0, \tag{3.75}$$

which can be derived from first principles using test functions (Drew, 1983). This is equivalent to (3.71) because of incompressibility. To find out which of (3.72), (3.73), and (3.74) are true, we need to manipulate (3.71) so that it is in terms of α rather than χ. Take the integral average of (3.71) over the computational cell Ω:

$$\frac{1}{|\Omega|}\iiint_\Omega \frac{\partial \chi}{\partial t}\, dV + \frac{1}{|\Omega|}\iiint_\Omega \nabla \cdot (\mathbf{u}\chi)\, dV = 0, \tag{3.76}$$

and we want to switch the order of differentiation and integration. Unfortunately, the characteristic function χ is discontinuous at the air-water interface. However, Drew (1983) suggests that, in this context, Leibniz's rule can be used for the time derivative and Gauss's rule for the space derivative. Then

$$\frac{\partial}{\partial t}\left(\frac{1}{|\Omega|}\iiint_\Omega \chi\, dV\right) + \nabla \cdot \left(\frac{1}{|\Omega|}\iiint_\Omega \mathbf{u}\chi\, dV\right) = 0. \tag{3.77}$$

Use (3.67) for the first term and (3.68) for the second:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}_{\text{water}}\alpha) = 0. \tag{3.78}$$

Rusche (2002, p.100) calls this the conditionally averaged continuity equation. Therefore,

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}\alpha) = \frac{\partial \alpha}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}_{\text{water}}\alpha) - \nabla \cdot (\bar{\mathbf{u}}_{\text{water}}\alpha) + \nabla \cdot (\bar{\mathbf{u}}\alpha) \tag{3.79}$$

$$= -\nabla \cdot (\bar{\mathbf{u}}_{\text{water}}\alpha) + \nabla \cdot (\bar{\mathbf{u}}\alpha) \tag{3.80}$$

$$= \nabla \cdot ((\bar{\mathbf{u}} - \bar{\mathbf{u}}_{\text{water}})\alpha). \tag{3.81}$$

Use (3.70) to get

$$(\bar{\mathbf{u}} - \bar{\mathbf{u}}_{\text{water}})\alpha = \alpha(\alpha\bar{\mathbf{u}}_{\text{water}} + (1-\alpha)\bar{\mathbf{u}}_{\text{air}} - \bar{\mathbf{u}}_{\text{water}}) \tag{3.82}$$

$$= \alpha(1-\alpha)(\bar{\mathbf{u}}_{\text{air}} - \bar{\mathbf{u}}_{\text{water}}), \tag{3.83}$$

which can be substituted into (3.81) to get (3.74). To conclude, equations (3.71) and (3.74) are consistent and thus the most appropriate to be used in the VOF method. However, often (3.73) is written instead of (3.74) in the OpenFOAM literature and guidance, which makes sense as values are averaged in the finite volume method anyway. Therefore, for clarity we will refer to (3.73) instead of (3.74).

3.4.2 Algebraic method

Algebraic VOF is based on (3.73). If a VOF paper takes (3.72) to be true, then it may use the term "artificial compression"[1] to refer to the quantity $\nabla \cdot ((\mathbf{u}_{\text{water}} - \mathbf{u}_{\text{air}})\alpha(1-\alpha))$; however, it is not artificial at all but derived naturally from (3.71). The problem now is that we need a numerical method to approximate the relative velocity

$$\mathbf{u}_{\text{r}} = \mathbf{u}_{\text{water}} - \mathbf{u}_{\text{air}}, \tag{3.84}$$

which is where the artificiality comes in (Cifani et al., 2016).

3.4.3 Geometric method

Geometric VOF is based on (3.71). Using Gauss's theorem, we can manipulate (3.76) into

$$\frac{\partial}{\partial t}(|\Omega|\alpha) + \iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n})\chi \, dS = 0 \tag{3.85}$$

$$\frac{\partial \alpha}{\partial t} + \frac{1}{|\Omega|} \iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n})\chi \, dS = 0 \tag{3.86}$$

$$\frac{\partial \alpha}{\partial t} = -\frac{1}{|\Omega|} \iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n})\chi \, dS. \tag{3.87}$$

where $\partial\Omega$ is the boundary of the computational cell Ω. Integrate (3.87) over the time interval $[t_0, t_1]$ to get

$$\int_{t_0}^{t_1} \frac{\partial \alpha}{\partial t} \, dt = -\frac{1}{|\Omega|} \int_{t_0}^{t_1} \left(\iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n})\chi \, dS \right) dt. \tag{3.88}$$

The Fundamental Theorem of Calculus says that

$$\int_{t_0}^{t_1} \frac{\partial \alpha}{\partial t} \, dt = \alpha(\Omega, t_1) - \alpha(\Omega, t_0), \tag{3.89}$$

which can be substituted into (3.88) to get

$$\alpha(\Omega, t_1) - \alpha(\Omega, t_0) = -\frac{1}{|\Omega|} \int_{t_0}^{t_1} \left(\iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n})\chi \, dS \right) dt \tag{3.90}$$

$$\alpha(\Omega, t_1) = \alpha(\Omega, t_0) - \frac{1}{|\Omega|} \int_{t_0}^{t_1} \left(\iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n})\chi \, dS \right) dt. \tag{3.91}$$

[1]This is not the same as artificial compressibility, which is used in Chapters 5–6.

Roenby et al. (2016, p.3) note that (3.91) is still exact. However, approximations must be introduced for practical purposes. Indeed, to update the volume fraction α in the computational cell Ω from time t_0 to time t_1, we need:

1. The value of α for the computational cell Ω at the time t_0
2. The geometry of the computational cell Ω and its boundary faces f
3. The normal vector field $\mathbf{n}$ of the boundary faces f
4. The velocity vector field $\mathbf{u}$
5. The characteristic function χ, a scalar field

While requirements 1–3 are known, requirements 4–5 must be approximated.

Interface reconstruction

We have a volume fraction α in each computational cell but, to progress any further with (3.91), we will need an approximation of the characteristic function χ. This requires us to guess the location of the air-water interface. Generally, we assume that the local curvature of the fluid is larger than the mesh size (Roenby et al., 2016, p.4), and so approximate each cell's air-water interface by a straight line (in the 2D case) or plane (in the 3D case). Any interface reconstruction algorithm needs to adhere to two requirements. First, the interface should split the cell so that the volume fraction α is retained. Second, the interface should be oriented such that "the nearest fluid exits first" (Noh and Woodward, 1976, p.11) in the advection step, which requires looking at the volume fraction of at least some of the surrounding cells. There are algorithms that meet this task with varying levels of success and computational complexity (Figure 3.11).

Less computationally expensive algorithms include the Simple Line Interface Calculation (SLIC) method of Noh and Woodward (1976) and the method of Hirt and Nichols (1981), where lines are reconstructed parallel to the grid. The SLIC method has a sweep in each co-ordinate direction, reconstructing the interface depending on the direction we look at it from, while the Hirt-Nichols algorithm uses a larger stencil to generate one orientation for each interface.

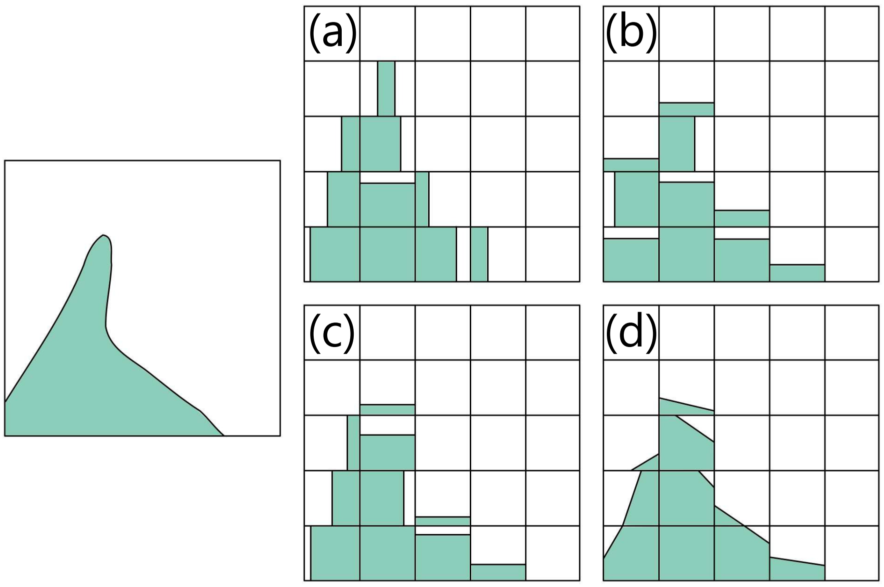

Figure 3.11: Interface reconstruction methods: (a) SLIC x-sweep, (b) SLIC y-sweep, (c) Hirt-Nichols and (d) Youngs reconstruction algorithms in the 2D case, based on Rudman (1997, p.676). The actual fluid configuration is shown on the left for comparison only; it is the volume fractions (numbers for each cell) that are available to the algorithms.

The Piecewise Linear Interface Calculation (PLIC) method goes a step further than the Hirt-Nichols algorithm, this time allowing interface orientations that are not parallel to the axes. The idea is to find the orientation of the surface, defined by its normal vector, and then slide this surface along the cell so the point where it divides the cell into the correct volume fractions. The creative problem is finding the normal vector, and there are different ways to do this (Youngs, 1982), but the computational problem arises in the advecting the "flux polyhedron" (Marić et al., 2013) as it depends on the local velocity.

As SLIC is too simple and PLIC too complex, it is tempting to look for a method that is somewhere between SLIC and PLIC in complexity. One proposed solution is to compute a SLIC flux for each direction and take a weighted average depending on the PLIC orientation (Marek et al., 2008). This removes some of the geometrical complexity for 3D simulations, but still provides better results than SLIC.

Recently, Roenby et al. (2016) developed a more sophisticated method than SLIC or PLIC, inspired by the popular post-processing technique of clipping the fluid at $\alpha = 0.5$ to show only the water phase. They took this idea into the interface reconstruction step of the solver itself, calling it isoAdvector. There are two parts of the interface reconstruction step: finding interfaces for different values of α (i.e. not just $\alpha = 0.5$), and then interpolating between the best two interfaces (determined by how well they split up the cell into the correct volume fraction α) to get a more optimal interface.

In conclusion, despite its apparent simplicity, reconstructing the interface at the cell level is an active area of research. We are not limited to methods designed decades ago, and we can use methods that harness improvements in computational resources.

Interface advection

Now we have some idea where the water lies in each computational cell, we can calculate the net volume of water crossing the boundary $\partial\Omega$ over the time interval $[t_0, t_1]$, that is,

$$\int_{t_0}^{t_1} \left(\iint_{\partial\Omega} (\mathbf{u} \cdot \mathbf{n}) \chi \, dS \right) dt \tag{3.92}$$

from (3.91). Marić et al. (2013) call this quantity the "fluxed phase volume." Decompose $\partial\Omega$ into a disjoint union of faces f to get

$$\sum_{f \in \partial\Omega} \int_{t_0}^{t_1} \left(\iint_{f} (\mathbf{u} \cdot \mathbf{n}) \chi \, dS \right) dt, \tag{3.93}$$

which is still exact.

In the 1D case, the cell Ω would be an interval, and so its boundary $\partial\Omega$ would be decomposed into the two endpoints—work out the flux at each one. The problem is that the VOF method is used for 2D and 3D simulations. So, one option is to split these higher dimension cases into 1D sweeps, moving the fluid first in one dimension and then the other(s). This seems similar to the SLIC interface reconstruction method, and indeed SLIC must be used with dimensional splitting, but the other interface reconstruction methods can be used with either split or unsplit advection. Interface reconstruction gives an approximate value for the indicator function χ everywhere, but we only have cell averages for the velocity $\mathbf{u}$. The challenge is to find a way to combine these two together to get the fluxed phase volume. Note that interface advection schemes tend to assume that the velocity $\mathbf{u}$ is constant over time and space in each computational cell, which means that the relative velocity $\mathbf{u}_r$ is assumed to be zero.

Examples of interface advection schemes include the split donor-acceptor method (Hirt and Nichols, 1981), unsplit Lagrangian backtracking for piecewise linear interfaces in arbitrary unstructured grids (Marić et al., 2013), and unsplit advection of iso-surfaces (Roenby et al., 2016). For the latter, recall that the flux across the cell interface f is calculated by interpolating the velocity to the cell interface to get $\mathbf{u}_f$, and then dotting it with the surface area vector $\mathbf{S}_f$, that is,

$$\varphi_f \approx \mathbf{u}_f \cdot \mathbf{S}_f = (\mathbf{u}_f \cdot \mathbf{n}_f) S_f, \tag{3.94}$$

where $S_f = |\mathbf{S}_f|$ is the area of the face f. The approximation comes because $\mathbf{u}_f$ is interpolated. This allows us to say, rather trivially,

$$\iint_f \mathbf{u} \cdot \mathbf{n} \, dS \approx \iint_f \frac{\varphi_f}{S_f} \, dS = \frac{\varphi_f}{S_f} \iint_f dS = \varphi_f. \tag{3.95}$$

However, Roenby et al. (2016) use this line of reasoning to suggest that

$$\iint_f (\mathbf{u} \cdot \mathbf{n}) \chi \, dS \approx \iint_f \frac{\varphi_f}{S_f} \chi \, dS = \frac{\varphi_f}{S_f} \iint_f \chi \, dS, \tag{3.96}$$

which is only valid if $\mathbf{u}$ is constant over space and so $\mathbf{u}_r = 0$. They also assume that $\mathbf{u}$ is constant over the time step t_0 to t_1 to get

$$\int_{t_0}^{t_1} \left(\iint_f (\mathbf{u} \cdot \mathbf{n}) \chi \, dS \right) dt \approx \frac{\varphi_f}{S_f} \int_{t_0}^{t_1} \left(\iint_f \chi \, dS \right) dt. \tag{3.97}$$

However, they do consider the interaction between $\mathbf{u}$ and χ by considering how the submerged face area

$$\iint_f \chi \, dS \tag{3.98}$$

changes over time as the upwind isosurface moves with $\mathbf{u}$. This involves interpolating velocities to the centres of the isofaces and then using this to approximate the intersection between the air-water interface and the cell interface.

3.4.4 Surface tension

Surface tension acts against the conservation of momentum in the source term

$$\iint_\Gamma \sigma \kappa \delta \hat{\mathbf{n}} \, dS \tag{3.99}$$

where Γ is the air-water interface, σ the surface tension coefficient, κ the local interfacial curvature, and δ a 3D Dirac delta function non-zero only at the air-water interface. A clear benefit for a surface-based method over a volume-based method is that the curvature is already given and does not need to be approximated. However, in a volume-based method like VOF, poor approximations of the curvature can lead to problems with spurious currents. One option is to approximate the curvature κ by

$$-\nabla \cdot \hat{\mathbf{n}}, \tag{3.100}$$

where $\hat{\mathbf{n}}$ is the normalised gradient of the volume fraction α. The derivative of a Heaviside step function is the Dirac delta function and so we can replace (3.99) by

$$\iiint_\Omega \sigma\kappa\nabla\alpha \, dV. \tag{3.101}$$

The Continuum Surface Force (CSF) model of Brackbill et al. (1992) can be used to approximate $\nabla\alpha$ by smoothing out the discontinuity of α (Deshpande et al., 2012; Cifani et al., 2016).

3.5 Conclusion

This chapter covered the theoretical background for CFD modelling, from the governing equations and turbulence to discretisation methods and ways to capture a free surface. The VOF method is discussed in some detail since that is the method used in OpenFOAM. In Chapter 4, we describe how these standard methods are implemented in OpenFOAM and use them to model leaky barriers. In Chapters 5–6, we go on to implement an alternative CFD solver based on the (density-based) method of artificial compressibility instead of the (pressure-based) PIMPLE algorithm.

Chapter 4. Modelling leaky barriers with `interFoam`

OpenFOAM (Weller et al., 1998) is an open-source library for Computational Fluid Dynamics (CFD). As it is open-source, we can look into the source code to check what it is doing. Furthermore, we can extend and develop the source code without having to write a new CFD library from scratch. This chapter outlines the existing functionality in OpenFOAM with reference to the theory presented in Chapter 3, explains new practical functionality added as part of this thesis, and then presents an application of OpenFOAM to model leaky barriers. This is the first study to use a non-fixed-lid approach for modelling leaky barriers in 3D. Later chapters will go further, with Chapter 5 presenting a new numerical multiphase solver and Chapter 6 implementing it in OpenFOAM. Note that there are multiple official forks of OpenFOAM; this thesis uses ESI-OpenCFD v1906 (OpenCFD, 2019a,b).

4.1 Existing OpenFOAM functionality

This section provides a high-level explanation of how the theory in Chapter 3 is implemented in the OpenFOAM software. The specific names of the functions, schemes, and algorithms are given so that it is clear how they can be accessed in practice by users of OpenFOAM, with a focus on the methods used later in the chapter to model leaky barriers. Note that there are three main dictionary files that are used to set the specific numerical schemes in OpenFOAM—`controlDict` for time steps and outputs, `fvSchemes` for discretisation schemes, and `fvSolution` for matrix solvers—and these are also mentioned in this section for the sake of reproducibility.

OpenFOAM is written in C++ as this programming language provides object orientation with templated classes (OpenCFD, 2019a). For example, there is a class for fields, with sub-classes for scalar and vector fields, and further sub-classes for scalar and vector fields defined on 2D surfaces and 3D volumes. This makes the low-level code easier to navigate and maintain.

Meanwhile, the high-level code benefits from operator overloading to make it clearer what equations are being solved. For instance, the solver `icoFoam` uses the PISO algorithm described in Section 3.3.6, with the file `icoFoam.C` containing the code

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nu, U)
);

if (piso.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p));
}
```

which solves the momentum predictor $\mathbf{Mu} = -\nabla p$. The `fvm` (finite volume matrix) and `fvc` (finite volume calculate) namespaces are used to define implicit and explicit terms respectively. Overall, this flexibility makes it relatively easy to create new solvers in OpenFOAM. It is more difficult to create new solvers that do not follow the standard methods used in OpenFOAM. For example, OpenFOAM favours pressure-based solvers like those outlined in Section 3.3.6, so it is more difficult to implement density-based solvers like the Godunov-type schemes used in Chapters 2 and 5–6. To implement such a solver, substantial amounts of code have to be written as the numerics do not already exist in the low-level source code.

4.1.1 Discretisation

As mentioned above, the governing equations are written in the solvers with high-level code which, thanks to operator overloading in C++, is very readable. For example, in the `icoFoam` code snippet, there are terms called `ddt`, `div`, `laplacian`, and `grad` belonging to the `fvm` and `fvc` namespaces. The discretisation methods for these terms are separated from the solver, instead determined at run time by reading the dictionary `fvSchemes` that contains a sub-dictionary for each type of scheme. This allows the user to customise the numerical setup to their own needs. A discussion for each of these sub-dictionaries follows, drawing on OpenFOAM-specific guidance from the source code, Kärrholm and Tao (2008), Tabor (2018), Guerrero (2019), OpenCFD (2019b), and Greenshields and Weller (2022), as well as theory introduced in Section 3.3.

`ddtSchemes`

OpenFOAM has several methods for evaluating the time derivative (3.40). The first-order method is called `Euler`, given by

$$\frac{\partial \theta}{\partial t} = \frac{\theta - \theta^{o}}{\Delta t} \tag{4.1}$$

where θ is the updated value and θ^{o} the value from the previous time step. One second-order method is called `backward`, given by

$$\frac{\partial \theta}{\partial t} = \frac{3\theta - 4\theta^{o} + \theta^{oo}}{2\Delta t} \tag{4.2}$$

where θ^{oo} is the value from before the last time step. Another second-order option is `CrankNicolson`, a modification of the Crank-Nicolson method (Crank and Nicolson, 1947).

There are two options for steady-state simulations. First, the `steadyState` method should be used for steady-state solvers. Second, if the solver is transient but it is only the steady state to which the solver converges that is of interest, then the `localEuler` method can be used to speed up the simulation time. This is where the Δt value is evaluated at each cell and the solution is not time accurate at all, but it converges to the correct steady state. As OpenFOAM's multiphase solvers are all transient, `steadyState` cannot be used for them but `localEuler` can, which is important for the simulations of leaky barriers in this chapter as the experimental validation data are steady state.

`divSchemes`

There is a trade off between stability and accuracy when calculating the divergence term (3.41) for convective derivatives, which is approximated by

$$\sum_{f \in \partial\Omega} \varphi_f \theta_f \tag{4.3}$$

where $\varphi_f = (\rho\mathbf{u})_f \cdot \mathbf{S}_\mathrm{f}$ is the flux and θ the advected quantity. The term θ_f is calculated at a cell interface by interpolating the advected quantity θ between the cells adjacent to that interface as described in Section 3.3.3. The first-order, stable choice is `upwind`; take the term from the upwind cell. The second-order, unbounded choice is `linear`; take a central difference between the two cells. While `upwind` may be acceptable for turbulence parameters as turbulence is diffusive by nature (Guerrero, 2019), neither `upwind` nor `linear` is ideal as we desire both stability and accuracy.

An alternative lies in TVD schemes that interpolate between the owner and neighbour cells in such a way that no extrema are introduced or exacerbated. As described in Section 3.3.3, this is done through a limiter $\psi(r)$ where r is an unstructured generalisation of the ratio between the upwind and downwind gradients, and $\psi(r)$ acts as a blending function between the `upwind` and `linear` schemes. TVD schemes available in OpenFOAM include `Minmod`, `limitedLinear`, `vanLeer`, `MUSCL` (distinct from the MUSCL reconstruction used in Chapters 2 and 5–6), and `SuperBee`. The values of $\psi(r)$ for these schemes are plotted on a Sweby diagram in Figure 4.1. On 3D irregular meshes, such as those used in this chapter to model leaky barriers, there are more likely to be oscillations when $\psi(r)$ approaches 2. Therefore, `Minmod`, `limitedLinear`, and `vanLeer` are the more stable TVD schemes. However, values of $\psi(r)$ nearer 0 will cause diffusion, and so `Minmod` is a diffusive choice.

Another alternative is `linearUpwind`, a second-order upwind scheme. As the name suggests, this involves linearly interpolating from the upwind cell based on the cells upwind of that. As there is some ambiguity interpolating on an unstructured mesh, this is done indirectly via the gradient of the upwind cell (Jasak et al., 1999), and so a method for calculating the gradient has to be specified alongside `linearUpwind`. While this scheme is less problematic than the `linear` scheme, it still suffers from unboundedness.

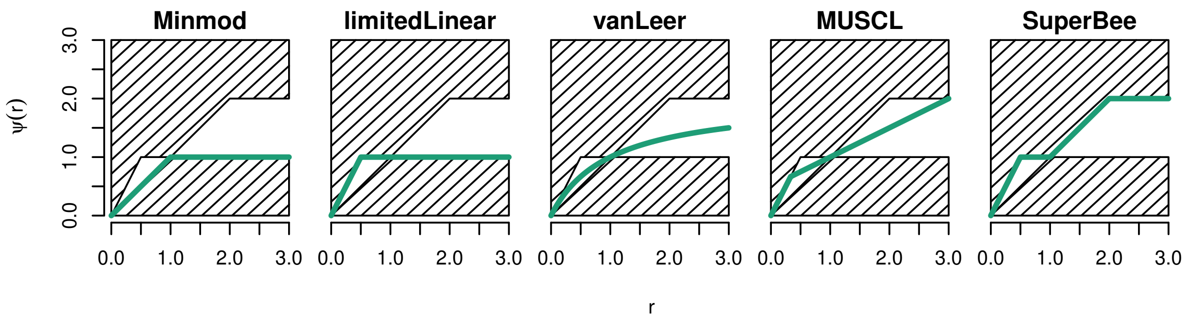

Figure 4.1: Selected second-order TVD schemes available in OpenFOAM, plotted on a Sweby diagram where the unshaded region represents the second-order TVD region (Sweby, 1984, p.1001).

`laplacianSchemes`

Laplacian terms (3.42) are approximated by

$$\sum_{f \in \partial\Omega} (\Gamma \nabla \theta)_f \cdot \mathbf{S}_f \tag{4.4}$$

where Γ is the diffusion coefficient and θ the scalar to be diffused. This requires an interpolation method for calculating the diffusion coefficient Γ_f (`interpolationScheme`) and a method for calculating the surface normal gradient $\nabla\theta_f \cdot \mathbf{S}_f$ (`snGradScheme`). The interpolation is usually chosen to be `linear`. The scheme for calculating the surface normal gradient is more complicated. It should take into account the mesh quality, specifically the orthogonality as discussed in detail in Section 3.3.4. This is important because the 3D meshes used in this chapter to model leaky barriers are not Cartesian meshes but unstructured, irregular meshes with cells of different shapes and sizes. While non-orthogonality should be minimised in the creation of the mesh, it cannot be eradicated completely.

Recall that non-orthogonal meshes require splitting the surface vector $\mathbf{S}_f$ into an orthogonal part Δ and a non-orthogonal part $\mathbf{k}$ that is calculated explicitly, as in (3.53):

$$\nabla\theta_f \cdot \mathbf{S}_f = \nabla\theta_f \cdot \Delta + \nabla\theta_f \cdot \mathbf{k}. \tag{4.5}$$

While there are an infinite number of possibilities for defining Δ, the over-relaxed approach given by (3.56) provides the most stability. Consequently, OpenFOAM has the following options for calculating the surface normal gradient:

- `orthogonal`. This scheme is, as the name suggests, for orthogonal meshes that require no non-orthogonal correctors. There is no splitting into Δ and $\mathbf{k}$ and the gradient is simply calculated with (3.52):

$$S_f \frac{\theta_N - \theta_P}{\mathbf{d}}. \tag{4.6}$$

- `corrected`. This scheme is the over-relaxed method for non-orthogonal correction, given by (3.56):

$$\Delta = |\mathbf{S}|^2 \frac{\mathbf{d}}{\mathbf{d} \cdot \mathbf{S}} \tag{4.7}$$

  It elongates both the orthogonal part Δ and the non-orthogonal part k, increasing stability and providing an unbounded, second-order, conservative scheme.

- `uncorrected`. This scheme is a bounded, first-order, non-conservative scheme that does not calculate the non-orthogonal part k at all; it only calculates the orthogonal part Δ. However, it is different from the `orthogonal` scheme as it calculates Δ as if over-relaxation was being used. Consequently, this method is not accurate, but it can be used as a last resort for highly non-orthogonal meshes where explicit non-orthogonal correctors would destabilise the solution. Due to the loss of accuracy, it is better to discard the mesh than use the `uncorrected` method.

- `limited`. This scheme, a blend between the `uncorrected` and `corrected` methods, can be thought of as clipping the explicit non-orthogonal correctors so that they are not larger than the implicit orthogonal parts. It is specified with a blending factor:

$$\begin{cases} 0 & \text{corresponds to } \texttt{uncorrected} \\ 1/3 & \text{non-orthogonal correction} \leq 0.5 \times \text{orthogonal part} \\ 1/2 & \text{non-orthogonal correction} \leq \text{orthogonal part} \\ 1 & \text{corresponds to } \texttt{corrected} \end{cases} \tag{4.8}$$

  as given by Greenshields (2020, Eq.4.2). This scheme is useful when there are a few very non-orthogonal cells that cannot be removed.

`gradSchemes`

The Green-Gauss and least-squares gradient calculation methods described in Section 3.3.2 are implemented in OpenFOAM as `Gauss` and `leastSquares`. Both methods can be limited to make them more stable, either limiting cell-to-cell or face-to-cell values, and either applied multi-dimensionally—to each direction separately—or across all directions. In order of increasing stability but decreasing accuracy, we have: `cellMDLimited`, `cellLimited`, `faceMDLimited`, and `faceLimited`. An examination of the source code shows that the `cellLimited` scheme, in turn, has three options: `minmod`, `Venkatakrishnan`, and `cubic`. These will be explained in more detail in Chapter 6 as the new solver harnesses the gradient schemes for another purpose than calculating a diffusion term.

4.1.2 Matrix solvers

As OpenFOAM uses implicit time integration, the discretised equations get assembled into a large sparse matrix, which is then inverted using iterative matrix solvers, read at run time from the case dictionary `fvSolution`. In the `icoFoam` code snippet, this is all encapsulated in the function called `solve`. In this section, we outline which matrix solvers and corresponding tolerances are suitable for different equations, drawing on OpenFOAM-specific guidance from the source code, Tabor (2018), Guerrero (2019), and OpenCFD (2019b), as well as theory introduced in Section 3.3.5.

The matrix solvers, smoothers, and preconditioners available in OpenFOAM are listed in Tables 4.1–4.3. Which ones should be chosen depends on whether we are solving the pressure (p) or convective ($\mathbf{u}$, k, ϵ, ω) equations. The momentum and scalar equations are asymmetric and also relatively inexpensive to solve compared to pressure, which is symmetric. Generally, `GAMG` works well for pressure, but a more stable choice is `PCG`, which also works particularly well for large cases run on many cores. Meanwhile, `smoothSolver` with `GaussSeidel` or `PBiCGStab` with `DILU` work well for momentum and scalar equations. `PBiCGStab` is generally quicker than `smoothSolver` but more likely to crash.

The choice of matrix solver and preconditioner/smoother gets specified in `fvSolution` alongside tolerances for convergence. Absolute and relative tolerances can be set, with the

Solver	Type	Specified with	Matrix symmetry
`smoothSolver`	solver with smoother	smoother	any
`GAMG`	multigrid	smoother	any
`PCG`	conjugate gradient	preconditioner	symmetric
`PBiCG`	conjugate gradient	preconditioner	asymmetric
`PBiCGStab`	conjugate gradient	preconditioner	any

Table 4.1: Matrix solvers available in OpenFOAM.

Smoother	Matrix symmetry
`DIC`	symmetric
`DICGaussSeidel`	symmetric
`DILU`	asymmetric
`GaussSeidel`	any
`symGaussSeidel`	any

Table 4.2: Smoothers available in OpenFOAM.

Preconditioner	Matrix symmetry
`DIC`	symmetric
`FDIC`	symmetric
`DILU`	asymmetric
`GAMG`	any

Table 4.3: Preconditioners available in OpenFOAM.

iteration stopping once one is met. The tighter the tolerance, the longer the simulation time. As the convective equations are relatively inexpensive to solve, the tolerances can be set as very tight. Pressure is more expensive to solve and, unfortunately, more important to get right as it governs mass conservation. It has to be solved multiple times during the pressure-velocity coupling algorithm at each time step. To improve simulation times, generally its tolerances are set to be less stringent for the preliminary loops compared to the final loop.

4.1.3 Pressure-velocity coupling

OpenFOAM uses the PIMPLE algorithm explained in Section 3.3.6 for pressure-velocity coupling. The terms on the left-hand side of the predictor and correctors are implicit (`fvm` namespace) while the terms on the right-hand side are explicit (`fvc` namepsace)—see the `icoFoam` code snippet above. While this method is simpler to formulate than a fully implicit solver, it lacks the unconditional stability of a fully implicit solver due to the explicit terms on the right-hand side (Elman et al., 2003).

The settings for the PIMPLE algorithm, specified by the user in the `fvSolution` dictionary, should be chosen based on the solver, time step, and mesh quality. Recall that the PIMPLE algorithm has two corrector loops: inner (PISO) loops and, to circumvent the PISO stability limit of CFL < 1, outer (SIMPLE) loops. Note also that the pressure corrector in Figure 3.9 is a Laplacian term. Consequently, this term requires non-orthogonal correction as described Section 3.3.4. Therefore, we have three corrector loops in total, the numbers of which are set in `fvSolution`, with the entry `nCorrectors` corresponding to the inner (PISO) loops, `nOuterCorrectors` to the outer (SIMPLE) loops, and `nNonOrthogonalCorrectors` to the non-orthogonal correctors. Section 4.1.4 outlines how these correctors should be chosen when modelling multiphase flows in particular.

Another important entry in `fvSolution` is `momentumPredictor`. The momentum predictor step in Figure 3.9 is an optional part of the algorithm as the velocity can be simply taken from the last time step instead, and the momentum corrector recalculates the velocity anyway. However, the momentum predictor is useful for convergence in "high speed flows which are dominated more by momentum exchange than mass conservation" (Greenshields and Weller, 2022, p.186). Switching off the momentum predictor is recommended for viscous or multiphase flows where this is not the case.

4.1.4 Volume of fluid solvers

Modelling free-surface flow around leaky barriers requires a multiphase method. OpenFOAM's multiphase solvers use the Volume of Fluid (VOF) method explained in Section 3.4 to deal with interfaces. The classic solver `interFoam` uses algebraic VOF, while the newer solver `interIsoFoam` uses geometric VOF. Several unofficial solvers have also been developed, for example, the geometric VOF solvers `voFoam` (Marić et al., 2013) and `gVoFoam` (Cifani et al., 2016), and the hybrid level-set/VOF solver `lentFoam` (Marić et al., 2015). In the following sections, the two official OpenFOAM solvers `interFoam` and `interIsoFoam` are briefly described with reference to the theory outlined in Section 3.4.

`interFoam` and MULES

Recall that algebraic VOF is based on the VOF equation (3.73):

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\alpha) + \underbrace{\nabla \cdot (\mathbf{u}_r \alpha (1-\alpha))}_{\text{compression term}} = 0 \tag{4.9}$$

where $\mathbf{u}_r$ is the relative velocity (3.84), which needs to be approximated. Deshpande et al. (2012) and Cifani et al. (2016) report that `interFoam` approximates the relative velocity at the cell interface f by

$$(\mathbf{u}_\mathrm{r})_f = \hat{\mathbf{n}}_f \min\left(c_\alpha |\mathbf{u}_f|, \max_{f \in \bigcup \Omega} |\mathbf{u}_f| \right) \tag{4.10}$$

or, equivalently,

$$(\varphi_\mathrm{r})_f = (\hat{\mathbf{n}}_f \cdot \mathbf{S}_f) \min\left(c_\alpha \frac{|\varphi_f|}{|\mathbf{S}_f|}, \max_{f \in \bigcup \Omega} \frac{|\varphi_f|}{|\mathbf{S}_f|} \right) \tag{4.11}$$

where $(\varphi_\mathrm{r})_f$ is the corresponding flux. The right-hand side of (4.10) has two parts: the vector $\hat{\mathbf{n}}_f$ multiplied by the scalar $\min(c_\alpha |\mathbf{u}_f|, \max_{f \in \bigcup \Omega} |\mathbf{u}_f|)$.

The vector part of (4.10), $\hat{\mathbf{n}}_f = (\nabla \alpha)_f / |(\nabla \alpha)_f|$, is the normalised gradient of α at the cell interface, where the gradient $(\nabla \alpha)_f$ is approximated using the central difference between the owner and neighbour cells. As the vector $\hat{\mathbf{n}}_f$ approximates the direction of greatest change of α, the vector $\hat{\mathbf{n}}_f$ is perpendicular to the air-water interface and pointing towards the water.

The scalar part of (4.10), $\min(c_\alpha|\mathbf{u}_f|, \max_{f\in\bigcup\Omega}|\mathbf{u}_f|)$, determines how great the magnitude of $(\mathbf{u}_\mathrm{r})_f$ should be and does not affect the direction. It contains the factor c_α, called `cAlpha` in the `fvSolution` dictionary, which describes how much compression we want:

- If $c_\alpha = 0$, then there is no compression term.
- If $c_\alpha = 1$, then the relative velocity $(\mathbf{u}_\mathrm{r})_f$ has the same magnitude as the cell interface velocity $\mathbf{u}_f$.
- If $0 < c_\alpha < 1$, then there is compression and $(\mathbf{u}_\mathrm{r})_f$ has a smaller magnitude than $\mathbf{u}_f$.
- If $c_\alpha > 1$, then $(\mathbf{u}_\mathrm{r})_f$ has a greater magnitude than $\mathbf{u}_f$, but not so high that it exceeds the maximum velocity in the whole domain.

Generally c_α is taken to be equal to 1. However, an examination of the source code suggests that while (4.11) is implemented in the slightly different solver `multiphaseInterFoam`, the solver `interFoam` calculates

$$(\varphi_\mathrm{r})_f = (\mathbf{n}_f \cdot \mathbf{S}_f)c_\alpha\frac{|\varphi_f|}{|\mathbf{S}_f|} \tag{4.12}$$

instead.

In solving the VOF equation (4.9) with the finite volume method, specifically Gauss's theorem, `interFoam` has to interpolate α to the cell interfaces to obtain a flux. It does this using a form of Flux Corrected Transport (FCT) called Multidimensional Universal Limiter for Explicit Solution (MULES), where the flux of alpha φ_α is split into different parts—the diffusive part φ_α^{UD}, the higher-order part φ_α^{HO}, and the compressive part φ_α^{C}—with different weightings. In semi-discretised form,

$$\frac{\partial\alpha}{\partial t} + \sum_f \varphi_\alpha = 0 \tag{4.13}$$

where

$$\varphi_\alpha = \varphi_\alpha^{UD} + \lambda\varphi_\alpha^{\mathrm{corr}} \tag{4.14}$$

is comprised of the diffusive part φ_α^{UD} and the anti-diffusive part

$$\varphi_\alpha^{corr} = \varphi_\alpha^{HO} + \varphi_\alpha^{C} - \varphi_\alpha^{UD}, \tag{4.15}$$

and λ is a limited weighting factor to be determined iteratively. Now, φ_α^{UD} solves the fluxes with an upwind scheme and φ_α^{HO} solves the fluxes with a higher-order scheme. The higher-order scheme is set using the entry for `div(phi,alpha)` in the `fvSchemes` dictionary, usually `vanLeer`. Note that both the φ_α^{UD} and φ_α^{HO} terms do not include the interface compression term in (4.9). This compression term is what makes up φ_α^{C}. It is calculated using the entry for `div(phirb,alpha)` in the `fvSchemes` dictionary, usually `linear`. It can be higher-order and unbounded as MULES uses the limiter λ to keep α within physical limits (Deshpande et al., 2012; Damián, 2013; Almeland, 2018).

As the name suggests, the default form of MULES is explicit, but there is also a semi-implicit form that allows arbitrarily large time steps. Which one is chosen is controlled by the `MULESCorr` entry in the `fvSolution` dictionary. The semi-implicit form is activated when `MULESCorr` is set to `yes`, and it splits the operators, calculating the upwind part implicitly (Almeland, 2018).

Pressure and velocity are coupled with the PIMPLE algorithm. Recall again that outer loops allow Courant numbers larger than 1. However, such outer loops would be pointless for explicit MULES as the Courant number for α—set using the `maxAlphaCo` entry in `controlDict`—must be lower than 0.25 anyway. The Courant number can instead be increased for explicit MULES by introducing sub-cycles of the VOF equation using the `nAlphaSubCycles` parameter in the `fvSolution` dictionary (Berberović et al., 2009). This is not the case for semi-implicit MULES, where the Courant number can be arbitrarily large. Instead of sub-cycling the VOF equation, correctors should be introduced using the `nAlphaCorr` entry in the `fvSolution` dictionary. These correctors employ fixed-point iteration to overcome the flux non-linearity problem (Damián, 2013).

Finally, `interFoam` uses the CSF model outlined in Section 3.4.4 to model surface tension. The coefficient σ is specified in the `transportProperties` dictionary. Unfortunately, as mentioned in Section 3.4.4, this requires an approximation of the surface curvature, and poor approximations can lead to problems with spurious currents (Deshpande et al., 2012).

`interIsoFoam` and isoAdvector

The OpenFOAM solver `interIsoFoam` reconstructs the air-water interface using iso-surfaces and advects the interface using (3.91) and (3.97):

$$\alpha(\Omega, t_1) = \alpha(\Omega, t_0) - \frac{1}{|\Omega|}\int_{t_0}^{t_1}\left(\iint_{\partial\Omega}(\mathbf{u}\cdot\mathbf{n})\chi\, dS\right) dt \tag{4.16}$$

$$\approx \alpha(\Omega, t_0) - \frac{1}{|\Omega|}\frac{\varphi_f}{S_f}\int_{t_0}^{t_1}\left(\iint_{f}\chi\, dS\right) dt. \tag{4.17}$$

This interface capturing method is called isoAdvector (Roenby et al., 2016), and it replaces MULES in the `interFoam` solver to obtain the `interIsoFoam` solver. As `interIsoFoam` shares much of its general structure with `interFoam`, it requires similar dictionary entries, including `nAlphaCorr` and `nAlphaSubCycles` in `fvSolution`. However, further settings can be specified in `fvSolution` to constrain the geometrical operations in isoAdvector, and a `fluxRequired` entry must be specified in `fvSchemes`.

While the inner workings of isoAdvector are documented better than MULES, there is not as much practical guidance or evaluation of results for `interIsoFoam` as there is for `interFoam`. In the original paper by Roenby et al. (2016), the developers only compared the ability of isoAdvector and MULES to deal with pure advection. They first note that the isoAdvector algorithm does not make much sense for Courant numbers significantly higher than 1, but isoAdvector is stable for Courant numbers less than 1, which is much better than the constraints for explicit MULES. The developers then replaced MULES by isoAdvector in `interFoam` (Roenby et al., 2018), hoping that its good performance in pure advection at larger time steps would translate well into good performance in the solver at larger time steps. Unfortunately, this was not the case (Roenby et al., 2017). Indeed, while isoAdvector keeps the interface sharper than MULES, it has more extreme wave dampening and phase error. The developers suggest that this might be due to the coupling of isoAdvector with PIMPLE, which is currently very simple. They are looking to use some of the extra information gained with isoAdvector to improve how PIMPLE deals with large density jumps—see, for instance, the pre-print by Vukčević et al. (2018).

Consequently, `interFoam` is currently more reliable than `interIsoFoam`, even if it suffers from the numerical diffusion that `interIsoFoam` is known to eradicate.

4.1.5 Boundary conditions

The boundary of the mesh is comprised of several regions, called patches, which allow different parts of the mesh boundary to have different boundary conditions (Greenshields and Weller, 2022). For instance, there may be an inlet patch, an outlet patch, a wall patch, or a symmetry patch in the mesh boundary, and each one gets a different boundary condition. There are two main types of boundary condition in OpenFOAM: basic and derived (Moukalled et al., 2016). Basic boundary conditions include Dirichlet (`fixedValue`), Neumann (`fixedGradient` or `zeroGradient`), and Robin (`mixed`). Derived boundary conditions are wrappers around basic boundary conditions, allowing easy use of them in different types of simulations, as described below for open-channel flow.

The OpenFOAM source code includes many tutorials, simple cases designed to illustrate OpenFOAM's vast functionality. A popular method for setting up OpenFOAM simulations is to pick the tutorial most similar to the intended application, and then modify it incrementally until the desired geometry and numerics are attained. As we are interested in modelling free-surface flows around leaky barriers, the `weirOverflow` tutorial is appropriate for this purpose. It is a simple open-channel flow case using the VOF solver `interFoam`. In this section, we outline the standard boundary conditions for each boundary patch in the `weirOverflow` tutorial as shown in Figure 4.2, with discussions based on the source code. The parameters of interest are the velocity $\mathbf{u}$, pressure p, phase fraction α, the transported turbulence parameters k and ϵ or ω, and the turbulent viscosity ν_t.

Inlet

We want to be able to set a fixed flow rate at the inlet. This can be done using the `variableHeightFlowRateInletVelocity` derived boundary condition for velocity, which is used in VOF simulations to set the inlet velocity to a fixed value so that the prescribed flow rate of the water phase is reached. More specifically, it calculates the cross-sectional

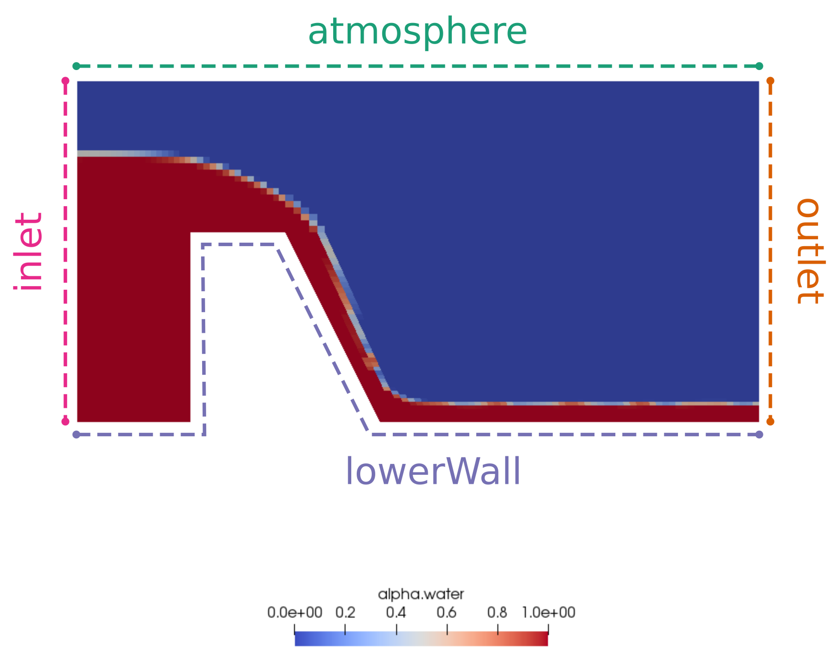

Figure 4.2: Boundary patches in `weirOverflow` tutorial case. Flow from left to right.

area of the water phase using the α field from the previous time step. Then it calculates an average velocity by dividing the desired flow rate by the cross-sectional area. This is then set as a `fixedValue` condition across the water phase by multiplying by α. Consequently, the velocity is set to be zero across the air phase. The corresponding `variableHeight-FlowRate` boundary condition for the phase fraction is

$$\begin{cases} \texttt{fixedValue } 0 & \text{if } \alpha < 0 \\ \texttt{zeroGradient} & \text{if } 0 < \alpha < 1 \\ \texttt{fixedValue } 1 & \text{if } \alpha > 1 \end{cases} . \tag{4.18}$$

Meanwhile, the boundary condition for pressure is `zeroGradient`, the transported turbulence parameters `fixedValue`, and the turbulent viscosity `calculated`, that is, already determined from the other parameters.

Outlet

We want the fluid to be able to leave the domain easily at the outlet. This is generally done by setting velocity, pressure, and the phase fraction to be `zeroGradient`, the transported turbulence parameters `inletOutlet`, while the turbulent viscosity `calculated`. The `inletOutlet` boundary condition is useful for when there may be reverse flow; it uses `zeroGradient` when it is acting as an outlet, but `fixedValue` when there is reverse flow. Practical experience has shown that this is an unsatisfactory set up for general open-channel flow outlets as it does not constrain the downstream depth, which is discussed further in Sections 4.1.9–4.2.1.

Atmosphere

Special consideration must be taken for pressure and velocity on an atmosphere patch. Generally, velocity is `pressureInletOutletVelocity` which, like `inletOutlet`, is `zeroGradient` when acting as an outlet, but when it is acting as an inlet it is only the tangential components of the velocity that are set as `fixedValue`. Meanwhile, pressure is `totalPressure`, which is set to be a reference pressure p_0 when acting as an outlet, or $p_0 - \frac{1}{2}|\mathbf{u}|^2$ when acting as an inlet. This combination of `pressureInletOutletVelocity` and `totalPressure` is good for patches that might become inlets of an unknown velocity as it allows the velocity to settle to its own value. It encourages stability because as the velocity increases, the pressure decreases, and thus the pressure gradient decreases (Greenshields, 2020; Greenshields and Weller, 2022). The phase fraction is `inletOutlet` so that water can leave the domain but not re-enter it. Again, the transported turbulence parameters are `inletOutlet` while the turbulent viscosity is `calculated`.

Walls

At the walls, velocity is set to `noSlip`, alpha `zeroGradient`, and pressure `fixedFluxPressure`. This pressure boundary condition means that the flux at the boundary is that specified by the velocity boundary condition and it is not changed by the pressure gradient. Meanwhile, the turbulence parameters have wall functions, which are described in Section

4.1.6.

Symmetry

Finally, a boundary patch may be designated as a symmetry patch, and then all its boundary conditions must be specified as `symmetry`. This can be used to cut down simulation time when the mesh geometry is symmetric across the patch and the flow is assumed to be symmetric across that patch (Greenshields and Weller, 2022). The `symmetry` boundary condition is essentially the same as a slip condition; the normal velocity is zero but the tangential velocity is retained. All the scalar quantities are `zeroGradient`. Note that this boundary condition is not used in `weirOverflow` since it is a 2D case, but it is used later this chapter to simplify the 3D simulations.

4.1.6 Turbulence modelling

The turbulence model is defined and implemented separately from the solver, allowing further customisation of the numerical setup. There are a range of RAS and LES turbulence models available in OpenFOAM, including the k-ϵ, k-ω, and k-ω SST models outlined in Section 3.2. Corresponding wall functions for the turbulence parameters, set out in Section 3.2.5, are also available as derived boundary conditions. The wall function for k is called `kqRWallFunction`, ϵ `epsilonWallFunction`, and ω `omegaWallFunction`. Recall that either y_+ or y_* are needed to calculate the turbulent viscosity ν_t, called `nut` in OpenFOAM. The value $y_+ = \left(\rho y \sqrt{\tau_w/\rho}\right)/\mu$ is a function of $\mathbf{u}$ while the value $y_* = \left(\rho y \sqrt{C_\mu^{1/2} k}\right)/\mu$ is a function of k. Thus `nutUWallFunction` is a boundary condition for ν_t that uses y_+, while `nutkWallFunction` is a boundary condition for ν_t that uses y_*. Going further, `nutUBlendedWallFunction` uses the automatic wall treatment of Menter et al. (2003) to determine if the cell centres are in the viscous sublayer or log-law region, and apply the correct boundary condition as necessary. Finally, the `yPlus` post-processing utility can be used to calculate the y_+ value across all the boundary patches, confirming if the boundary cells are a suitable size.

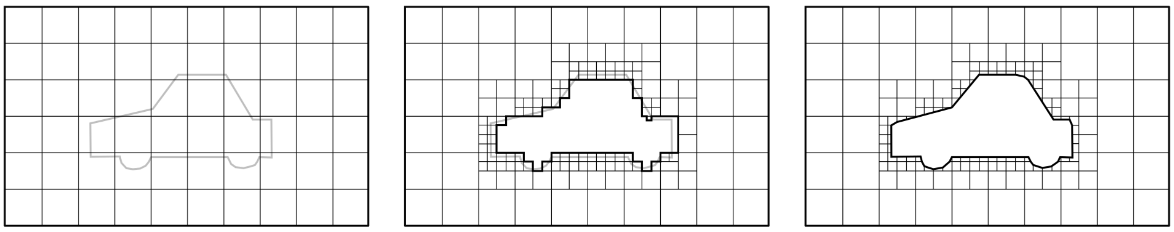

Figure 4.3: Stages of mesh generation with `snappyHexMesh`. From left to right: background mesh, octree-based castellation, and snapping, based on OpenCFD (2019b, pp.50–54).

4.1.7 Mesh generation

OpenFOAM has two main mesh generation tools: `blockMesh` and `snappyHexMesh`. The utility `blockMesh` generates a structured mesh, while `snappyHexMesh` follows the steps illustrated in Figure 4.3 to cut out stl geometry files from the mesh, refining the boundary cells. While this method results in meshes made primarily of hexes, which is good for solution accuracy, the refinement process results in some non-orthogonality. The levels of non-orthogonality and skewness can be found using the utility `checkMesh`, which will then inform the best choice of numerical scheme. Meshes can also be imported from other CFD software, for example, the utility `fluent3DMeshToFoam` converts meshes generated with commercial software from Ansys. Finally, mesh refinement and manipulation can result in a numbering of the cells that leads to a high bandwidth matrix that is more computationally expensive to invert. Consequently, the utility `renumberMesh` should be used to reduce the bandwidth of the matrix.

4.1.8 Parallelisation

Fully 3D simulations are computationally expensive. Generally, they are parallelised across multiple processors so that they can be completed more quickly. This is straightforward for explicit simulations, but implicit simulations can require the exchange of information across processors every iteration of the matrix solver (Jamshed, 2015). OpenFOAM uses implicit methods, so it uses the parallelisation method called domain decomposition. The domain is decomposed into a disjoint union of sub-domains, with each one allocated to a different processor. OpenFOAM offers different options for domain decomposition, including scotch decomposition, which is an optimisation method to minimise the number of processor boundaries (OpenCFD, 2019b).

After the domain has been decomposed, the processors communicate with each other using the open-source message passing interface OpenMPI (OpenCFD, 2019b). This is done through the wrapper library `Pstream`, which allows other types of communication to be implemented without changing the CFD code itself. OpenFOAM does not use the "halo layer" approach, which duplicates cells adjacent to a processor boundary. Instead, OpenFOAM uses the "zero halo layer" approach with out-of-core addressing (Jasak, 2012).

Practically, 10,000 computational cells per core works well for OpenFOAM, but it depends on the hardware and case. Dedicating more cores to the case risks a loss of efficiency due to the extra messages that must be passed between processors (Jamshed, 2015). Moreover, the `fileHandler collated` option should be used for large cases so that only one file is output instead of one for each processor, and it should be noted that MPI is more suitable for HPC clusters than servers.

4.1.9 Weaknesses

OpenFOAM is not built for modelling open-channel flow. The biggest practical problem is that there is no boundary condition suitable for the downstream outlet (Flora, 2012; Bayon-Barrachina and Lopez-Jimenez, 2015; Olsen, 2015). A further practical problem is that there is no utility in the standard installation to extract the depth-averaged velocity, required to compare VOF results with hydraulic modelling results. However, as the source code is open, this extra functionality can be added relatively easy.

However, there are more fundamental problems in OpenFOAM's solvers. As established in Section 4.1.4, OpenFOAM's VOF solver `interFoam` is diffusive, while its newer solver `interIsoFoam` has not been extensively tested. Moreover, OpenFOAM's reliance on PIMPLE means that the VOF solvers get none of the benefits of implicit schemes but all of the drawbacks. First, there are the explicit components in the PIMPLE algorithm that mean that it lacks unconditional stability. Second, there is the explicit nature of isoAdvector and explicit MULES. These two points mean that the time step must be small, with a Courant smaller than 1—making the choice of an implicit scheme redundant. Furthermore, while explicit schemes are usually most accurate around CFL = 1, implicit schemes need much smaller time steps to obtain comparable accuracy. Otherwise, there are phase and dampening errors (Hirt, 2014). We will address this numerical problem by developing a new solver in Chapters 5–6, but the practical problems regarding the outlet boundary condition and extracting the depth-averaged velocity will be addressed now. This will be paired with the existing numerics to simulate four leaky barrier flume experiments in 3D.

4.2 New OpenFOAM functionality

4.2.1 Depth outlet boundary condition

In many hydraulic model applications for subcritical flow, the discharge $q = hu$ is set at the inlet and the depth h at the outlet. This allows backwater effects from any obstacles beyond the computational outlet (for example, from tides or a bridge blockage downstream) to influence the solution within the domain. In OpenFOAM, while there is an inlet boundary condition that allows the discharge to be set (`variableHeightFlowRateInletVelocity`), there is currently no outlet boundary condition that allows the depth to be set. Besides the obvious problem that this makes it more difficult to compare the entire domain with results from a shallow water flow model, experience has shown that this can also result in a strange feedback loop where the entire domain fills up with progressively slower water. It is possible to have a weir near the downstream outlet to give the desired depth (Mukha et al., 2020), but this indirect approach can be difficult to control. A more direct option is to split up the outlet patch so that water can only exit from the intended depth, but this is neither straightforward nor reliable (Olsen, 2015). Flora (2012) suggested a workaround by applying an average velocity across the whole outlet patch, and Bayon-Barrachina and Lopez-Jimenez (2015) took a similar approach, but a programmed solution would be more convenient and generalisable to transient simulations. The boundary condition developed here essentially automates this method, allowing it to be used for transient simulations where the flow rate changes between time steps.

While the `variableHeightFlowRateInletVelocity` boundary condition reads in Q from a dictionary, calculates A by reading in the current α field, and uses the equation $Q = Au$ to calculate u, the idea here is the opposite: calculate Q by reading in the current u field, read in h from a dictionary (giving A), and then use the equation $Q = Au$ to calculate a new value of u. Note that the value of A is restricted to the resolution of the grid, which may result in slight under- or over-estimates of the velocity u. A possible solution to this problem would be to consider cells partially full of water; however, this would require excessively complex geometrical operations.

If the velocity is applied across the whole outlet patch, the increased velocity of the air means that air has to be drawn in from the atmosphere patch, which can result in high velocities and instability. To counter this, the velocity is applied only to cells beneath the depth marker and a small buffer above it. The buffer is there so that small fluctuations in water level do not get stuck inside the domain, unable to leave.

Another problem is that the sudden slowing down at the boundary condition can result in eddies that temporarily provide a negative Q at the boundary. This results in a negative u, and a feedback loop causes the simulation to blow up. Thus the boundary condition defaults to a fixed value of zero velocity if the total flow rate out of the domain is negative, allowing such problems to settle out. As this settling out takes time, the boundary condition should be used with caution in transient simulations.

4.2.2 Depth-averaged velocity function object

Another problem lies in post-processing. It is useful to be able to extract the values of the depth h and depth-averaged velocity u from the VOF results as these are the quantities of interest in hydraulic models. While the depth is simple to extract from OpenFOAM simulations with either a ParaView filter or standard surface function object, the depth-averaged velocity is more complicated. This is because the approaches of hydraulic models and VOF are fundamentally different. Gschaider (2014, 2011) suggested a workaround with a community contribution to OpenFOAM called `swak4Foam` but, again, a programmed solution would be more convenient.

Case	Flume		Barrier		Mesh			Boundaries	
	Dimensions (m)	Slope (-)	Gap (m)	Size (m)	Dimensions (m)	Cells (mil.)	Bad cells	Q_{in} (m^3/s)	h_{out} (m)
N33								0.0096	0.08
N53	$0.3 \times 17.8 \times 0.3$	1:1600	0.025	0.1	$0.15 \times 7.8 \times 0.30$	0.87	9	0.0060	0.06
N43								0.0032	0.05
W	$1.0 \times 12.0 \times 0.6$	1:50	0.035	0.2	$0.50 \times 3.0 \times 0.50$	1.46	43	0.0318	0.04

Table 4.4: Mesh and boundary conditions for each case. Note that the upstream boundary condition is half the measured value due to the symmetry plane, and the downstream boundary condition has a low level of precision due to the dependence of the custom boundary condition on the mesh resolution.

The depth-averaged x-direction velocity at a point x_0 is given by

$$u(x_0) = \frac{1}{|\{x_o\} \cap W|} \iint_{\{x_o\} \cap W} u(x_0, y, z)\, dy\, dz. \tag{4.19}$$

where u is the x-direction velocity and W is the volume filled with water. We call this the depth-averaged velocity because width averaging is trivial in a standard channel; the width is simply the width of the domain, whereas the depth is of the water and not of the entire domain. To approximate (4.19) at the discrete level, one practical solution is to partition the x-axes into subintervals and use

$$u(x_0) \approx \frac{\sum_i \left(u(\Omega_i) \cdot |\Omega_i|\right)}{\sum_i |\Omega_i|}, \tag{4.20}$$

where the computational cells Ω_i are those whose x-components of the cell centres lie in the subinterval around x_0 and whose α values are greater than 0.5.

4.3 Application to leaky barriers

This section simulates four steady-state flume experiments using OpenFOAM, comparing the results with data outlined in Appendix A. Table 4.4 summarises the cases, three for the narrow flume (N33, N53, N43) and one for the wide flume (W). It might seem that the width dimension of the flume could be ignored, and that a 2D simulation (one dimension along the length of the flume, and one dimension along the height) would suffice instead of

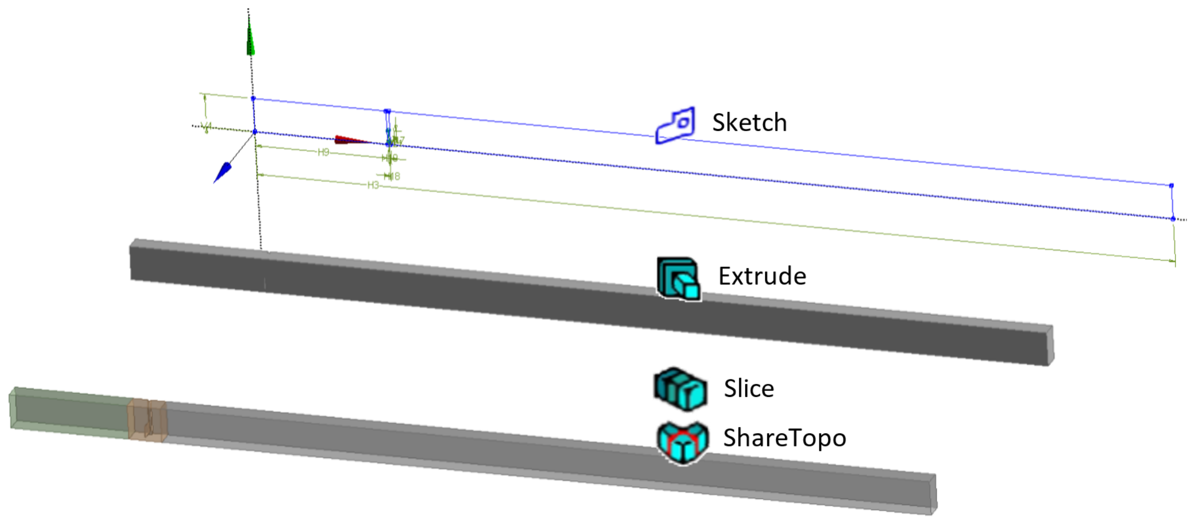

Figure 4.4: Creation of narrow flume geometry in Ansys DesignModeler.

a more expensive 3D simulation. However, initial explorations showed that a 2D approach does not allow the nappe over the barrier to be aerated, which it should be for the narrow flume cases (N33, N53, N43). Instead, any air initially aerating the nappe gets entrained and transported downstream. As the simulation is 2D, there is no mechanism for the entrained air to be replaced, and so the nappe clings to the barrier. This problem was also experienced by Patil (2018) in simulating sea defences with OpenFOAM, but there is no reason that it would be an OpenFOAM-specific problem unless special boundary conditions exist in commercial software to deal with it. In any case, we perform a 3D simulation to avoid this problem.

4.3.1 Methodology

Recreating the flume

As both flumes are long, a truncated domain was modelled. A symmetry plane down the middle of the flume decreased computation time further. A mesh for each flume was created with Ansys Workbench 18.1. First, geometry was created with DesignModeler (Figure 4.4), which was then meshed with Meshing (Figure 4.5). Ansys tools were used instead of OpenFOAM tools due to their increased flexibility and levels of specification. In particular, hybrid meshes can be created with Ansys. This meant the majority of the domain

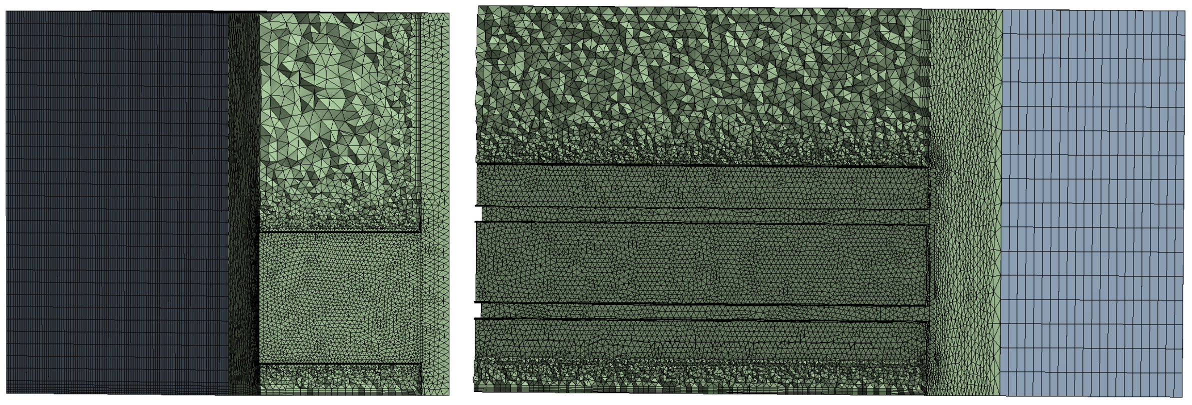

Figure 4.5: Cross sections of the meshes at the location of the leaky barrier. The green zone is the tetrahedal mesh around the leaky barrier, while the blue zone is the hexahedral mesh upstream from the barrier. Left: narrow flume, right: wide flume.

could be filled with computationally efficient and higher quality hexahedral cells, while tetrahedral cells could easily wrap around the leaky barriers. The boundaries between the two zones were conformal, allowing individual cell faces to match up across the boundaries.

Recall that wall functions require boundary cells to be of a specific non-dimensional size. This can be achieved in Ansys by using inflation layers, where the cell size increases smoothly from the boundary cells to the non-boundary cells. However, inflation layers are not available for manually created hexahedral meshes, and so the hexahedral zone was generated using an automatic multizone method, which can be used with inflation. This resulted in slightly skewed hexahedral cells, but allowed inflation layers to line up smoothly in both the hexahedral and tetrahedral zones. Table 4.4 shows that only very small numbers of cells failed the mesh quality check. This is acceptable because it is impossible to create a perfect mesh.

Modelling the free-surface flow

Recall that only the fixed-lid approach has been used for 3D modelling of leaky barriers in the literature. This is simple and relatively cheap, but also a major limitation as the free surface is not an output of the model. OpenFOAM's VOF solver `interFoam` allows the free surface to move throughout the simulation. As discussed in Section 4.1.5, such free-surface flow requires specialist boundary conditions. In particular, these cases replicated the boundary conditions from the standard `weirOverflow` tutorial case, which allows a flow rate to be specified at the inlet and air to enter and exit freely through the upper boundary, but with the custom downstream boundary condition developed in Section 4.2.1 allowing a fixed water depth to be specified at the outlet and a `symmetry` boundary condition specified along the symmetry plane. Turbulence parameters were calculated from the standard relations with an intensity of 5% and a length scale corresponding to the size of the barrier, and a wall function based on y_+ and valid in both the viscous sublayer and log-law region was used for the walls and leaky barrier. While this kind of VOF simulation is more computationally expensive than a fixed-lid simulation, OpenFOAM simulations can be parallelised. The domain was decomposed into 16 subdomains, with each subdomain sent to its own core on a blade server (Intel Xeon E5-2680 v4 2.4 GHz).

Starting stable, ending accurate

As the cases under consideration are steady state, the simulations started off with local time-stepping and first-order spatial discretisation. When the simulations converged to a steady state, the spatial discretisation accuracy was increased to second-order and the time discretisation made no longer local. Interestingly, local time-stepping did not work with Case W, but this was less of a concern as the thicker barrier meant larger cell sizes, and thus global time steps, could be used. The settings for the second, accurate stage of modelling are shown in Table 4.5 (case files in Appendix B).

4.3.2 Results and discussion

Comparison with experimental data

The simulated free surface was extracted using the contour of $\alpha = 0.5$. Figure 4.6 shows that the simulated free surface generally compares well with observations. The location of the hydraulic jump is not captured so well, in fact, it is very sensitive to small changes in parameters. This could be caused by the lack of precision in the downstream boundary condition, or simply the difficulty of measuring the discharge in the narrow flume. Note

Category	Setting	Value
Solver	application	interFoam
	VOF	semi-implicit
	alpha correctors	3
	inner pressure-velocity loops	3
	turbulence model	k-ω SST
Time integration	scheme	Euler-implicit
	max Courant number	0.8–0.9
	max alpha Courant number	0.8–0.9
Divergence	momentum	second-order TVD
	turbulence	second-order TVD
	VOF	second-order TVD
	VOF compression	linear interpolation
	viscosity	linear interpolation
Other schemes	gradient	Gauss with cell limiting
	Laplacian	non-orthogonal correction
	surface-normal gradient	non-orthogonal correction
p_rgh	matrix solver	PCG, DIC preconditioner
	tolerance	1.00e-07
	relative tolerance	0.05 (0.0 in final PISO loop)
U, k, omega	matrix solver	smooth, symmetric Gauss-Seidel
	tolerance	1.00e-08
	relative tolerance	0.01

Table 4.5: Numerical settings for OpenFOAM.

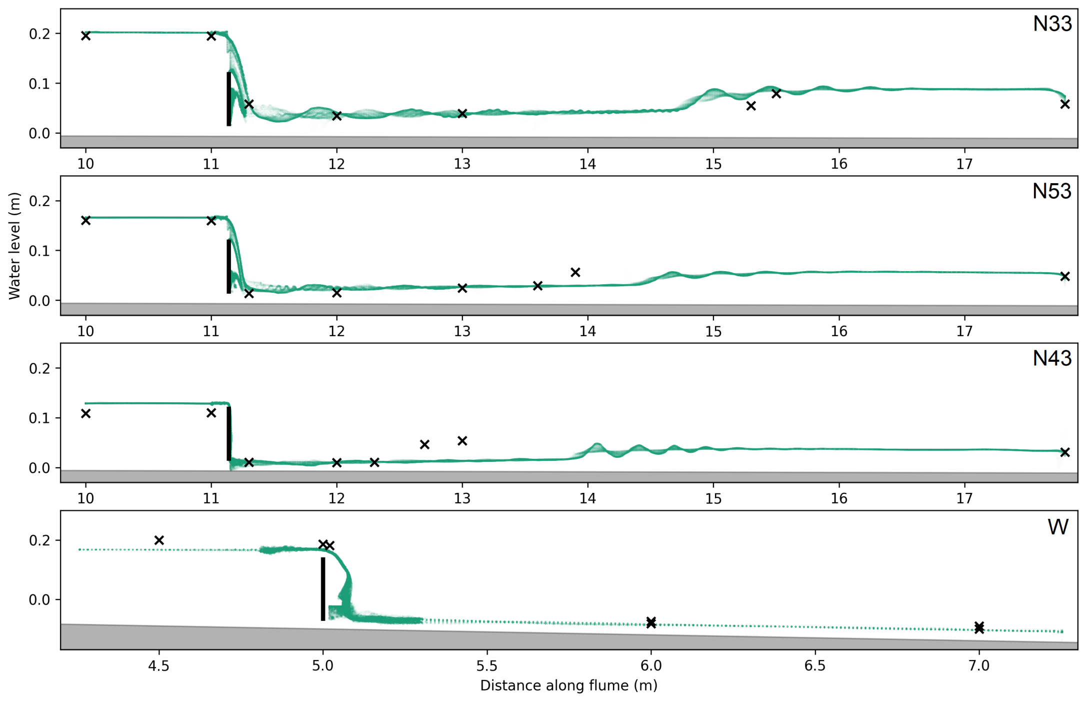

Figure 4.6: Observed (black crosses) and simulated (green lines) free surfaces for each case.

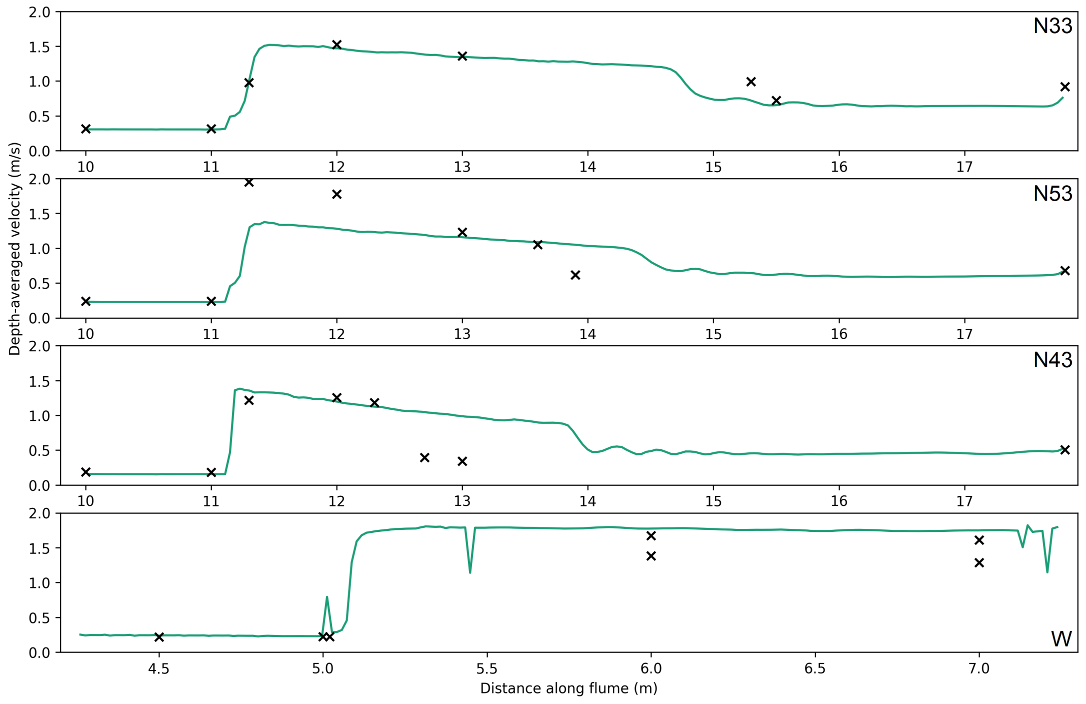

Figure 4.7: Observed (black crosses) and simulated (green lines) depth-averaged x-direction velocities for each case.

that the oscillations around the hydraulic jump are not numerical oscillations but were present in the flume experiments.

The simulated depth-averaged velocity was calculated using the custom function object from Section 4.2.2. The observed depth-averaged velocity was found using the steady-state continuity equation, rather than direct velocity measurements. Figure 4.7 shows that this depth-averaged velocity is generally recreated well by the simulations but has some discrepancies, which could be because of sensitivity. For instance, Figure 4.6 shows that two depth measurements were taken at each point downstream of the barrier in Case W. These were approximate minimum and maximum depths because the flow was very turbulent and there was no single correct depth. However, Figure 4.7 shows that the minimum and maximum depth measurements result in very different minimum and maximum depth-averaged velocities. Meanwhile, the spikes in the simulated depth-averaged velocity for Case W are probably due to the effect of the barrier slope on the sampling procedure.

This acceptable performance compared to measurements suggests that, with caution, the results can be used to gain insights on the steady-state flow around leaky barriers.

Visualising flow around the barrier

Figures 4.8 and 4.9 show the flow patterns around the leaky barrier, visualised using ParaView and clipped at $\alpha = 0.5$ to show only the water. This visualisation method hides numerical diffusion, but it is useful nonetheless, and there are three different features to draw out from it. First, note that the nappe is aerated in Cases N33 and N53 but not W, which can be explained by the presence of the leaky barrier support in the narrow flume. Such behaviour was observed in the flume experiments; see Figure A.4 for the lack of nappe in the wide flume. Second, the visualisation clearly shows the stagnation point upstream of the barrier, which is not so clear in flume experiments. This could be found systematically to develop discharge equations for combined weir/gate flows, which is what leaky barriers are sometimes represented with in the literature (Cabaneros et al., 2018; Metcalfe et al., 2017) and what they were represented as in Chapter 2 and Leakey et al. (2020), demonstrating the added value of using CFD alongside flume experiments. Third, and perhaps most significantly, note the hydraulic jump in Cases N33 and N53. As was

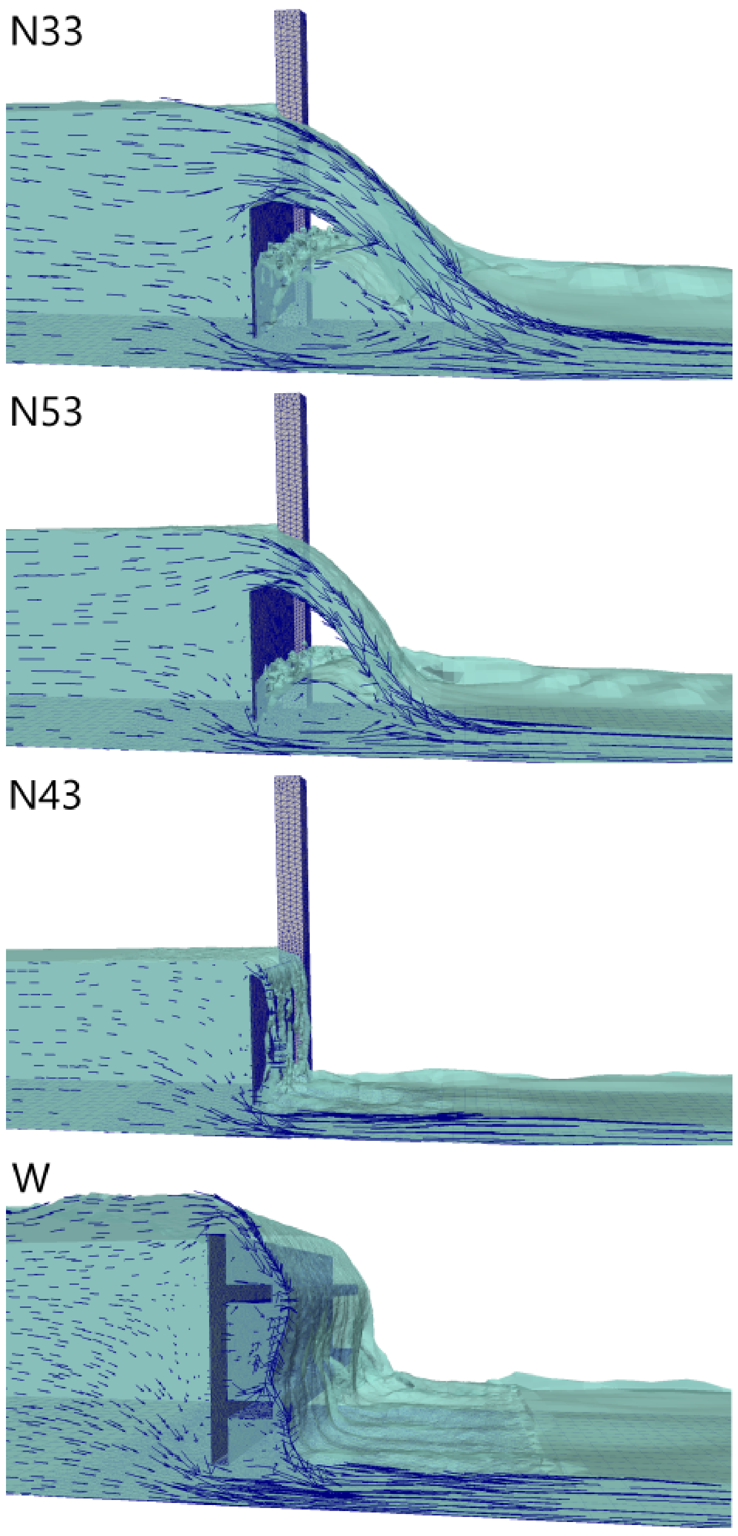

Figure 4.8: Velocities around the leaky barrier on the symmetry plane for each case, uniform distribution of glyphs rather than every cell.

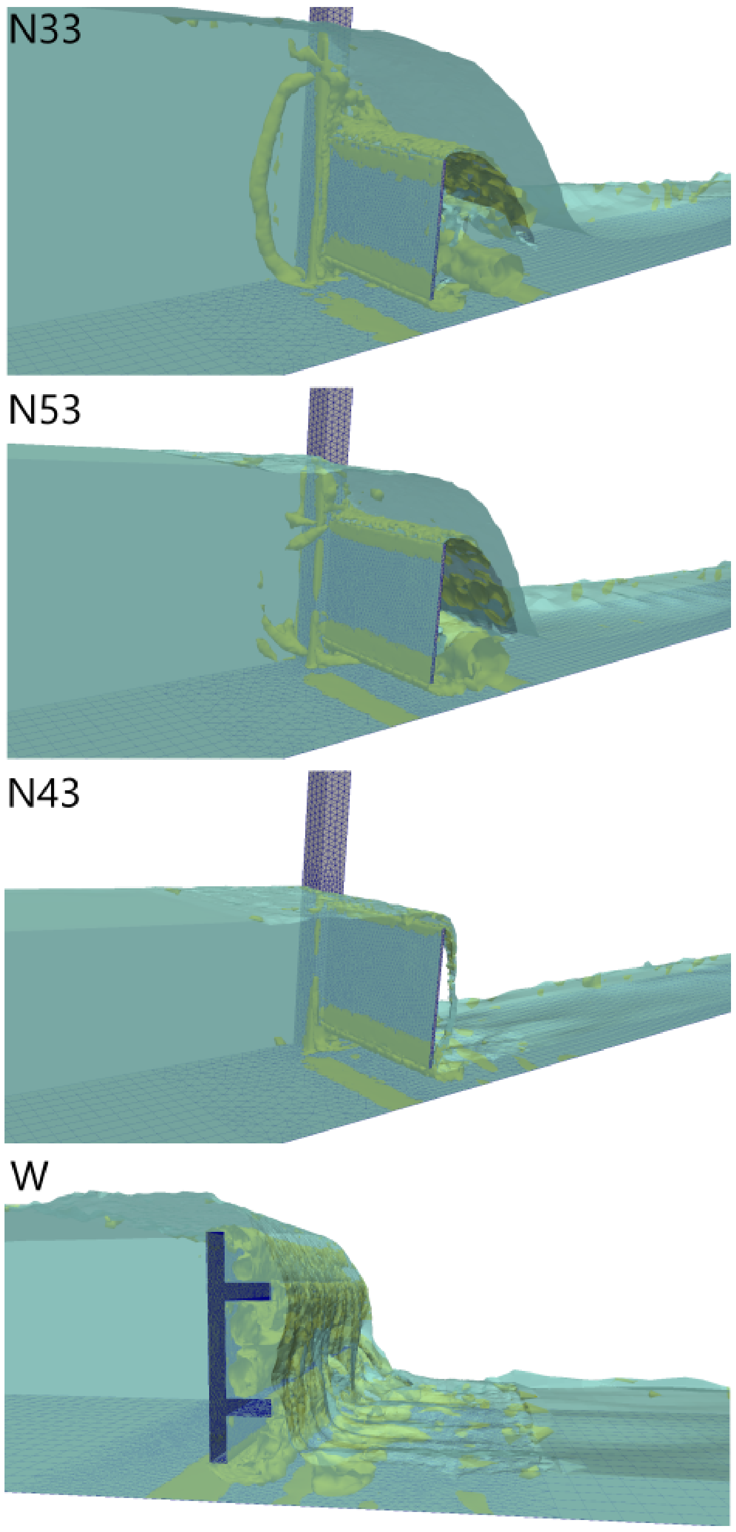

Figure 4.9: Q-criterion contours ($Q = 100$) around the leaky barrier for each case, showing the non-turbulent eddies as it is a Reynolds averaged simulation.

Case	Max y_+ (-)	Max bed shear stress (m^2/s^2)	Total force on barrier (N)
N33	1.073e+02	(2.154e-02, 2.819e-03, 3.512e-02)	(1.411e+01, 2.836e-02, 7.251e-04)
N53	7.544e+01	(1.126e-02, 1.498e-03, 1.794e-02)	(1.119e+01, -4.682e-02, 4.024e-05)
N43	9.452e+01	(1.917e-03, 6.548e-04, 8.902e-03)	(7.118e+00, -7.345e-02, -1.225e-03)
W	1.505e+02	(1.973e-02, 1.987e-03, 1.102e-02)	(1.273e+02, -2.896e+01, -1.776e-02)

Table 4.6: Post-processing results for selected boundary patches.

also observed in the flume experiments, the flow over the barrier submerges the flow under the barrier. This has a practical application for representing leaky barriers as a combined weir/gate structure. Specifically, the hydraulic jump means that the free gate equation is not always valid even when there is supercritical flow downstream of the barrier, and so the submerged equation should sometimes be used instead. Further research would need to take place to find out at what point the submerged equation should be used.

Flow adjacent to walls

The flow in the boundary cells was analysed using standard OpenFOAM function objects. Table 4.6 shows that the maximum y_+ values were all comfortably in range, and so the wall functions were valid. Additionally, the maximum bed shear stress and the total force on the barrier increase with the flow rate. This sort of analysis could be useful for predicting scour and failure potential in a systematic study of leaky barrier designs, and again demonstrates the added value of using CFD alongside flume experiments.

4.4 Conclusion

While leaky barriers are a popular method for reducing flood risk, it is difficult to quantify their impact in the field. Until now, leaky barriers have only been modelled by changing parameters in hydrological and hydraulic models, by analogy to well-understood simple hydraulic structures or—much more rarely—by fixed-lid 3D modelling. All these approaches have inherent limitations. Notably, the fixed-lid approach cannot capture any backwater effect.

This chapter explained how OpenFOAM and its VOF solvers work, developed new practical functionality for modelling open-channel flow in OpenFOAM, and used this to model 3D non-fixed-lid-flow around leaky barriers for the first time. The free surface and depth-averaged velocity extracted from the simulations compared well with experimental flume data. Moreover, flow features observed in the flume experiments, like the submergence of the gate caused by the weir nappe, were captured in the simulations. Consequently, we can have confidence that `interFoam` provides acceptable results in such steady-state cases.

This kind of combined physical and mathematical modelling could be applied systematically to model different leaky barrier designs and extract steady-state discharge equations with confidence. This could then provide a much-needed upgrade to the capabilities of current hydrological and hydraulic models to handle different designs of barrier. Moreover, the ability to calculate bed shear stress and the force on the barrier could inform a more holistic barrier design process that takes into account the potential for scour or even failure.

However, the applicability of this method to transient simulations is not a given due to the semi-implicit nature of the pressure-velocity coupling in OpenFOAM. Moreover, being semi-implicit, OpenFOAM simulations require expensive approximate matrix inversion, which is difficult to parallelise. Therefore, Chapter 5 develops an explicit free-surface solver that builds upon some of the recent advances in CFD, and Chapter 6 implements it in OpenFOAM.

Chapter 5. Developing a new free-surface solver

As discussed in Chapter 4, the semi-implicit nature of OpenFOAM's volume of fluid solvers mean that they have all of the disadvantages and none of the advantages of an implicit solver. New numerical and computational methods have been developed since FOAM was originally created by Weller et al. (1998), so it is worth using these to make a solver more suitable for free-surface flow and efficient parallelisation. However, the OpenFOAM framework itself remains a useful test bed for new academic code as not everything has to be built from scratch. In the present chapter, we develop a new Computational Fluid Dynamics (CFD) solver for free-surface flow based on the method of artificial compressibility. The scheme is general, theoretically suitable for viscous flows on 3D unstructured meshes. We implement it for inviscid flows on 2D Cartesian meshes, and demonstrate its effectiveness against four sets of benchmark tests. This work in this chapter has been published in Leakey et al. (2022b). In Chapter 6, we go on to transfer the solver into OpenFOAM code for use with arbitrary 3D unstructured meshes.

5.1 Introduction to the solver

The Navier-Stokes equations do not include an equation for pressure. For incompressible flows, numerical methods must find a way around this problem to update the pressure field. As discussed in Sections 3.3.6 and 4.1.3, the method taken in OpenFOAM is the PIMPLE algorithm, a predictor-corrector method based on matrix inversion. An alternative approach is artificial compressibility, which adds a pseudo-time derivative of pressure to the incompressibility constraint. This makes the governing equations hyperbolic, allowing the solution to advance in pseudo-time until it converges, at which point the pseudo-time derivative disappears and the original equations are satisfied (Drikakis and Rider, 2005; Toro, 2009; Kwak and Kiris, 2011). As the equations are hyperbolic, they can be solved with methods designed for compressible flows, such as Godunov-type schemes. Chorin (1997) created artificial compressibility for steady-state cases, but it has since been generalised to transient cases using dual time stepping, that is, driving the solution to the incompressible limit at each time step.

Artificial compressibility has a key computational advantage compared to the SIMPLE, PISO, and PIMPLE algorithms. The artificial compressibility method is easier to parallelise, especially if the pseudo-time stepping is explicit (Nourgaliev et al., 2004; Loppi, 2019; Vermeire et al., 2019). Hodges (2020, p.5) notes that this is because artificial compressibility "replaces the global solution of an elliptic matrix inversion with local hyperbolic wave propagation," making it simpler, and therefore more efficient, for processors to communicate with each other. Consequently, the artificial compressibility method has much potential for application on modern hardware architectures (Loppi, 2019; Vermeire et al., 2019).

Despite this potential for parallelisation, the artificial compressibility method has not been thoroughly developed for free-surface flows (Shapiro and Drikakis, 2005a). In the artificial compressibility framework, researchers have modelled free-surface flows by coupling the artificial compressibility equations with the VOF equation (Pan and Chang, 2000; Bhat and Mandal, 2019a,b; Ntouras and Papadakis, 2020) and level-set equation (Nourgaliev et al., 2004). However, the most developed method is to couple the equations with a transport equation for density, that is, to solve the variable-density artificial-compressibility equations, and to mark the location of the free surface by the density discontinuity. Kelecy and Pletcher (1997) pioneered this method in a Godunov-type scheme, Qian et al. (2006) added the Cartesian cut-cell technique to simulate moving bodies, Shin et al. (2012) added the hybrid Cartesian/immersed boundary technique to simulate moving bodies, and Bassi et al. (2018) used the method as part of a discontinuous Galerkin scheme, also deriving shock and rarefaction relations for the equations.

Even though the variable density approach is the most developed free-surface method in the artificial compressibility framework, these key papers were spaced out over several decades. Further work developed upon these specific numerical schemes, rather than investigating alternatives. The scheme developed by Qian et al. (2006) has been tested further (Ma et al., 2011; Wang and Wang, 2009, 2010a,b, 2012; Wu et al., 2014; Wang et al., 2020), used to model wave overtopping (Gao et al., 2007, 2009; Ingram et al., 2009), and extended to 3D (Hu et al., 2013). The scheme developed by Shin et al. (2012) has also been tested further (Shin, 2013), and discontinuous Galerkin schemes using the exact Riemann solver put forward by Bassi et al. (2018) have been tested as well (Manzanero et al., 2020; Massa et al., 2020). This is good, but more is needed. Before the method can be used practically with confidence, it needs to be systematically explored, with advances from the wider CFD literature incorporated, and some specific problems solved.

In this chapter, we develop a Godunov-type scheme for the variable-density artificial-compressibility equations, as done by Kelecy and Pletcher (1997) and Qian et al. (2006). The scheme is general, theoretically suitable for viscous flows on 3D unstructured meshes. Here we implement it for inviscid flows on 2D Cartesian meshes, and demonstrate its effectiveness against four sets of benchmark tests. Before setting out the specific objectives of this chapter, recall from Section 2.1 that the general idea of a Godunov-type scheme is to calculate the numerical fluxes at cell interfaces with Riemann solvers, which derive or approximate the wave pattern propagating from a single step discontinuity. This has the benefit of capturing shocks automatically, and so lends itself to a range of applications, from modelling flood waves with the shallow water equations (as in Chapter 2) to modelling blasts with the Euler equations (Toro, 2009, 2001). One of the most common methods to make Godunov-type schemes second-order accurate in space is MUSCL reconstruction (van Leer, 1976) as is used in Chapter 2, and higher-order accuracy can be obtained with the Weighted Essentially Non-Oscillatory (WENO) method (Liu et al., 1994). While Godunov-type schemes can give stable results for under-resolved flows, compensating for poor mesh resolution with a higher-order method can lead to spurious vortices (Brown, 1995; Drikakis and Smolarkiewicz, 2001). The focus of the present chapter is three-fold: robust Riemann solvers, pressure gradients for MUSCL reconstruction, and ease of parallelisation.

Riemann solvers

In the variable-density artificial-compressibility equations, the density discontinuity is marked by a contact wave. Consequently, the free surface can be captured by the contact wave in a Riemann solver, without resorting to surface-capturing or surface-tracking methods such as VOF. This means that the robustness of the Riemann solver is paramount. In this study, we apply for the first time the Osher Riemann solver of Dumbser and Toro (2011) to the variable-density artificial-compressibility equations. The results are compared with the Riemann solver of Bassi et al. (2018), exact under certain wave configurations, and the Roe Riemann solver implemented by Kelecy and Pletcher (1997) and Qian et al. (2006).

Pressure gradients

Without special treatment, standard MUSCL reconstruction will not account for the impact of a density discontinuity on the pressure gradient. This can result in an imbalance of forces at the free surface and, in turn, either parasitic velocities (DeGroot and Straatman, 2011; Efremov et al., 2017; Hashemi et al., 2020) or erroneous compression at what should be the incompressible limit. As will be shown later, the compression can resemble Gibbs-type oscillations, but it is a distinct problem of its own that persists even with the most severe of limiters. For the variable-density artificial-compressibility equations, only Qian et al. (2006) have addressed this problem and only in the hydrostatic case; they did not consider how there can still be a force imbalance in the absence of gravity. However, it has been investigated more comprehensively in the wider CFD literature, both for hydrostatic pressure (Zingale et al., 2002; Botta et al., 2004; Krause, 2019; Fuchs et al., 2010) and non-hydrostatic pressure (Ntouras and Papadakis, 2020; Queutey and Visonneau, 2007; Kruisbrink et al., 2018; Vukčević et al., 2018). In this study, we introduce a new method for calculating the pressure gradient that has not been considered before either in the artificial compressibility literature or the more general CFD literature. Based on a rearrangement of the momentum equation and exploitation of the other TVD gradients and source terms, it captures the pressure gradient discontinuity at the free surface automatically.

Ease of parallelisation

Kelecy and Pletcher (1997), Qian et al. (2006), and Bassi et al. (2018) all used implicit schemes. However, this fails to take advantage of one of the key advantages of artificial compressibility. If the pseudo-time stepping is explicit, then it is relatively easy to parallelise on modern hardware architectures (Nourgaliev et al., 2004; Loppi, 2019; Vermeire et al., 2019). In this study, we use the explicit Runge-Kutta scheme of Vermeire et al. (2019) to advance in pseudo-time.

5.2 Governing equations

5.2.1 Artificial compressibility

Incompressible flows with variable density can be modelled using the system of equations (3.1)–(3.3). The problem with this system of equations is that there is no equation for pressure. In the method of artificial compressibility, pseudo-time derivatives are added to equations (3.1)–(3.3) to get

$$\frac{\partial \rho}{\partial \tau} + \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{5.1}$$

$$\frac{\partial}{\partial \tau}(\rho \mathbf{u}) + \frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \left(\mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right) + \rho \mathbf{g} \tag{5.2}$$

$$\frac{1}{\beta}\frac{\partial p}{\partial \tau} + \nabla \cdot \mathbf{u} = 0 \tag{5.3}$$

where τ is pseudo-time and β is the artificial compressibility coefficient. This is solved in pseudo-time until convergence, and thus the pseudo-time derivatives go to zero, retrieving the original equations again. Again, the term t is needed for the solution to be transient, while τ is needed to make the equations hyperbolic; these will form two nested loops in a numerical scheme (dual time stepping).

If the solution converges in pseudo-time, then $\partial p/\partial \tau \approx 0$ and the artificial compressibility coefficient β should not impact the solution. However, when the maximum density is around $\rho = 10^3$ as for water, setting $\beta < 10^3$ can result in instability and setting $\beta > 10^4$ can inhibit convergence, meaning $\partial p/\partial \tau \approx 0$ is not achieved (Kelecy and Pletcher, 1997). Therefore some trial and error is required to pick a suitable value of β. In practice, we have found that setting $\beta = 1100$ is just large enough to avoid instability, and so will have minimal impact on convergence.

There have not been many theoretical investigations of the governing equations (5.1)–(5.3). Shapiro and Drikakis (2005a) derived three different formulations of characteristics-based schemes and compared these at the constant density limit, while Bassi et al. (2018) derived properties of the rarefaction and shock waves from the Riemann invariants and Rankine-Hugoniot jump conditions respectively. The focus of the present paper is developing and testing a numerical scheme rather than deriving further theoretical properties, but the developments by Bassi et al. (2018) are discussed further in Section 5.5.1.

5.2.2 Compact form

The governing equations (5.1)–(5.3) can be written in the compact form

$$\frac{\partial \mathbf{Q}}{\partial \tau} + \mathbf{I}_C \frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F} = \mathbf{B} \tag{5.4}$$

where the conservative state vector is

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ p/\beta \end{bmatrix}, \tag{5.5}$$

the cancellation matrix (Vermeire et al., 2019) is

$$\mathbf{I}_C = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix}, \tag{5.6}$$

the flux vectors are

$$\mathbf{F} = \mathbf{F}_{inv} - \mathbf{F}_{vis} \tag{5.7}$$

where the inviscid flux is

$$\mathbf{F}_{inv} = \left(\begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u \end{bmatrix}, \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v \end{bmatrix}, \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w \end{bmatrix} \right) \tag{5.8}$$

and the viscous flux is

$$\mathbf{F}_{vis} = \left(\begin{bmatrix} 0 \\ 2\mu \frac{\partial u}{\partial x} \\ \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ 2\mu \frac{\partial v}{\partial y} \\ \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ 2\mu \frac{\partial w}{\partial z} \\ 0 \end{bmatrix} \right), \tag{5.9}$$

and the body force source term is

$$\mathbf{B} = \begin{bmatrix} 0 \\ \rho \mathbf{g}_x \\ \rho \mathbf{g}_y \\ \rho \mathbf{g}_z \\ 0 \end{bmatrix}. \tag{5.10}$$

5.2.3 Eigenstructure

The eigenstructure of the governing equations, that is, the eigenvalues and the right and left eigenvectors of the Jacobian, is important for developing a Godunov-type scheme. The following calculations were completed and checked by hand with the help of Wolfram Alpha, verified with the mathematical software Maple, and then compared against published results for different formulations of the equations (Kelecy and Pletcher, 1997; Pan and Chang, 2000; Qian et al., 2006) and slightly different equations (Elsworth and Toro, 1992; Bhat and Mandal, 2019a).

First block

Consider the x-direction inviscid fluxes $\mathbf{F}_{x,inv}$. The Jacobian is

$$\mathbf{A}_x = \frac{\partial \mathbf{F}_{x,inv}}{\partial \mathbf{Q}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -u^2 & 2u & 0 & 0 & \beta \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ -u/\rho & 1/\rho & 0 & 0 & 0 \end{bmatrix} \tag{5.11}$$

with eigenvalues

$$\lambda_1 = \lambda_2 = \lambda_3 = u \tag{5.12}$$

$$\lambda_{4,5} = \frac{1}{2}(u \pm c) \tag{5.13}$$

where $c = \sqrt{u^2 + 4\beta/\rho}$. Note that this means $u = \lambda_4 + \lambda_5$ and $c = \lambda_4 - \lambda_5$. So the right and left eigenvector matrices are

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & \lambda_4 & \lambda_5 \\ u & 0 & 0 & u\lambda_4 + \beta/\rho & u\lambda_5 + \beta/\rho \\ 0 & 1 & 0 & v\lambda_4 & v\lambda_5 \\ 0 & 0 & 1 & w\lambda_4 & w\lambda_5 \\ 0 & 0 & 0 & -\lambda_5/\rho & -\lambda_4/\rho \end{bmatrix} \tag{5.14}$$

$$\mathbf{L}_x = \begin{bmatrix} 1 + u^2\rho/\beta & -\rho u/\beta & 0 & 0 & -\rho \\ u^2 v\rho/\beta & -uv\rho/\beta & 1 & 0 & -\rho v \\ u^2 w\rho/\beta & -uw\rho/\beta & 0 & 1 & -\rho w \\ -u\lambda_4\rho/\beta c & \lambda_4\rho/\beta c & 0 & 0 & \rho/c \\ u\lambda_5\rho/\beta c & -\lambda_5\rho/\beta c & 0 & 0 & -\rho/c \end{bmatrix}. \tag{5.15}$$

It is also useful to define the matrix

$$|\boldsymbol{\Lambda}_x| = \begin{bmatrix} |\lambda_1| & & & & \\ & |\lambda_2| & & & \\ & & |\lambda_3| & & \\ & & & |\lambda_4| & \\ & & & & |\lambda_5| \end{bmatrix}. \tag{5.16}$$

The results are similar for the second and third blocks, but with permutations of the matrix entries. This means that we can use a 3D rotation matrix and its inverse to convert Riemann problems into x-direction Riemann problems, which is covered later.

Second block

The Jacobian is

$$\mathbf{A}_y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -uv & v & u & 0 & 0 \\ -v^2 & 0 & 2v & 0 & \beta \\ -vw & 0 & w & v & 0 \\ -v/\rho & 0 & 1/\rho & 0 & 0 \end{bmatrix} \tag{5.17}$$

with eigenvalues

$$\lambda_1 = \lambda_2 = \lambda_3 = v \tag{5.18}$$

$$\lambda_{4,5} = \frac{1}{2}(v \pm c) \tag{5.19}$$

and eigenvectors

$$\mathbf{R}_y = \begin{bmatrix} 1 & 0 & 0 & \lambda_4 & \lambda_5 \\ 0 & 1 & 0 & u\lambda_4 & u\lambda_5 \\ v & 0 & 0 & v\lambda_4 + \beta/\rho & v\lambda_5 + \beta/\rho \\ 0 & 0 & 1 & w\lambda_4 & w\lambda_5 \\ 0 & 0 & 0 & -\lambda_5/\rho & -\lambda_4/\rho \end{bmatrix} \tag{5.20}$$

$$\mathbf{L}_y = \begin{bmatrix} 1 + v^2\rho/\beta & 0 & -\rho v/\beta & 0 & -\rho \\ v^2 u\rho/\beta & 1 & -vu\rho/\beta & 0 & -\rho u \\ v^2 w\rho/\beta & 0 & -vw\rho/\beta & 1 & -\rho w \\ -v\lambda_4\rho/\beta c & 0 & \lambda_4\rho/\beta c & 0 & \rho/c \\ v\lambda_5\rho/\beta c & 0 & -\lambda_5\rho/\beta c & 0 & -\rho/c \end{bmatrix}. \tag{5.21}$$

Third block

The Jacobian is

$$\mathbf{A}_z = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -uw & w & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ -w^2 & 0 & 0 & 2w & \beta \\ -w/\rho & 0 & 0 & 1/\rho & 0 \end{bmatrix} \tag{5.22}$$

with eigenvalues

$$\lambda_1 = \lambda_2 = \lambda_3 = w \tag{5.23}$$

$$\lambda_{4,5} = \frac{1}{2}(w \pm c) \tag{5.24}$$

and eigenvectors

$$\mathbf{R}_z = \begin{bmatrix} 1 & 0 & 0 & \lambda_4 & \lambda_5 \\ 0 & 1 & 0 & u\lambda_4 & u\lambda_5 \\ 0 & 0 & 1 & v\lambda_4 & v\lambda_5 \\ w & 0 & 0 & w\lambda_4 + \beta/\rho & w\lambda_5 + \beta/\rho \\ 0 & 0 & 0 & -\lambda_5/\rho & -\lambda_4/\rho \end{bmatrix} \tag{5.25}$$

$$\mathbf{L}_z = \begin{bmatrix} 1 + w^2\rho/\beta & 0 & 0 & -\rho w/\beta & -\rho \\ w^2 u\rho/\beta & 1 & 0 & -wu\rho/\beta & -\rho u \\ w^2 v\rho/\beta & 0 & 1 & -wv\rho/\beta & -\rho v \\ -w\lambda_4\rho/\beta c & 0 & 0 & \lambda_4\rho/\beta c & \rho/c \\ w\lambda_5\rho/\beta c & 0 & 0 & -\lambda_5\rho/\beta c & -\rho/c \end{bmatrix}. \tag{5.26}$$

5.3 Discretisation

5.3.1 Finite volume method

For transient equations such as (5.4), artificial compressibility uses dual time stepping, that is, driving the solution to the incompressible limit each real-time step. To advance the variables in pseudo-time, we follow the discretisation method of Vermeire et al. (2019), applying it to variable-density flows for the first time. Moving the real-time derivative to the right-hand side of the compact form (5.4), along with the fluxes and gravity source term, yields

$$\frac{\partial \mathbf{Q}}{\partial \tau} = -\nabla \cdot \mathbf{F} - \mathbf{I}_C \frac{\partial \mathbf{Q}}{\partial t} + \mathbf{B}. \tag{5.27}$$

Integrating over the computational cell Ω and applying Gauss's theorem results in the semi-discretised equation

$$\frac{\partial \mathbf{Q}_i}{\partial \tau} = -\frac{1}{|\Omega|} \sum_{f \in \partial\Omega} \mathbf{F}_f \cdot \mathbf{S}_f - \mathbf{I}_C \frac{\partial \mathbf{Q}_i}{\partial t} + \mathbf{B}_i \tag{5.28}$$

where the subscript i denotes cell averages and the subscript f face averages. Integrating over the pseudo-time interval $[\tau_0, \tau_1]$ and applying the Fundamental Theorem of Calculus, we get

$$\mathbf{Q}_i^{m+1} = \mathbf{Q}_i^m + \int_{\tau_0}^{\tau_1} \left(-\frac{1}{|\Omega|} \sum_{f \in \partial\Omega} \mathbf{F}_f \cdot \mathbf{S}_f - \mathbf{I}_C \frac{\partial \mathbf{Q}_i}{\partial t} + \mathbf{B}_i \right) d\tau. \tag{5.29}$$

This equation, still exact with no approximations, shows that the solution can march forward in pseudo-time with the real-time term $\mathbf{I}_C \cdot \partial \mathbf{Q}_i / \partial t$ treated as a source term.

5.3.2 Runge-Kutta pseudo-time stepping

The exact equation (5.29) is discretised in pseudo-time using the Jameson (1991) explicit Runge-Kutta multistage method, again as in Vermeire et al. (2019). Computationally, this means that there are three nested loops, as shown in Algorithm 5.1. The outer loop loops over real-time, the middle loops over pseudo-time, and the inner loops over Runge-Kutta stages. Each real-time iteration involves enough pseudo-time iterations to get the solution to converge within a prescribed tolerance, and each pseudo-time iteration involves a set number of Runge-Kutta stages. The real-time derivative is updated every pseudo-time step, which takes care of the real-time stepping point-implicitly, while pseudo-time is dealt with explicitly.

Formally, let $\mathbf{Q}^{n,m,s}$ denote the conserved variables $\mathbf{Q}$ evaluated at the nth real-time step, the mth pseudo-time iteration and the sth stage in the Runge-Kutta scheme. The

Algorithm 5.1 Three nested loops with the same array $\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, p/\beta]$ operated on throughout. Backward-Euler real-time derivative shown for illustration; higher-order methods require more real-time steps saved than $\mathbf{Q}^n$.

```
$t \leftarrow 0$
**while** $t < t_{max}$ **do**                                              ▷ start real-time loop
    $\mathbf{Q}^n \leftarrow \mathbf{Q}$
    $t \leftarrow t + \Delta t$
    $\tau \leftarrow 0$
    **while** residuals $>$ tolerance **do**                              ▷ start pseudo-time loop
        $\mathbf{Q}^m \leftarrow \mathbf{Q}$
        $\tau \leftarrow \tau + \Delta\tau$
        real-time term $\leftarrow \mathbf{I}_C \frac{Q - Q^n}{\Delta t}$
        body force $\leftarrow \rho\mathbf{g}$
        **for** $s = 1, ..., s_{max}$ **do**                                 ▷ start Runge-Kutta loop
            calculate $\alpha_s, \alpha_{PI}, \mathbf{F}_f$
            $\mathbf{Q} \leftarrow \mathbf{Q}^m + \Delta\tau \frac{\alpha_s}{\alpha_{PI}} \left( -\frac{1}{|\Omega|} \sum_f \mathbf{F}_f \cdot \mathbf{S}_f - \text{real-time term} + \text{body force} \right)$
        **end for**                                                      ▷ end Runge-Kutta loop
    **end while**                                                        ▷ end pseudo-time loop
**end while**                                                            ▷ end real-time loop
```

	B_0	B_1	B_2	B_3
Backward-Euler	1	-1	–	–
BDF2	$\frac{3}{2}$	-2	$\frac{1}{2}$	–
BDF3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$

Table 5.1: Backward differencing coefficients from Vermeire et al. (2019, Table 1) with typo fixed according to the standard textbook (Iserles, 2008, p.27).

variables are advanced in pseudo-time using the update equation

$$\mathbf{Q}_i^{n+1,m+1,0} = \mathbf{Q}_i^{n+1,m} \tag{5.30}$$

$$\begin{aligned}\mathbf{Q}_i^{n+1,m+1,s} &= \mathbf{Q}_i^{n+1,m+1,0} \\ &\quad + \frac{\alpha_s \Delta\tau}{\alpha_{PI}} \left(-\frac{1}{|\Omega|} \sum_{f\in\partial\Omega} \mathbf{F}_f^{n+1,m+1,s-1} \cdot \mathbf{S}_f - \mathbf{I}_C \frac{\partial}{\partial t}(\mathbf{Q}_i^{n+1,m+1,0}) + \mathbf{B}_i^{n+1,m+1,0} \right) \\ &\qquad \text{for } s = 1, \ldots, s_{max}\end{aligned} \tag{5.31}$$

$$\mathbf{Q}_i^{n+1,m+1} = \mathbf{Q}_i^{n+1,m+1,s_{max}} \tag{5.32}$$

where α_s is the Runge-Kutta coefficient. The last coefficient $\alpha_{s_{max}}$ must be equal to 1 and, for second-order accuracy in time, the second-last coefficient $\alpha_{s_{max}-1}$ must be equal to 1/2.

The real-time derivative in (5.31) is given by the backward differencing scheme

$$\frac{\partial}{\partial t}\left(\mathbf{Q}_i^{n+1,m+1,0}\right) = \frac{B_0 \mathbf{Q}_i^{n+1,m+1,0} + \sum_{\sigma=0}^{M} B_{\sigma+1} \mathbf{Q}_i^{n-\sigma}}{\Delta t} \tag{5.33}$$

where M is the order of the scheme and the coefficients B_σ are given in Table 5.1. Note that the first real-time derivative can only be Backward-Euler, the second real-time derivative Backward-Euler or BDF2, and so on. In any case, we only use Backward-Euler in this chapter. The effect of having the first term on the right-hand side of (5.33) be $\mathbf{Q}_i^{n+1,m+1,0}$ and not $\mathbf{Q}_i^{n+1,m+1,s}$ is remedied almost exactly with the "point-implicit scaling coefficient"

$$\alpha_{PI} = 1 + B_0 \alpha_S \frac{\Delta\tau}{\Delta t} \tag{5.34}$$

in (5.31), which comes from some rearranging and helps with stability when $\Delta\tau/\Delta t$ is large (Loppi, 2019; Vermeire et al., 2019). For the derivation, note that

$$\frac{\partial}{\partial t}\left(\mathbf{Q}_i^{n+1,m+1,s}\right) = \frac{B_0\mathbf{Q}_i^{n+1,m+1,s} + \sum_{\sigma=0}^{M} B_{\sigma+1}\mathbf{Q}_i^{n-\sigma}}{\Delta t} \tag{5.35}$$

$$= \frac{B_0\mathbf{Q}_i^{n+1,m+1,s} - B_0\mathbf{Q}_i^{n+1,m+1,0} + B_0\mathbf{Q}_i^{n+1,m+1,0} + \sum_{\sigma=0}^{M} B_{\sigma+1}\mathbf{Q}_i^{n-\sigma}}{\Delta t} \tag{5.36}$$

$$= \frac{B_0}{\Delta t}\left(\mathbf{Q}_i^{n+1,m+1,s} - \mathbf{Q}_i^{n+1,m+1,0}\right) + \frac{\partial}{\partial t}\left(\mathbf{Q}_i^{n+1,m+1,0}\right) \tag{5.37}$$

and, ignoring $\mathbf{I}_C$, the term

$$\frac{B_0}{\Delta t}\left(\mathbf{Q}_i^{n+1,m+1,s} - \mathbf{Q}_i^{n+1,m+1,0}\right) \tag{5.38}$$

can be collected with like terms in (5.31) to obtain α_{PI}. Ignoring the cancellation matrix in this derivation does not affect the converged solution. Indeed, Vermeire et al. (2019) set $\alpha_{PI} = 1$ for constant density flows but, in the present study of variable density flows, it was vital for stability to use the correct value (5.34).

An explicit scheme was chosen for pseudo-time because explicit schemes have numerical advantages compared to implicit schemes. Indeed, while implicit schemes are unconditionally stable, they are less accurate than explicit schemes when large time steps (or Courant numbers) are used (Fernández-Pato et al., 2018). Moreover, in implicit schemes it is impossible to determine in advance the upper limit of the Courant number before the accuracy starts to decrease (Morales-Hernández et al., 2014).

The specific time discretisation chosen also has computational advantages. First, as the scheme is explicit in pseudo-time, it is easy to parallelise (Loppi, 2019; Vermeire et al., 2019). Second, as it is point-implicit in real-time, large real-time steps are possible (Loppi, 2019). Third, the specific multistage Runge-Kutta method it uses has low memory requirements. Essentially, the same array $\mathbf{Q}^{n,m,s}$ gets operated on through the real-time, pseudo-time, and Runge-Kutta loops. Only the initial Runge-Kutta stage (5.30) and the bracketed part of the update equation (5.31) for the latest iteration need to be stored (Jameson, 1991; Blazek, 2005). Consequently, there is potential for the method to be implemented very efficiently.

5.3.3 Local pseudo-time steps

As the pseudo-time stepping is explicit, it has the well-known stability restriction of

$$\mathrm{CFL}_\tau \leq \begin{cases} 1 & \text{if 1D} \\ \frac{1}{2} & \text{if 2D} \\ \frac{1}{4} & \text{if 3D} \end{cases} \tag{5.39}$$

where CFL_τ is the Courant number for pseudo-time. However, slightly smaller Courant numbers (smaller by around 0.1–0.5) may be required to account for the fact that the restriction is based on the last pseudo-time step and velocities may increase in the current pseudo-time step.

Using one global time step constrained by the most extreme restriction across the mesh will slow down convergence substantially. To see this, consider a variable-density case with a high density ratio, such as water and air. The acoustic wave speed will be much lower through the water than the air, and consequently pseudo-time convergence will be much slower in the water than the air (Pelanti, 2017). Local pseudo-time stepping allows the water and air to converge at similar speeds while still meeting the stability restriction in each cell. Moreover, if the convective speed $\frac{1}{2}u$ is small, then the Mach number u/c will be low. This will lead to slow convergence and the need for a Low Mach preconditioner. However, artificial compressibility is a form of Low Mach preconditioning, and so all that is required is a large enough β (Guillard and Viozat, 1999; Turkel, 1993; Drikakis and Rider, 2005; Rahman and Siikonen, 2008).

In the absence of viscosity and turbulence, a stable local pseudo-time step can be given by

$$\Delta\tau_i = \mathrm{CFL}_\tau \frac{|\Omega_i|}{(\hat{\Lambda}_{\tau,x} + \hat{\Lambda}_{\tau,y} + \hat{\Lambda}_{\tau,z})_i} \tag{5.40}$$

where

$$\hat{\Lambda}_{\tau,x} = \frac{1}{2}(|u| + c)\Delta\hat{S}_x \tag{5.41}$$

$$\hat{\Lambda}_{\tau,y} = \frac{1}{2}(|v| + c)\Delta\hat{S}_y \tag{5.42}$$

$$\hat{\Lambda}_{\tau,z} = \frac{1}{2}(|w| + c)\Delta\hat{S}_z \tag{5.43}$$

and

$$\Delta\hat{S}_x = \frac{1}{2}\sum_f |S_x|_f \tag{5.44}$$

$$\Delta\hat{S}_y = \frac{1}{2}\sum_f |S_y|_f \tag{5.45}$$

$$\Delta\hat{S}_z = \frac{1}{2}\sum_f |S_z|_f \tag{5.46}$$

define the spectral radii for the convective fluxes (Blazek, 2005). The division by two in (5.41)–(5.43) is because of the definition of $\lambda_{4,5}$, while the division by two in (5.44)–(5.46) is because $\Delta\hat{S}$ is the projection of the entire computational cell Ω onto the relevant plane.

5.3.4 Global real-time steps

As real-time is treated point-implicitly, there is no stability restriction as there is for pseudo-time, which is explicit. Nonetheless, it is useful to be able to set how relatively large the real-time steps are. So, the global real-time step can be given by

$$\Delta t = \min_i \left(\mathrm{CFL}_t \frac{|\Omega_i|}{(\hat{\Lambda}_{t,x} + \hat{\Lambda}_{t,y} + \hat{\Lambda}_{t,z})_i} \right) \tag{5.47}$$

where CFL_t is the Courant number for real-time,

$$\hat{\Lambda}_{t,x} = |u| \cdot \Delta\hat{S}_x \tag{5.48}$$

$$\hat{\Lambda}_{t,y} = |v| \cdot \Delta\hat{S}_y \tag{5.49}$$

$$\hat{\Lambda}_{t,z} = |w| \cdot \Delta\hat{S}_z \tag{5.50}$$

and $\Delta\hat{S}_x, \Delta\hat{S}_y, \Delta\hat{S}_z$ are as previously defined in (5.44)–(5.46).

5.4 Reconstruction at cell interfaces

In this study, we use standard MUSCL reconstruction (van Leer, 1976) to find Riemann states of density and velocity at each cell interface, as in Kelecy and Pletcher (1997) and Qian et al. (2006). However, special treatment is required to reconstruct pressure at the free surface as the density discontinuity induces a pressure gradient discontinuity. This section outlines the standard and pressure-specific approaches in turn. Note that here we use TVD limiters in a different way than how they are used in OpenFOAM as described in Chapters 3–4.

5.4.1 Standard reconstruction for structured meshes

The primitive variables $\mathbf{W}$ are reconstructed at the interface f between the cells i and j to give the left and right values

$$\mathbf{W}_L = \mathbf{W}_i + \nabla \mathbf{W}_i \cdot (\mathbf{x}_f - \mathbf{x}_i) \tag{5.51}$$

$$\mathbf{W}_R = \mathbf{W}_j + \nabla \mathbf{W}_j \cdot (\mathbf{x}_f - \mathbf{x}_j) \tag{5.52}$$

where a TVD limiter must be applied to $\nabla \mathbf{W}$ to avoid Gibbs-type oscillations. In the x-direction of a structured mesh, this reduces to

$$\nabla \mathbf{W}_i = G\left(\frac{\mathbf{W}_{i+1} - \mathbf{W}_i}{dx/2}, \frac{\mathbf{W}_i - \mathbf{W}_{i-1}}{dx/2}\right) \tag{5.53}$$

where G is the limiter function. We use the k-limiter

$$G(a, b) = s \cdot \max(0, \min(k|b|, s \cdot a), \min(|b|, ks \cdot a)) \tag{5.54}$$

$$s = \operatorname{sgn}(b) \tag{5.55}$$

where $k = 1$ is the minmod limiter and $k = 2$ the superbee limiter (Qian et al., 2006). This standard limiter is suitable for reconstructing the density and velocity.

5.4.2 Standard reconstruction for unstructured meshes

General 3D unstructured meshes require a slightly different approach (Blazek, 2005). In the context of a Godunov-type scheme, the idea is to construct a limiter function $\Phi \in [0, 1]$ to be used as

$$\mathbf{W}_{f,i} = \mathbf{W}_i + \Phi_i \nabla \mathbf{W}_i \cdot (\mathbf{x}_f - \mathbf{x}_i) \tag{5.56}$$

where $\nabla \mathbf{W}_i$ is now the gradient of the primitive variables before limiting and $\mathbf{W}_{f,i}$ is the Riemann state at face f on the side belonging to cell i. Then the Riemann states either side of the face can be put into the Riemann solver to get the numerical flux. We consider this further in Chapter 6 when we implement the method on unstructured meshes.

5.4.3 Existing methods for pressure gradient

Without special treatment, standard MUSCL reconstruction—structured or unstructured—will not account for the impact of a density discontinuity on the pressure gradient. This can result in an imbalance of forces at the free surface and, in turn, either parasitic velocities (DeGroot and Straatman, 2011; Efremov et al., 2017; Hashemi et al., 2020) or erroneous compression at what should be the incompressible limit. As will be shown later, the compression can resemble Gibbs-type oscillations, but it is a distinct problem of its own that persists even with the most severe of limiters. Only Qian et al. (2006) have addressed this problem for the variable-density artificial-compressibility equations, and only in the hydrostatic case. However, the problem persists in the absence of gravity. Both the hydrostatic and non-hydrostatic problems are considered below, before outlining a new method for calculating the pressure gradient based on a rearrangement of the momentum equation.

Hydrostatic

First, consider a stationary column of air overlaying a stationary column of water under the influence of gravity. The momentum equation should settle to $\nabla p = \rho \mathbf{g}$ at the incompressible limit, that is, a hydrostatic pressure distribution. A scheme that can reproduce such a steady state (up to machine precision) is called well-balanced (Botta et al., 2004).

It is the free surface that complicates matters. The jump in density leads to a jump in ∇p which, if not calculated carefully, can lead to spurious or parasitic velocities (DeGroot and Straatman, 2011; Efremov et al., 2017; Hashemi et al., 2020). In a Godunov-type scheme, the problem arises because the gradients calculated in the MUSCL reconstruction step indiscriminately include information from both sides of the free surface, as illustrated in Figure 5.1. No matter the order of interpolation or refinement of the grid, the MUSCL reconstruction phase will always have this problem, but the impact will be on a different scale (Krause, 2019). Broadly, there are two ways to deal with this: either at the global level or the local level (Zingale et al., 2002). The global method removes hydrostatic pressure from the governing equations, while the local method involves including hydrostatic

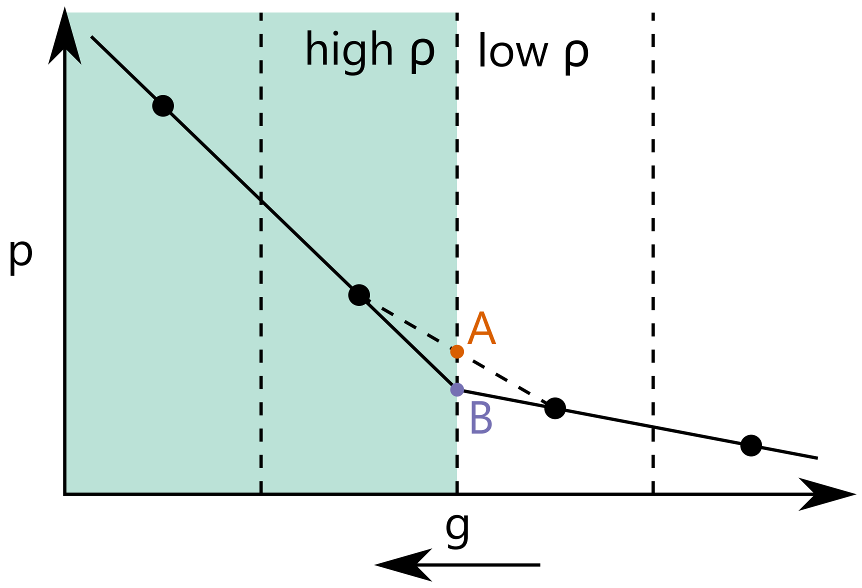

Figure 5.1: Jump in the hydrostatic pressure gradient at high density ratios, based on Qian et al. (2006, p.26) who note that, without special treatment, slope limiters will give a value of p somewhere between points A and B.

pressure in the reconstruction phase (Qian et al., 2006; Botta et al., 2004; Krause, 2019; Fuchs et al., 2010). Similarly, in the Smooth Particle Hydrodynamics (SPH) method, the problem is caused by the domain of the kernel function crossing over the air-water interface (Kruisbrink et al., 2018). This can be addressed locally by introducing a correction term (Kruisbrink et al., 2018; Zhou et al., 2016), dividing by the density (Luo et al., 2015), or changing the weighting in the kernel function (Woolfson, 2007).

In their local method for a Godunov-type scheme, Qian et al. (2006) split the full (dynamic) pressure gradient into hydrostatic and kinematic parts

$$\nabla p^{dyn} = \nabla p^{hyd} + \nabla p^{kin}. \tag{5.57}$$

For structured meshes, they first calculated the dynamic gradients $\nabla p_{i,L}^{dyn}$ and $\nabla p_{i,R}^{dyn}$ with one-sided differences. They then subtracted the hydrostatic gradients $\nabla p_{i,L}^{hyd} = (\rho_{i-1} + \rho_i)\mathbf{g}/2$ and $\nabla p_{i,R}^{hyd} = (\rho_i + \rho_{i+1})\mathbf{g}/2$ from the dynamic gradients $\nabla p_{i,L}^{dyn}$ and $\nabla p_{i,R}^{dyn}$ to get the kinematic gradients $\nabla p_{i,L}^{kin}$ and $\nabla p_{i,R}^{kin}$. They then applied a standard limiter to the kinematic gradients $\nabla p_{i,L}^{kin}$ and $\nabla p_{i,R}^{kin}$ to get the limited kinematic gradient ∇p_i^{kin}. Finally, they added the hydrostatic gradient $\nabla p_i^{hyd} = \rho_i \mathbf{g}$ to the limited kinematic gradient ∇p_i^{kin}

to get the limited dynamic pressure gradient ∇p_i^{dyn}. This method successfully resolves the pressure gradient discontinuity in the hydrostatic case, keeping the scheme well-balanced. However, as it is complicated to implement standard limiters on unstructured meshes (Barth and Jespersen, 1989; Venkatakrishnan, 1995), it is even more complicated for this limiter. A more easily implementable limiter would be preferable for practical applications that require unstructured grids. Moreover, this limiter does not address non-hydrostatic pressure gradient discontinuities. Nonetheless, it was implemented in this chapter for the purpose of comparison.

Non-hydrostatic

There is still a pressure gradient discontinuity in the absence of gravity. Let us denote the jump in any variable x over the free surface by $[x]$. The kinematic boundary condition, given by $[\mathbf{u}] = 0$, can be combined with the momentum equation to derive the dynamic boundary condition, given by $[\nabla p/\rho] = 0$ (Ntouras and Papadakis, 2020; Queutey and Visonneau, 2007; Kruisbrink et al., 2018; Vukčević et al., 2018). Since $[\rho] \neq 0$, we must also have $[\nabla p] \neq 0$ to satisfy the dynamic boundary condition. Kruisbrink et al. (2018) bypassed this problem by counteracting spurious forces with a quasi-buoyancy force, while Vukčević et al. (2018) used the ghost fluid method to interpolate variables in a one-sided manner consistent with the dynamic boundary condition. The difficulty with these methods is that they require a method to detect the free surface, which is ambiguous in pseudo-time as the water is allowed to artificially compress. Alternatively, the pressure gradient can be normalised by the density (Ntouras and Papadakis, 2020; Queutey and Visonneau, 2007).

5.4.4 New method for pressure gradient

Here, we introduce a new method for calculating the pressure gradient. Without loss of generality, consider the x-direction momentum equation in the absence of viscosity. Note that the pseudo-time derivatives disappear at the incompressible limit. The momentum equation can then be rearranged, giving

$$\frac{\partial p}{\partial x} = -\frac{\partial}{\partial t}(\rho u) - \frac{\partial}{\partial x}(\rho u^2) - \frac{\partial}{\partial y}(\rho u v) - \frac{\partial}{\partial z}(\rho u w) + \rho \mathbf{g}_x. \tag{5.58}$$

All the terms on the right-hand side are known. First, the real-time derivative $\partial(\rho u)/\partial t$ is already calculated as a source term. Second, the spatial derivatives can be expanded using the product and chain rules to get

$$\frac{\partial}{\partial x}(\rho u^2) = u^2\frac{\partial}{\partial x}(\rho) + 2\rho u\frac{\partial}{\partial x}(u) \tag{5.59}$$

$$\frac{\partial}{\partial y}(\rho uv) = uv\frac{\partial}{\partial y}(\rho) + \rho u\frac{\partial}{\partial y}(v) + \rho v\frac{\partial}{\partial y}(u) \tag{5.60}$$

$$\frac{\partial}{\partial z}(\rho uw) = uw\frac{\partial}{\partial z}(\rho) + \rho u\frac{\partial}{\partial z}(w) + \rho w\frac{\partial}{\partial z}(u) \tag{5.61}$$

where the density and velocity are set explicitly, and the derivatives $\partial/\partial x$, $\partial/\partial y$, and $\partial/\partial z$ are given by the TVD gradients already calculated for the density and velocity. Formally,

$$\begin{aligned}(\nabla p)_i = -\left(\frac{\partial}{\partial t}(\rho\mathbf{u})\right)_i + \\ \begin{bmatrix} -u_i^2\left(\frac{\partial\rho}{\partial x}\right)_i-2\rho_i u_i\left(\frac{\partial u}{\partial x}\right)_i-u_i v_i\left(\frac{\partial\rho}{\partial y}\right)_i-\rho_i u_i\left(\frac{\partial v}{\partial y}\right)_i-\rho_i v_i\left(\frac{\partial u}{\partial y}\right)_i-u_i w_i\left(\frac{\partial\rho}{\partial z}\right)_i-\rho_i u_i\left(\frac{\partial w}{\partial z}\right)_i-\rho_i w_i\left(\frac{\partial u}{\partial z}\right)_i \\ -u_i v_i\left(\frac{\partial\rho}{\partial x}\right)_i-\rho_i u_i\left(\frac{\partial v}{\partial x}\right)_i-\rho_i v_i\left(\frac{\partial u}{\partial x}\right)_i-v_i^2\left(\frac{\partial\rho}{\partial y}\right)_i-2\rho_i v_i\left(\frac{\partial v}{\partial y}\right)_i-v_i w_i\left(\frac{\partial\rho}{\partial z}\right)_i-\rho_i v_i\left(\frac{\partial w}{\partial z}\right)_i-\rho_i w_i\left(\frac{\partial v}{\partial z}\right)_i \\ -u_i w_i\left(\frac{\partial\rho}{\partial x}\right)_i-\rho_i u_i\left(\frac{\partial w}{\partial x}\right)_i-\rho_i w_i\left(\frac{\partial u}{\partial x}\right)_i-v_i w_i\left(\frac{\partial\rho}{\partial y}\right)_i-\rho_i v_i\left(\frac{\partial w}{\partial y}\right)_i-\rho_i w_i\left(\frac{\partial v}{\partial y}\right)_i-w_i^2\left(\frac{\partial\rho}{\partial z}\right)_i-2\rho_i w_i\left(\frac{\partial w}{\partial z}\right)_i \end{bmatrix} \\ + \rho_i\mathbf{g}\end{aligned} \tag{5.62}$$

where, again, the real-time derivative $(\partial/\partial t)_i$ has already been calculated as part of the source term (5.33), and the spatial derivatives $(\partial/\partial x)_i$, $(\partial/\partial y)_i$, and $(\partial/\partial z)_i$ have already been approximated with standard TVD gradients (5.53) or (5.56).

This pressure gradient calculation takes advantage of the real-time source term to ensure that the TVD interpolation satisfies the differential form of the momentum equation at the incompressible limit. Moreover, when the velocity is zero, the momentum equation (5.58) reduces to

$$\frac{\partial p}{\partial x} = \rho\mathbf{g}_x, \tag{5.63}$$

that is, local hydrostatic reconstruction, and so the method is well-balanced. Furthermore, it is straightforward to implement on unstructured meshes, unlike the split limiter. Of course, it does not include the pseudo-time term as pseudo-time is explicit. However, as long as there is convergence over pseudo-time, it is the incompressible limit that matters rather than how the solution is reached.

5.5 Flux calculation

The inviscid flux $\mathbf{F}_{inv}$ is calculated with a Riemann solver. This study compares the Bassi, Roe, and Osher Riemann solvers. The viscous flux $\mathbf{F}_{vis}$ can be calculated easily using standard finite difference methods (Bhat and Mandal, 2019a; Kelecy and Pletcher, 1997), but the effect of viscosity is ignored in this chapter as only convection-dominated flows are considered in the benchmark tests. Turbulence is also ignored.

First note that, for cell faces with normal vectors pointing in any other direction than the x-axis, we use 3D rotation matrices to convert the Riemann problem to a x-direction Riemann problem as in Tanaka (1994). At each face, the rotation matrix is given by

$$\mathbf{R} = \begin{bmatrix} 1 & & & & \\ & n_x & n_y & n_z & \\ & t_{1,x} & t_{1,y} & t_{1,z} & \\ & t_{2,x} & t_{2,y} & t_{2,z} & \\ & & & & 1 \end{bmatrix} \tag{5.64}$$

where $\mathbf{n}$ is the unit normal face vector, and $\mathbf{t}_1$ and $\mathbf{t}_2$ are unit vectors tangential to $\mathbf{n}$, making an orthonormal set $\{\mathbf{n}, \mathbf{t}_1, \mathbf{t}_2\}$. Given any face vector $\mathbf{n}$, there are infinite possibilities for $\mathbf{t}_1$ and $\mathbf{t}_2$—one possibility is given in Algorithm 5.2. The Riemann states are then $\mathbf{Q}_L = \mathbf{R} \cdot \mathbf{Q}_a$ and $\mathbf{Q}_R = \mathbf{R} \cdot \mathbf{Q}_b$, assuming $\mathbf{n}$ points out of cell a and into cell b. After flux calculation, we use the inverse rotation matrices to transform the flux back to the original reference frame. As $\mathbf{R}$ is a rotation matrix, its inverse is $\mathbf{R}^{-1} = \mathbf{R}^T$. Thus the flux is

$$\mathbf{F}_{inv} = \mathbf{R}^T \cdot \mathbf{F}_{x,inv}(\mathbf{R} \cdot \mathbf{Q}_a, \mathbf{R} \cdot \mathbf{Q}_b) \tag{5.65}$$

where $\mathbf{F}_{x,inv}(\mathbf{Q}_L, \mathbf{Q}_R)$ is the Riemann solver for the x-direction inviscid fluxes.

5.5.1 Bassi Riemann solver

Bassi et al. (2018) assumed that the contact wave lies between the acoustic (shock and/or rarefaction) waves to derive the Riemann invariants and Rankine-Hugoniot conditions for the variable-density artificial-compressibility equations. They found that the equations have the unusual property of shocks and rarefactions travelling at the same speed, with

Algorithm 5.2 One possibility for calculating $\mathbf{t}_1$ and $\mathbf{t}_2$ from $\mathbf{n}$.

```
if $n_x == 0$ then
    $\mathbf{t}_1 \leftarrow (1, 0, 0)$
else if $n_y == 0$ then
    $\mathbf{t}_1 \leftarrow (0, 1, 0)$
else
    $\mathbf{t}_1 \leftarrow \frac{1}{\sqrt{n_x^2+n_y^2}}(n_y, -n_x, 0)$
end if
$\mathbf{t}_2 \leftarrow \mathbf{n} \times \mathbf{t}_1$
```

the head and tail of the rarefaction coinciding, and the values of the conserved variables in the star region independent of the wave type. That is, shocks and rarefactions are equivalent, which means that the exact solution of the Riemann problem can be found non-iteratively. Consequently, using such an algorithm to calculate the numerical flux at cell interfaces (i.e. using an exact Riemann solver) is not as prohibitively expensive as for other governing equations.

The exact Riemann solver is as follows. Without loss of generality, for the x-split equations, the left and right acoustic wave speeds are

$$S_L = \frac{1}{2}(u_L - c_L) \tag{5.66}$$

$$S_R = \frac{1}{2}(u_R + c_R) \tag{5.67}$$

regardless of whether they are shocks or rarefactions. In the star region, we have

$$u_* = \frac{p_R - p_L + \rho_R u_R \lambda_{5,R} - \rho_L u_L \lambda_{4,L}}{\rho_R \lambda_{5,R} - \rho_L \lambda_{4,L}} \tag{5.68}$$

$$p_* = p_R + \rho_R \lambda_{5,R}(u_R - u_*) \tag{5.69}$$

$$\rho_{*L} = \frac{\rho_L \lambda_{4,L}}{u_* - \lambda_{5,L}} \tag{5.70}$$

$$\rho_{*R} = \frac{\rho_R \lambda_{5,R}}{u_* - \lambda_{4,R}} \tag{5.71}$$

again regardless of whether the acoustic waves are shocks or rarefactions, and the tangential velocities only change across the contact wave travelling with speed u_*, that is,

$$v_{*L} = v_L \tag{5.72}$$

$$v_{*R} = v_R \tag{5.73}$$

$$w_{*L} = w_L \tag{5.74}$$

$$w_{*R} = w_R. \tag{5.75}$$

The exact Riemann solver determines the location of the waves with respect to the τ-axis, and then substitutes the appropriate values of density, velocity, and pressure into the flux function $\mathbf{F}_{x,inv}$.

As this exact solution is so simple, it has been used to calculate numerical fluxes in some discontinuous Galerkin models (Bassi et al., 2018; Manzanero et al., 2020). However, Bassi et al. (2018, Appendix B4) did acknowledge that it is possible for the contact wave to overtake the acoustic wave, that is, either $u_* < \lambda_{5,L}$ or $u_* > \lambda_{4,R}$, meaning that the assumptions break down and the Riemann solver is no longer exact. Consequently, approximate Riemann solvers that do not make this assumption must be investigated as well.

Moreover, this suggests that caution is needed even when using such approximate Riemann solvers, which do not assume specific ordering of the waves. This is because, if the contact wave overtakes the acoustic wave, the standard CFL condition based solely on the acoustic waves could result in a time step too large for stability. However, this seems not to have been a problem in Kelecy and Pletcher (1997), Qian et al. (2006), and Bassi et al. (2018).

While Bassi et al. (2018) do note that there will be critical values of the artificial compressibility parameter β for their exact Riemann solver to be valid, they do not provide details, saying that it will form a part of future work. However, there are some straightforward special cases that help shed light on the problem.

Zero velocity

If $u_L = u_R = 0$, we need

$$-\frac{1}{2}\sqrt{4\beta/\rho_L} < \frac{p_R - p_L}{-\rho_R\frac{1}{2}\sqrt{4\beta/\rho_R} - \rho_L\frac{1}{2}\sqrt{4\beta/\rho_L}} < \frac{1}{2}\sqrt{4\beta/\rho_R} \tag{5.76}$$

which, if $p_L > p_R$, is equivalent to

$$\frac{p_R - p_L}{1 + \sqrt{\rho_L/\rho_R}} < \frac{p_L - p_R}{1 + \sqrt{\rho_R/\rho_L}} < \beta, \tag{5.77}$$

and, if $p_R < p_L$, is equivalent to

$$\frac{p_L - p_R}{1 + \sqrt{\rho_R/\rho_L}} < \frac{p_R - p_L}{1 + \sqrt{\rho_L/\rho_R}} < \beta. \tag{5.78}$$

Taking these two together, we have the condition

$$\beta > \begin{cases} \frac{p_L - p_R}{1+\sqrt{\rho_R/\rho_L}} & \text{if } p_L > p_R \text{ (corresponds to right wave)} \\ \frac{p_R - p_L}{1+\sqrt{\rho_L/\rho_R}} & \text{if } p_R > p_L \text{ (corresponds to left wave)} \end{cases} \tag{5.79}$$

for the Riemann solver to be valid.

Equal velocity and pressure

If $p_L = p_R$ and $u_L = u_R = u_c$, then

$$u_* = \frac{u_c(\rho_R\lambda_{5,R} - \rho_L\lambda_{4,L})}{\rho_R\lambda_{5,R} - \rho_L\lambda_{4,L}} = u_c \tag{5.80}$$

and so we need

$$\frac{1}{2}(u_c - \sqrt{u_c^2 + 4\beta/\rho_L}) < u_c < \frac{1}{2}(u_c + \sqrt{u_c^2 + 4\beta/\rho_R}). \tag{5.81}$$

Since β and ρ are always positive, we always have

$$|u_c| < \sqrt{u_c^2 + 4\beta/\rho}, \tag{5.82}$$

which means the Riemann solver is always valid.

Convenient ratio

If $p_L = p_R$ and $u_L\sqrt{\rho_L} = -u_R\sqrt{\rho_R}$, then

$$u_* = \frac{\rho_R u_R \lambda_{5,R} - \rho_L u_L \lambda_{4,L}}{\rho_R \lambda_{5,R} - \rho_L \lambda_{4,L}} = 0 \tag{5.83}$$

because, for the numerator of u_*,

$$\rho_R u_R \lambda_{5,R} - \rho_L u_L \lambda_{4,L} = \frac{1}{2}\left(\rho_R u_R \left(u_R - \sqrt{u_R^2 - 4\beta/\rho_R}\right) - \rho_L u_L \left(u_L + \sqrt{u_L^2 + 4\beta/\rho_L}\right)\right) \tag{5.84}$$

$$= \frac{1}{2}\left(\rho_R u_R^2 - \rho_L u_L^2 - u_R\sqrt{\rho_R}\sqrt{\rho_R u_R^2 - 4\beta} - u_L\sqrt{\rho_L}\sqrt{\rho_L u_L^2 - 4\beta}\right) \tag{5.85}$$

$$= 0 \tag{5.86}$$

and so the Riemann solver is always valid.

5.5.2 Roe Riemann solver

The Roe Riemann solver (Roe, 1981) is an approximate method that captures the contact wave. It is given by

$$\mathbf{F}_{x,inv} = \frac{1}{2}\left(\mathbf{F}_{x,inv}(\mathbf{Q}_R) + \mathbf{F}_{x,inv}(\mathbf{Q}_L) - |\tilde{\mathbf{A}}_x|\left(\mathbf{Q}_R - \mathbf{Q}_L\right)\right) \tag{5.87}$$

where

$$|\tilde{\mathbf{A}}_x| = \tilde{\mathbf{R}}_x|\tilde{\boldsymbol{\Lambda}}_x|\tilde{\mathbf{L}}_x \tag{5.88}$$

is evaluated at $\tilde{\mathbf{Q}}$ and the tilde denotes the Roe average. Kelecy and Pletcher (1997) and Qian et al. (2006) used this Riemann solver for the variable-density artificial-compressibility equations, with the Roe averages

$$\tilde{\rho} = \sqrt{\rho_L \rho_R} \tag{5.89}$$

$$\tilde{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{5.90}$$

$$\tilde{v} = \frac{\sqrt{\rho_L}v_L + \sqrt{\rho_R}v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{5.91}$$

$$\tilde{w} = \frac{\sqrt{\rho_L}w_L + \sqrt{\rho_R}w_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}. \tag{5.92}$$

Pressure does not appear in the eigenstructure so it does not need a Roe average. This Riemann solver has also been used for the artificial compressibility equations paired with VOF (Pan and Chang, 2000; Ntouras and Papadakis, 2020).

Generally, the Roe Riemann solver should have an entropy fix to account for transonic rarefactions (Toro, 2009), but this was not considered by either Kelecy and Pletcher (1997) or Qian et al. (2006). However, as mentioned above, the rarefaction fans for the variable-density artificial-compressibility equations have the strange property of the head and tail speeds coinciding (Bassi et al., 2018). Consequently, transonic rarefactions are impossible, and so an entropy fix is not needed.

5.5.3 Osher Riemann solver

The Osher Riemann solver (Osher and Solomon, 1982) is another approximate method that captures the contact wave. It is given by

$$\mathbf{F}_{x,inv} = \frac{1}{2}\left(\mathbf{F}_{x,inv}(\mathbf{Q}_R) + \mathbf{F}_{x,inv}(\mathbf{Q}_L) - \int_{Q_L}^{Q_R} |\mathbf{A}_x| \, d\mathbf{Q}\right) \tag{5.93}$$

where the integration path must be determined. Traditionally, the idea is to construct a path passing through each of the intermediate states in the wave structure, but this can be complicated. Dumbser and Toro (2011) provide a simpler method that only requires the eigenstructure of the equations, and not the specific wave structure of the Riemann problem. They integrate along a straight line in phase space using three-point Gaussian quadrature

$$\int_{Q_L}^{Q_R} |\mathbf{A}_x| \, d\mathbf{Q} = \frac{1}{2}(\mathbf{Q}_R - \mathbf{Q}_L) \sum_{j=1}^{3} w_j |\mathbf{A}_{x,j}| \tag{5.94}$$

where

$$|\mathbf{A}_{\mathrm{x,j}}| = \mathbf{R}_{\mathrm{x,j}} |\boldsymbol{\Lambda}_{\mathrm{x,j}}| \mathbf{L}_{\mathrm{x,j}} \tag{5.95}$$

is evaluated at

$$\mathbf{Q}_j = \mathbf{Q}_L + (\mathbf{Q}_R - \mathbf{Q}_L)\left(\frac{1}{2}(\xi_j + 1)\right) \tag{5.96}$$

with the integration points $\xi_1 = -\sqrt{3/5}, \xi_2 = 0, \xi_3 = \sqrt{3/5}$ and corresponding weights $w_1 = 5/9, w_2 = 8/9, w_3 = 5/9$. This simplification of the Osher Riemann solver has been shown to be robust for the Baer-Nunziato equations (Dumbser and Toro, 2011), the Euler equations (Lee et al., 2013), and the shallow water equations (Glenis et al., 2018), all of which feature a contact wave. In this paper, we implement this Riemann solver for the variable-density artificial-compressibility equations for the first time.

5.6 2D Cartesian meshes

The numerical method was implemented in Python for 2D Cartesian meshes. This implementation includes all three Riemann solvers outlined above: Bassi, Roe, and Osher. It also includes five gradient calculations for the MUSCL reconstruction step: minmod and superbee, hydrostatic splitting with minmod and superbee applied to the kinematic pressure, and the new pressure gradient method valid at the incompressible limit. Moreover, it takes advantage of object orientation to create objects at run time using class factories, and then access their methods using polymorphism. This structure means that, as in Glenis et al. (2018), if/else statements get moved out of functions called many times every iteration to functions called once at the start of the model run, improving efficiency, readability, and maintainability.

The following benchmark tests illustrate how different Riemann solvers and pressure gradients behave, as well as demonstrating the validity of the general method in modelling free-surface flow. Viscosity and turbulence are ignored throughout and the real-time backward differencing scheme, if used, is set to Backward-Euler. This allows the focus to be on the developments of this chapter—Riemann solvers and pressure gradients—before implementing the method in OpenFOAM for unstructured 3D meshes in Chapter 6.

5.6.1 Boundary conditions

In the Python prototype, the boundary conditions are implemented with ghost cells, rather than prescribing the boundary flux directly. This means that the Riemann solver automatically calculates the correct flux. At walls, a zero-gradient condition is applied to density and a no-slip condition to velocity, although negating the tangential velocity will have no effect in the absence of viscosity.

Pressure, again, is not so simple. Consider the hydrostatic case, as in Figure 5.2. A

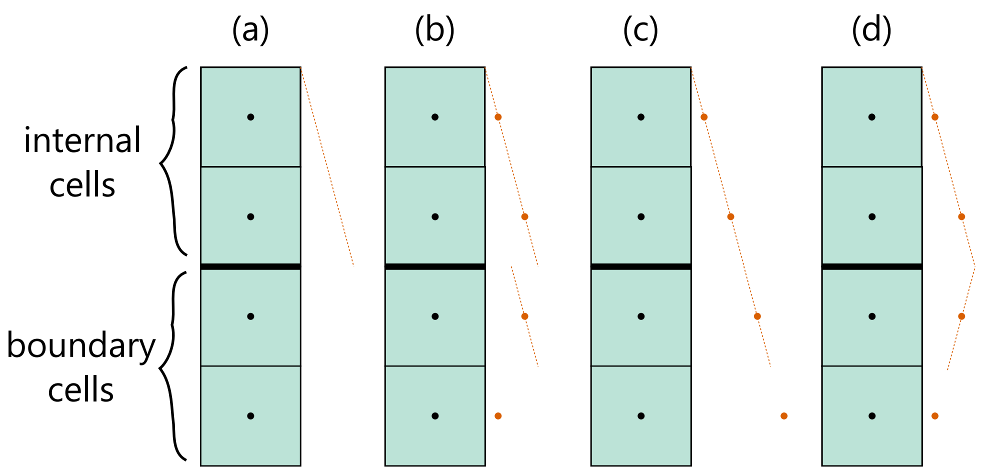

Figure 5.2: Options for setting pressure in the ghost cells, illustrated here for a hydrostatic case. From left to right: (a) reality, (b) zero-gradient boundary condition with local hydrostatic reconstruction, (c) linear or hydrostatic extrapolation boundary condition, (d) zero-gradient boundary condition with local hydrostatic reconstruction (reversed gravity).

zero-gradient condition naively applied to pressure, as in Bhat and Mandal (2019a), will underestimate pressure at the lower wall, meaning that there is not enough support for the fluid above (Zingale et al., 2002), and so the solution will settle to the wrong value at the incompressible limit. One way to overcome this is to interpolate the pressure to the ghost cells, either linearly (Bhat and Mandal, 2019b) or hydrostatically (Qian et al., 2006). However, interpolation has disadvantages. First, it requires calculating the gradient twice: once for the MUSCL reconstruction and once for ghost cells. Second, for hydrostatic interpolation, while the normal velocity is indeed zero at the wall, it is not zero throughout the whole cell, and so the kinematic pressure gradient is ignored. In this chapter, we use an alternative method, acceptable only when gravity forms part of the pressure gradient calculation in the MUSCL reconstruction stage, which is indeed the case for the split pressure (Qian et al., 2006) and new incompressible limit methods. The idea is to apply a zero-gradient condition to pressure but reverse the gravity direction in the ghost cells when calculating the pressure gradient. This automatically incorporates both the hydrostatic and kinematic pressure throughout the whole cell and means that the gradient only needs to be calculated once.

5.6.2 Riemann problem benchmarks

To compare how the different Riemann solvers perform, five 1D Riemann problems were used as benchmark tests, as in Bassi et al. (2018). Table 5.2 shows the initial conditions, artificial compressibility parameter β, and the output pseudo-time τ for each Riemann problem. RP1 is the Elsworth and Toro (1992) test for the constant-density artificial-

Case	ρ_L	ρ_R	u_L	u_R	p_L	p_R	β	τ
1	1.0	1.0	1.0	1.0	0.1	1.0	0.81	0.1
2	1.0	1.0	0.0	0.0	1000.0	0.01	1000.0	0.005
3	1.4	1.0	0.0	0.0	1.0	1.0	1.0	2.0
4	1.4	1.0	0.1	0.1	1.0	1.0	1.0	2.0
5	0.001 25	1.0	1.0	1.0	0.1	1.0	1.0	0.005

Table 5.2: Initial conditions for Riemann problems, based on Bassi et al. (2018, Tables 3–4).

compressibility equations, designed to assess the robustness of the Riemann solvers, even though smooth velocity and pressure fields are generally expected in the artificial compressibility method. RP2–RP4 come from Toro (2009); RP2 assesses how the Riemann solvers deal with a large pressure jump, RP3 a stationary contact wave, and RP4 a slowly moving contact wave. RP5 is RP1 but with a large density jump, devised by Bassi et al. (2018) to test how the Riemann solvers might perform at a free surface. Since all five Riemann problems involve a contact wave bounded by two acoustic waves, the exact solution (5.66)–(5.71) can be used to check the results. Indeed, recall from Section 5.5.1 that, if $u_L = u_R = 0$, the value of β must satisfy (5.79) for the contact wave to be bounded by the contact waves. This, then, applies to Cases 2 and 3. Case 2 needs $\beta > 499.995$, and Case 3 needs $\beta > 0$, which lines up with the critical values given by Bassi et al. (2018, Table 4).

As the implementation is 2D, a mesh was created with 100 cells in the x-direction and one cell in the y-direction, with a zero-gradient boundary condition applied to all variables. There is no gravity term. For the pseudo-time stepping, which is explicit, three Runge-Kutta stages and a Courant number of 0.98 were used with global time stepping, while the real-time term was not activated. The minmod slope limiter was used for density, velocity, and pressure. Minmod was used for pressure because the new method is not accurate in pseudo-time, only at the incompressible limit, and the split method is identical to the unsplit method in the absence of gravity. The only difference between the simulations was the Riemann solver.

As shown in Figure 5.3, all three Riemann solvers perform almost identically for RP1–RP4. RP5 is more extreme, involving a large density and pressure discontinuity, and it reveals the difference between the Riemann solvers. The Roe Riemann solver performs the worst out of the three. The Bassi Riemann solver, whose performance corresponds with the graphs in the original study (Bassi et al., 2018, Figure 6), is not the best performing either. This is surprising as the numerical flux should be exact when the contact wave is bounded by the acoustic waves which, indeed, was never violated in this test. The Osher Riemann solver performs the best. The results suggest that the Roe Riemann solver should be avoided and, as the validity of the assumptions underpinning the Bassi Riemann solver cannot be guaranteed, the Osher Riemann solver provides a robust alternative.

5.6.3 Hydrostatic pressure benchmarks

To assess the well-balanced properties of the different pressure gradient calculations, a simple 1D hydrostatic problem was used as a benchmark test, as in Qian et al. (2006). The test involves a column of air ($\rho = 1$) overlaying a column of water ($\rho = 1000$) under the influence of gravity. The initial conditions are $\mathbf{u} = 0$ and $p = 0$ everywhere, and the density and velocity values should not change but the pressure should settle to hydrostatic.

As the implementation is 2D, a mesh was created with 20 cells in the x-direction and one cell in the y-direction, with all four boundaries modelled as walls, and gravity of $\mathbf{g} = (9.81, 0)$. For pseudo-time, which is explicit, three Runge-Kutta stages and a Courant number of 0.98 were used with local time stepping. The simulation was run for one real-time step of size $\Delta t = 0.05$ until the residuals for ρu were less than 1×10^{-5}. The Osher Riemann solver was used with the minmod limiter for density and velocity in all simulations. The value of the artificial compressibility coefficient was set to $\beta = 2000$—a smaller value led to instability in the split simulations, a problem acknowledged by Kelecy and Pletcher

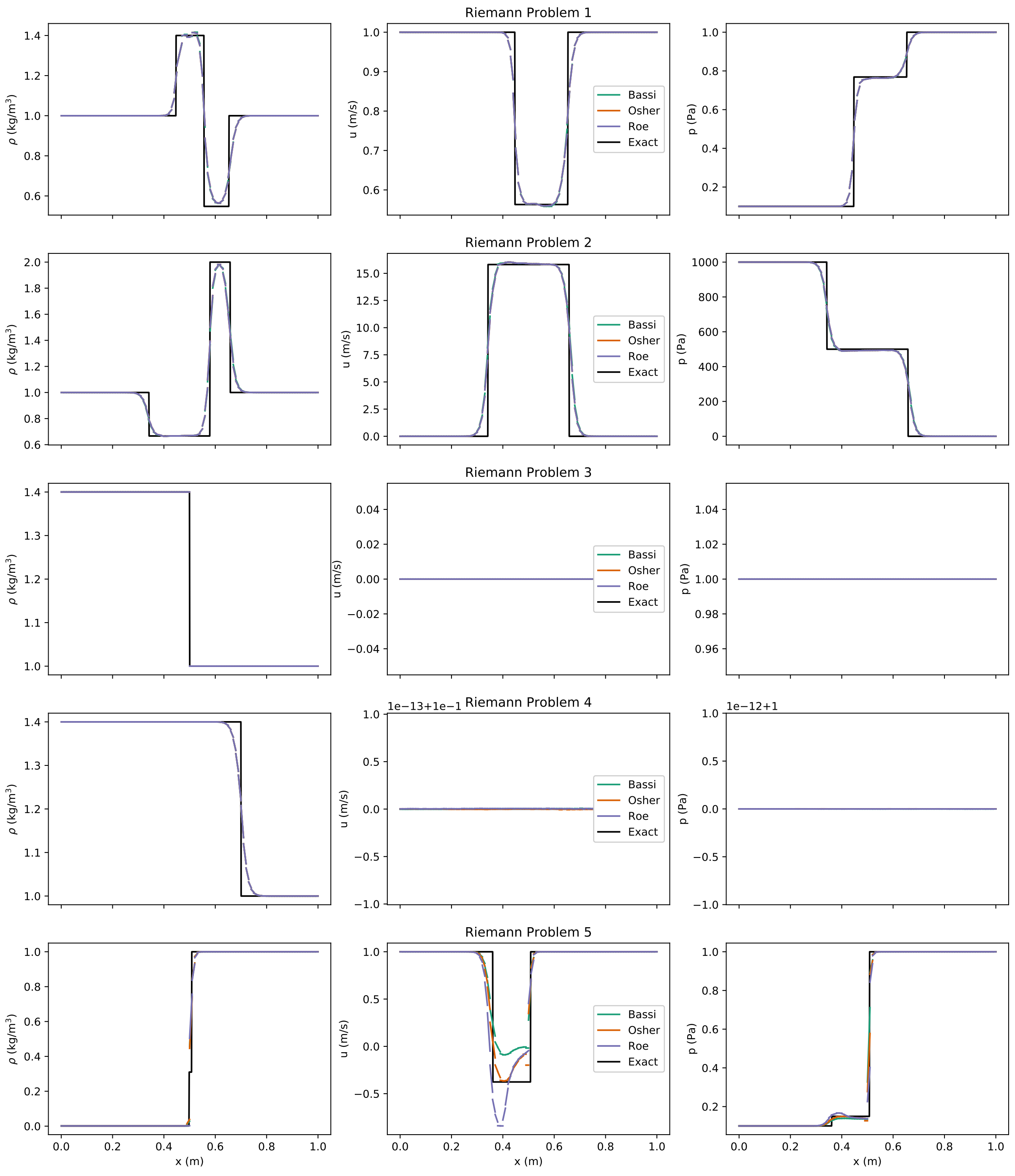

Figure 5.3: Comparison of Riemann solvers in 1D Riemann problems from Bassi et al. (2018).

(1997). The only difference between the simulations was the pressure gradient calculation.

Figure 5.4 shows that limiters with no special treatment do not converge to the correct solution. The anomalies near the boundaries can be attributed to the reversed gravity in the ghost cells not having any effect for standard limiters, as discussed in Section 5.6.1. However, the spurious velocities near the free surface are precisely due to standard limiters including information from the other side of the free surface, as discussed in Section 5.4.3. In both cases, incorrect pressure gradients lead to incorrect densities or velocities, which is important to remember for the next benchmark test.

In contrast, the two split limiters and the new incompressible limit method converge to the correct solution. This demonstrates that special treatment is required to ensure a balance with the gravity source term, thereby eradicating spurious densities and velocities.

5.6.4 Dam break benchmarks

In dam breaks, a column of water is initially at rest behind an imaginary, infinitesimally narrow dam. The dam instantaneously disappears, and the column of water is free to flow. Dam break simulations are useful for testing the robustness of a free-surface method under extreme conditions, and their widespread use means that experimental data is available for validation.

In this study, we simulated a dam break in a small domain to examine pressure gradient calculations in detail, and then a much larger domain to test the method against the full range of the Martin and Moyce (1952) experimental data. In both cases, the initial conditions involved a high density ratio, $\rho = 1$ for air and $\rho = 1000$ for water, with $\mathbf{u} = 0$ everywhere and, since the initial condition for pressure is somewhat arbitrary (Kelecy and Pletcher, 1997), p is set to zero everywhere initially. All four boundaries were modelled as walls, but this should not affect comparisons with the experimental data as water is much denser than the air (Bhat and Mandal, 2019a; Kelecy and Pletcher, 1997). Gravity was set to be $\mathbf{g} = (0, -9.81)$.

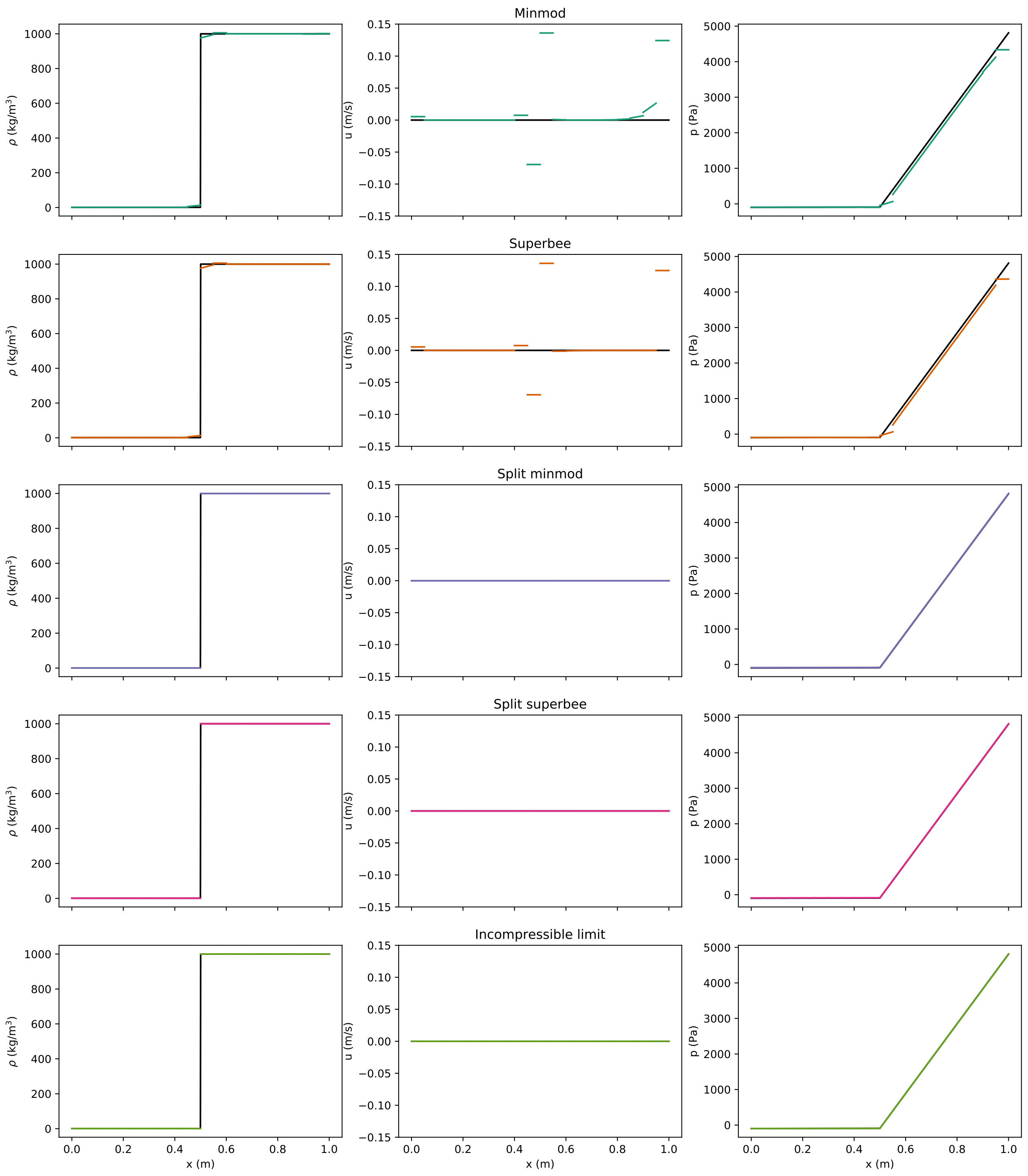

Figure 5.4: Comparison of pressure gradient calculations in 1D hydrostatic test case.

Small domain (non-hydrostatic pressure)

The previous benchmark showed that split limiters are well-balanced under hydrostatic pressure, unlike standard limiters. However, recall from Section 5.4.3 that pressure gradient discontinuities occur even in the absence of gravity. While split limiters do not address this problem, the new incompressible limit method deals with it automatically. This was explored with the first real-time step of a dam break.

A mesh was created with 20 cells in both directions. The Osher Riemann solver was used with the minmod limiter for density and velocity in all simulations. For pseudo-time, which is explicit, three Runge-Kutta stages and a Courant number of 0.48 were used with local time stepping. The simulations were run for one real-time step of size $\Delta t = 0.01$ until the residuals for ρu and ρv were smaller than 1×10^{-3}. The value of the artificial compressibility coefficient was set to $\beta = 1100$ as smaller values led to instability. The only difference between the simulations was the pressure gradient calculation.

Figure 5.5 shows the 2D fields in the first real-time step, and Figure 5.6 a cross-section along the lowest row of cells. As gravity does not act in the x-direction, the split and unsplit slope limiter calculations are the same in this direction. However, the split results are shown here because the unsplit results will be impacted by gravity in the y-direction, as illustrated in Figure 5.4 from the previous benchmark.

Consider Figure 5.6 and the pressure gradient just to the left of the density discontinuity at $x = 0.4$. The split limiters underestimate its magnitude, especially minmod. This is because information is taken from both sides of the free surface, but there should be a discontinuity defined by the dynamic boundary condition $[\nabla p/\rho] = 0$. The new incompressible limit method performs better than the split limiters. In fact, it performs surprisingly well considering that it is not based on the value of pressure, only the other variables, TVD gradients, and source terms.

Now recall from the hydrostatic benchmark test that, the better the pressure gradient calculation at capturing the pressure gradient discontinuity, the smaller the errors in the density and velocity. Consider Figure 5.6 and the density just to the left of the density discontinuity at $x = 0.4$. All three simulations seem to have converged to a solution with

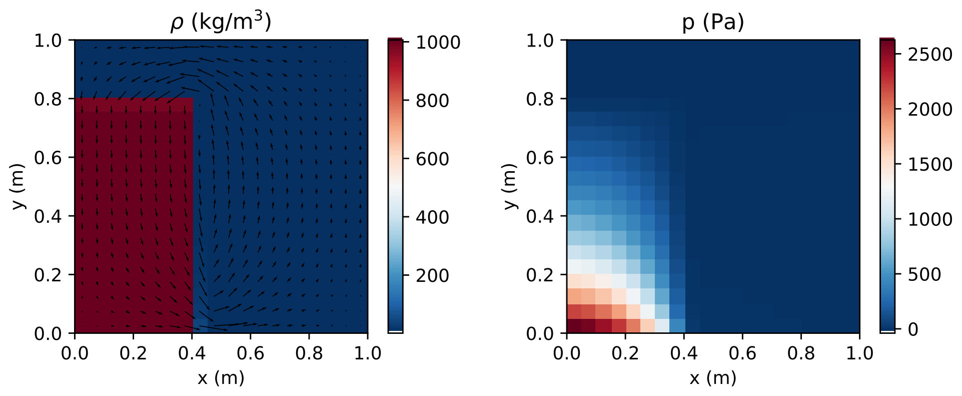

Figure 5.5: First real-time step of dam break (small domain) with new incompressible limit pressure gradient calculation.

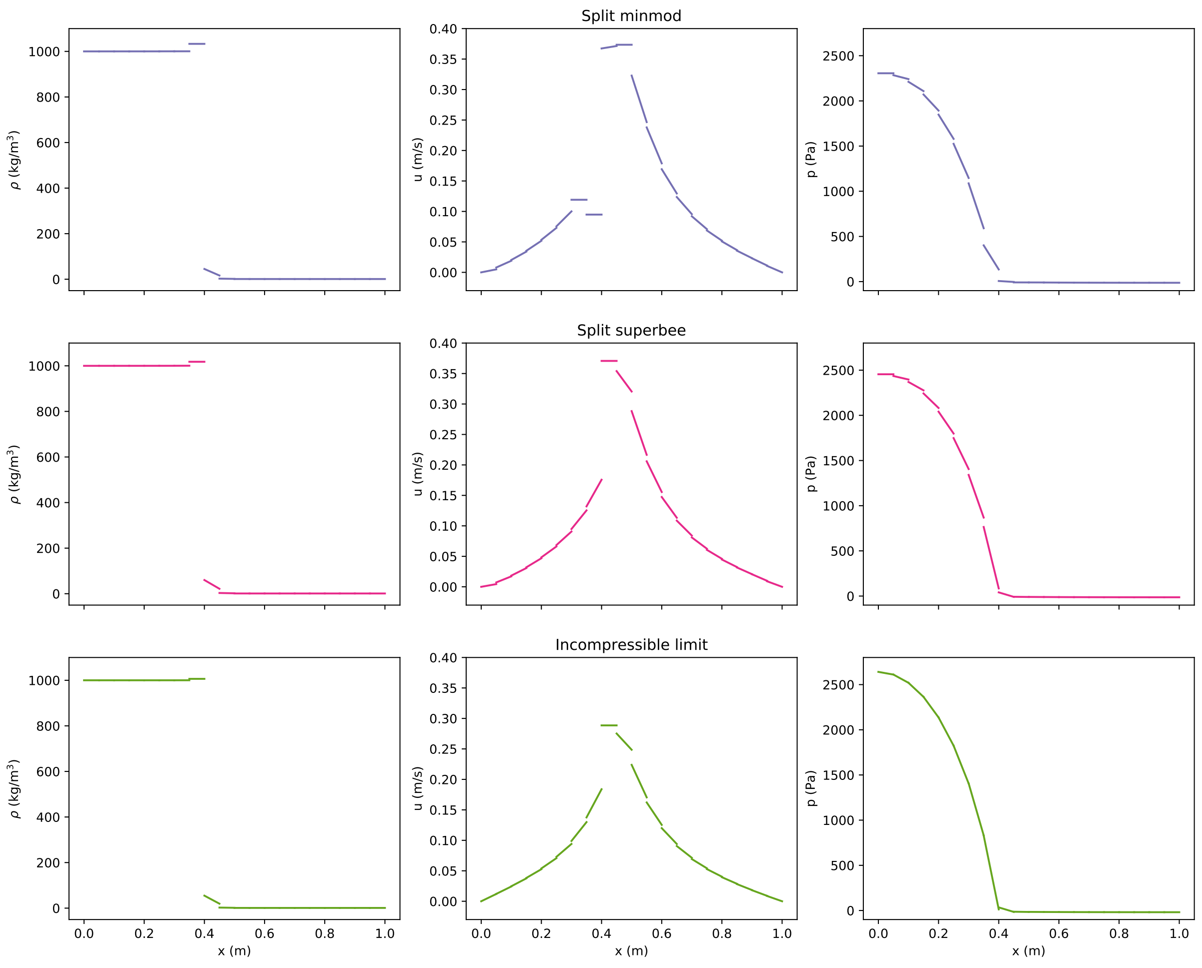

Figure 5.6: Comparison of pressure gradient calculations in first real-time step of dam break (small domain), along the lowest cross-section parallel to the x-axis, which is not the direction of gravity.

varying degrees of compression. While this compression resembles Gibbs-type oscillations in the density field, Figure 5.6 clearly shows that the minmod density limiter has fully limited the gradient to zero, and so the density limiter is not causing the problem. As the only difference between the simulations is the pressure gradient calculation, we must conclude that this is what is causing the problem. The better the method at capturing the pressure gradient discontinuity, the smaller the error. The standard limiters, due to their inability to capture the pressure gradient discontinuity, cause the largest errors. Even though the correct solution for the velocity field is unknown, we might similarly conclude that the pressure gradient is also causing the differences in the velocity values.

Large domain (Martin and Moyce, 1952)

So far, the benchmark tests have not gone further than one real-time step. In this benchmark, we recreated the Martin and Moyce (1952) dam break experiments for square columns with dimension $a = 2\frac{1}{4}$ inches $= 0.05715$ metres and $a = 4\frac{1}{2}$ inches $= 0.1143$ metres. Kelecy and Pletcher (1997) also used these data for the same equations, and Bhat and Mandal (2019a) used them for artificial compressibility coupled with VOF. However, both studies had a domain too small to cover the full temporal range of data, with the Kelecy and Pletcher (1997) domain having width $5a$ and height $1.25a$. Here, we used a domain of width $15a$ and height $1.25a$, divided into 240 cells in the x-direction and 20 cells in the y-direction. This allowed the surge to propagate unimpeded until the end of the experiment.

The other parameters included two pseudo-time Runge-Kutta stages, an artificial compressibility coefficient of $\beta = 1100$, and convergence tolerances of 0.01 for ρ, ρu, ρv, and p. Both the pseudo-time and real-time Courant numbers were set to 0.45. The real-time Courant number was set this low not for stability (the real-time derivative is treated point-implicitly) but because it led to better convergence in pseudo-time. In all cases, the minmod slope limiter was used for density and velocity, but the Riemann solver and pressure gradient calculations were changed. This was to determine if the variability experienced in the previous benchmarks is important in long simulations.

Figure 5.7 shows real-time snapshots of one simulation. They are similar to the results

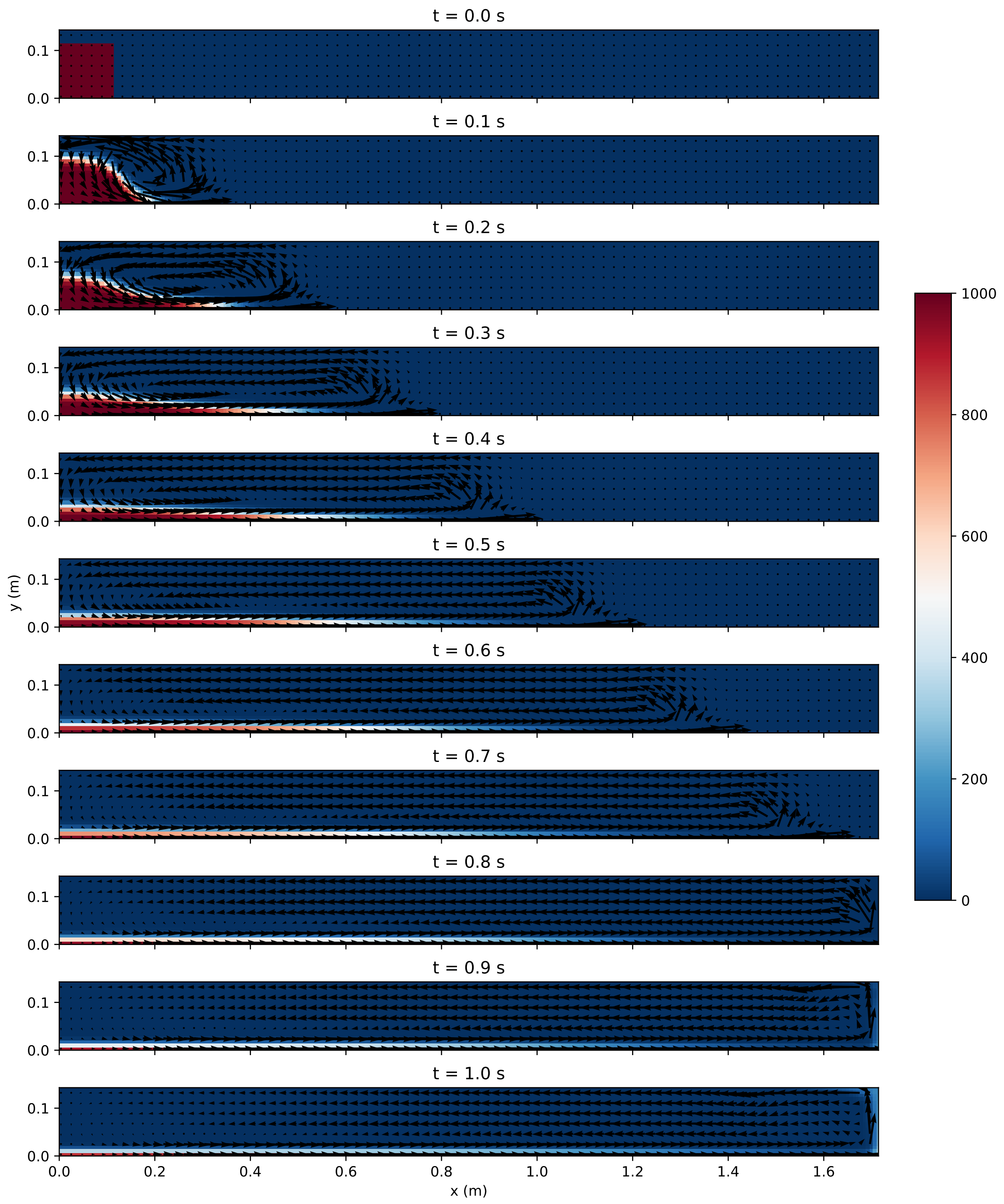

Figure 5.7: Density in the dam break (large domain) with length scale $a = 4\frac{1}{2}$ inches, the Bassi Riemann solver, and the incompressible limit pressure gradient calculation. Velocity shown once every nine cells for clarity.

in Bhat and Mandal (2019a) and Kelecy and Pletcher (1997). In particular, the difference between the density of the water and the density of the air induces a large horizontal pressure gradient, which causes a surge front to propagate out from the column of water, and a vortex to appear around the free surface. These snapshots explicitly show the density field, rather than a contour to represent the free surface as in Bhat and Mandal (2019a) and Kelecy and Pletcher (1997), meaning that the numerical diffusion is not hidden. Indeed, the transition between water and air straddles several cells, and there is much diffusion towards the end of the simulation when the surge front hits the far wall. This diffusion could be addressed with the slope modification method (Pan and Chang, 2000) or an interface compression term (Bhat and Mandal, 2019b), which have both been implemented for artificial compressibility coupled with VOF previously.

Figure 5.8 shows the post-processed results, where the column height and surge front position have been normalised by dividing by a, and the time multiplied by $\sqrt{g/a}$. This allows the two experiments to be compared even though they are at different scales. The Riemann solver and pressure gradient calculation do not have much of an effect on the results, especially for the $a = 4\frac{1}{2}$ inches case, and the simulated values are close to the measured values. There are some differences between the simulations and observations, particularly in the surge front position towards the end of the $a = 4\frac{1}{2}$ inches case, but it is impossible to know if these are due to the model or due to the data, which were collected 70 years ago. In any case, the results show that the method performs well under this extreme scenario, and the general outputs are not very sensitive to the Riemann solver or pressure gradient calculation. It is interesting to note that at no point was the wave ordering assumption behind the Bassi Riemann solver violated, which suggests it may also be acceptable for general use alongside the Osher Riemann solver. However, further investigations are needed to find out when the wave ordering assumption is broken, as acknowledged by Bassi et al. (2018, Appendix B4).

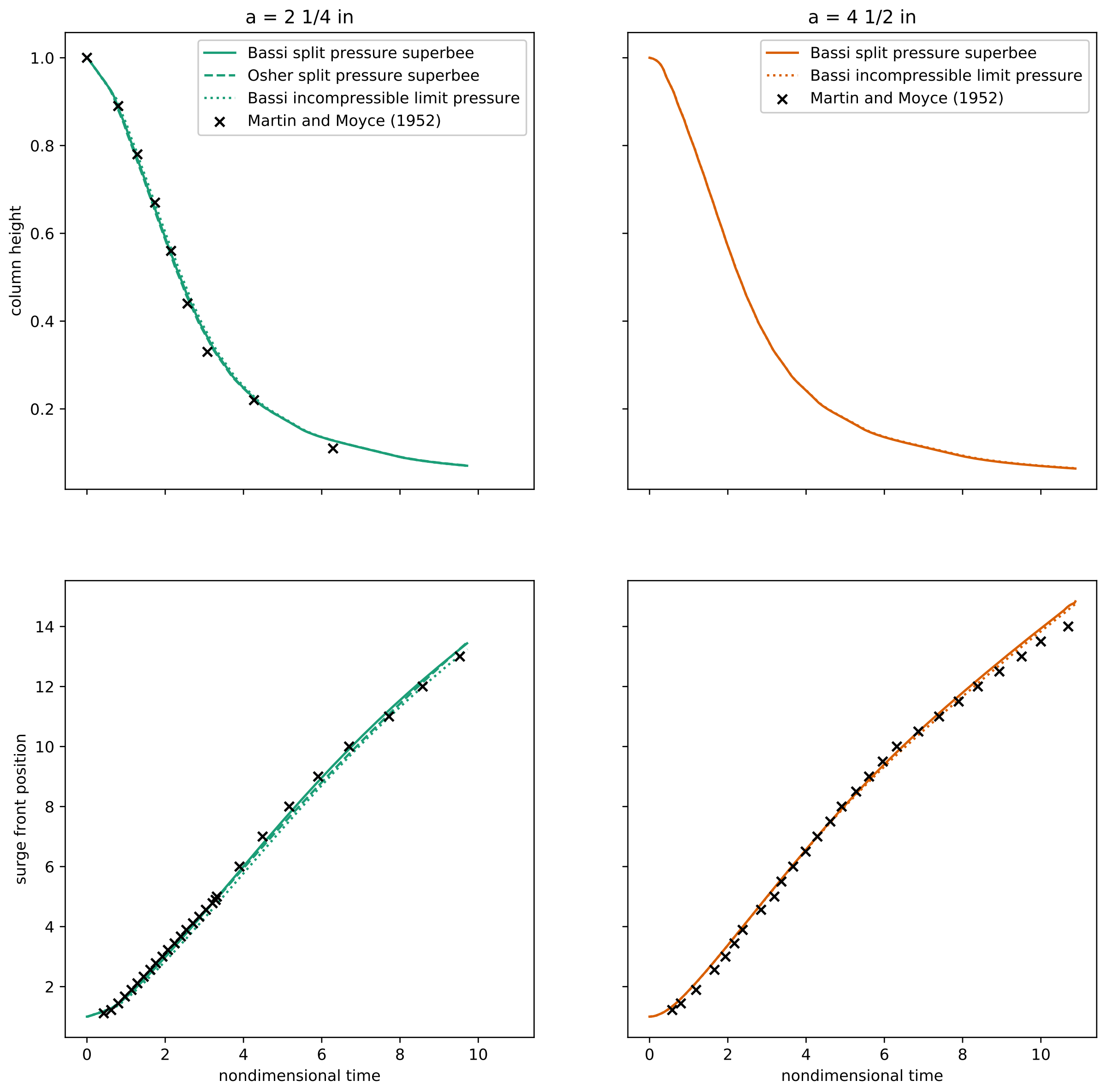

Figure 5.8: Comparison of dam break (large domain) simulations with experimental data from Martin and Moyce (1952), where a is the square column's initial width and height. Note that all values are nondimensionalised and there are no experimental data for column height for $a = 4\frac{1}{2}$ in.

5.7 Conclusion

Only a few researchers (Kelecy and Pletcher, 1997; Qian et al., 2006; Bassi et al., 2018) have investigated the variable-density artificial-compressibility equations previously, with their developments spaced out over several decades. This chapter presents a Godunov-type scheme, taking advantage of recent advances in the wider field by applying a more robust Riemann solver (Dumbser and Toro, 2011) and a more easily parallelisable time discretisation (Loppi, 2019; Vermeire et al., 2019) to these equations for the first time. We also developed a new method for calculating the pressure gradient in the MUSCL reconstruction step. By using the information already available from the TVD gradients and source terms, the new method automatically captures the pressure gradient discontinuity at the free surface. This pressure gradient method is applicable to any scheme with dual time stepping.

Benchmark tests presented in this chapter demonstrate the capabilities of the Godunov-type scheme. First, a battery of five Riemann problems showed that it can capture discontinuities in the flow, and that the Osher Riemann solver provides a robust alternative to the Bassi Riemann solver. The Osher Riemann solver may be needed as the Bassi Riemann solver is only exact for certain wave configurations. Second, the new pressure gradient method has been shown to be well-balanced under hydrostatic cases, like split limiters. Third, the new pressure gradient method has been shown to capture non-hydrostatic density discontinuities, unlike split limiters. This suggests that the new pressure gradient method is preferable to existing options. Finally, the Martin and Moyce (1952) dam break experiments were successfully recreated. In particular, while Kelecy and Pletcher (1997) only used a temporal subset of this experimental data, here an extended domain allows the full propagation of the surge front to be assessed. This shows that the method is suitable for practical applications.

However, practical simulations do not generally use 2D Cartesian meshes (see, for instance, the simulations of leaky barriers in Chapter 4). Consequently, the solver must be implemented on 3D unstructured meshes in parallel for practical applications, one of the focuses of Chapter 6.

Chapter 6. Convergence on 3D unstructured meshes

This chapter reproduces research published in Leakey et al. (2022a), generalising the specific variable-density artificial-compressibility scheme set out in Chapter 5 to 3D unstructured meshes. Previous to this, numerical methods to solve the variable-density artificial-compressibility equations have not been implemented on 3D unstructured meshes. They have been implemented on 2D structured meshes (Qian et al., 2006; Bassi et al., 2018; Massa et al., 2020; Shin et al., 2012; Evtushok et al., 2021), 3D structured meshes (Kelecy and Pletcher, 1997; Shapiro and Drikakis, 2005b; Hu et al., 2013; Manzanero et al., 2020), and 2D unstructured meshes (Wang et al., 2020). While most of these papers use the Cartesian cut-cell method to model simple obstacles, an implementation on 3D unstructured meshes is needed for more complex geometries, for instance, the leaky barriers modelled in Chapter 4. This is lacking in the current literature, although other multiphase artificial compressibility methods have been implemented on unstructured meshes, both 2D (Zhao et al., 2002; Sambe et al., 2011; Bhat and Mandal, 2019a; Ntouras and Papadakis, 2020) and 3D (Hino, 1999; Evstigneev, 2008; Lv et al., 2008; Oxtoby et al., 2015; Economon, 2020), as well as structured meshes (Nourgaliev et al., 2004; Bhat and Mandal, 2019b; Pan and Chang, 2000; Kajzer and Pozorski, 2020). There is thus significant potential for further developing multiphase artificial compressibility methods on unstructured meshes, especially for investigating the limiter convergence problem only briefly hinted at by Economon (2020).

Capitalising on the powerful tools provided by OpenFOAM, we generalise the specific variable-density artificial-compressibility scheme in Chapter 5 to 3D unstructured meshes in parallel, testing the solver against dam-break and hydrostatic pressure benchmarks. As outlined in Section 4.1.3, most OpenFOAM solvers use the PIMPLE algorithm to update the pressure field. Since the new solver is based on artificial compressibility and not

PIMPLE, it does not need much of the numerics built into in OpenFOAM. Nonetheless, it is useful to develop the solver in OpenFOAM because the low-level structure does not have to be built from scratch. In particular, here we rely on the existing infrastructure for input/output, unstructured 3D meshes, scalar fields, and vector fields, as well as some gradient schemes. A downside is that the form of parallelisation in OpenFOAM, domain decomposition, is not optimal for artificial compressibility as the waves propagate locally (Hodges, 2020). A more efficient implementation could be on GPUs, as in Loppi (2019) and Kajzer and Pozorski (2020), and use a similar structure to the object-oriented Python prototype as in Section 5.6. However, we focus on the numerics in this chapter, leaving the implementation of efficient parallelisation to future work.

6.1 Implementation

This chapter goes further than Chapter 5, implementing the numerical scheme (5.30)–(5.32) in OpenFOAM so that it can be used on 3D unstructured meshes and in parallel. The new implementation uses the Roe Riemann solver and the new pressure gradient calculation developed in Chapter 5. While ghost cells could be used for the Python implementation in Chapter 5, OpenFOAM does not allow this, and so the fluxes have to be set explicitly at the boundaries, which is straightforward for walls anyway. The solver has a structure inspired by foam-extend's `dbnsFoam` (Jasak, 2014). However, it was written from scratch to take into account the different governing equations and the fact that several cell-limited gradient schemes were added to OpenFOAM since `dbnsFoam` was released. Therefore, it has the potential to be forward-compatible with any limiters added to OpenFOAM in the future.

Switching to an unstructured mesh does require slightly changing the scheme from Chapter 5. The difference is in the MUSCL reconstruction step, which is where the primitive variables are extrapolated to cell faces to provide the left and right states for the Riemann solver. This extrapolation relies upon a limited gradient to avoid the introduction of new extrema, Gibbs-type oscillations. Unstructured meshes require a different approach than structured meshes to do MUSCL reconstruction, and there are two components to this difference: calculating the gradient and limiting the gradient.

First, the gradient calculation for Cartesian meshes is very simple, but unstructured meshes require a method such as Green-Gauss or least-squares. These well-known standard methods are already implemented in OpenFOAM, so using them is simply a case of writing `fvc::grad()` in the code and specifying `Gauss` or `leastSquares` in `fvSchemes`. The gradients can be limited by specifying a limiter alongside `Gauss` or `leastSquares` in `fvSchemes`. Note that this use of `fvc::grad()` is the only time we use OpenFOAM numerics here, but it is not the gradient calculation itself which is problematic.

The problem lies in the limiters, a problem that did not exist in the implementation in Chapter 5. The difference is that, while the limiter calculation for structured meshes is straightforward, involving constructing left- and right-sided gradients at each cell and applying a standard limiter to these, it is not obvious what the left- and right-sided gradients are on an unstructured mesh (Darwish and Moukalled, 2003), as outlined in Section 5.4.2. The standard way to overcome this problem is the framework of Barth and Jespersen (1989), which is the method implemented in the cell-limited gradient schemes in OpenFOAM.

6.1.1 Barth and Jespersen (1989)

Let W be a scalar variable or one of the components of a vector variable. In the context of a Godunov-type scheme, the idea is to construct a limiter function $\Phi \in [0, 1]$ to be used as

$$\mathbf{W}_{f,i} = \mathbf{W}_i + \Phi_i \nabla \mathbf{W}_i \cdot (\mathbf{x}_f - \mathbf{x}_i) \tag{6.1}$$

where ∇W_i is gradient of the variable before limiting and $W_{f,i}$ is the Riemann state at face f on the side belonging to cell i. Then the Riemann states either side of the face can be put into the Riemann solver to get the numerical flux. Barth and Jespersen (1989) introduce a method that ensures the interpolated variable $W_{f,i}$ does not exceed that of the neighbouring cells in magnitude. Note that this means all the neighbouring cells of cell i, not just the cell on the other side of face f.

Practically, the calculation loops over all the cell interfaces twice (Blazek, 2005). There is a loop to calculate the maximum and minimum values in the neighbouring cells,

$$W_{i,\max} = \max\left(W_i, \max_{j\in\text{nei}} W_j\right) \tag{6.2}$$

$$W_{i,\min} = \min\left(W_i, \min_{j\in\text{nei}} W_j\right), \tag{6.3}$$

and then a loop to determine how much limiting is required,

$$\Phi_i = \min_f \begin{cases} \phi\left(\frac{\Delta_{i,\max}}{\Delta_{i,f}}\right) & \text{if } \Delta_{i,f} > 0 \\ \phi\left(\frac{\Delta_{i,\min}}{\Delta_{i,f}}\right) & \text{if } \Delta_{i,f} < 0 \\ 1 & \text{if } \Delta_{i,f} = 0 \end{cases} \tag{6.4}$$

where

$$\phi(y) = \min(1, y) \tag{6.5}$$

and

$$\Delta_{i,\max} = W_{i,\max} - W_i \tag{6.6}$$

$$\Delta_{i,\min} = W_{i,\min} - W_i \tag{6.7}$$

$$\Delta_{i,f} = \nabla W_i \cdot \mathbf{d}_{i,f} \tag{6.8}$$

$$\mathbf{d}_{i,f} = \mathbf{x}_f - \mathbf{x}_i. \tag{6.9}$$

Thus, the quantity $\Delta_{i,f}$ is a measure of how much W changes between the centre of cell i and the centre of the cell interface f. The function ϕ clips $\Delta_{i,f}$ so that it does not exceed $\Delta_{i,\min}$ or $\Delta_{i,\max}$ in magnitude.

In the uniform 1D case, the Barth-Jespersen limiter can be recast into the Spekreijse (1987) form

$$W^L_{i+1/2} = W_i + \frac{1}{2}\psi(r_i)(W_i - W_{i-1}) \tag{6.10}$$

where the limiter is

$$\psi(r) = \frac{1}{2}(r+1)\min\left(\phi\left(\frac{4r}{r+1}\right), \phi\left(\frac{4}{r+1}\right)\right) \tag{6.11}$$

and it is applied to the ratio

$$r = \frac{W_{i+1} - W_i}{W_i - W_{i-1}}, \tag{6.12}$$

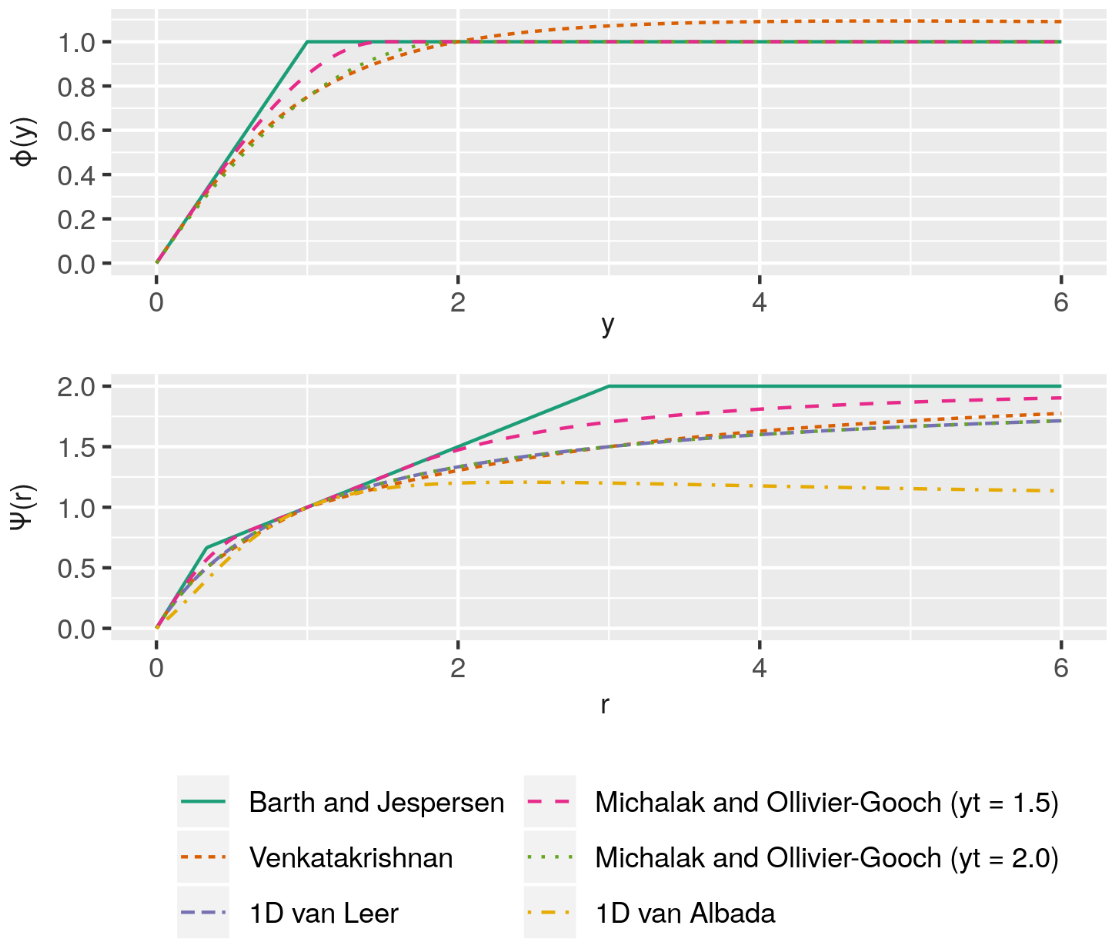

Figure 6.1: Comparison of ϕ (used in unstructured limiter Φ) and ψ (used in Spekreijse's 1D formulation) between different limiters.

as shown in Venkatakrishnan (1995). Both ϕ from (6.5) and ψ from (6.11) are plotted in Figure 6.1.

The practical problem with this method is that the non-differentiability of (6.5) can inhibit convergence in steady-state cases, instead leading to bounded odd-even modes. The simulation is not unstable but neither does it converge. It would continue indefinitely if it was not stopped. This is a well-known problem (Economon, 2020; Blazek, 2005; Venkatakrishnan, 1995; Michalak and Ollivier-Gooch, 2008; Aftosmis et al., 1995; Anderson et al., 1986; Berger et al., 2005; Li et al., 2020; Park and Kim, 2012; Park et al., 2010). Fortunately, there are ways to overcome it, including replacing the non-differentiable function (6.5) with a differentiable alternative and switching off the limiter in areas of relatively uniform flow. Here, we investigate these techniques in the context of artificial compressibility, where pseudo-time convergence is required not just once, but at every real-time step, and thus the limiters must converge more reliably than for a simple steady-state simulation.

6.1.2 Venkatakrishnan (1995)

Venkatakrishnan (1995) addresses the convergence problem by replacing the non-differentiable function (6.5) with the differentiable function

$$\phi(y) = \frac{y^2 + 2y}{y^2 + y + 2}. \tag{6.13}$$

As shown in Figure 6.1, this function is much smoother than (6.5), and the fact that $\phi > 1$ for $y > 2$ does not affect ψ, which is bounded by the Barth-Jespersen limiter for all r.

A slight modification, not plotted in Figure 6.1, ensures that the limiter is also not activated in smooth regions:

$$\Phi_i = \min_f \begin{cases} \frac{1}{\Delta_{i,f}} \left(\frac{(\Delta_{i,\max}^2 + \epsilon_i^2)\Delta_{i,f} + 2\Delta_{i,f}^2 \Delta_{i,\max}}{\Delta_{i,\max}^2 + 2\Delta_{i,f}^2 + \Delta_{i,\max}\Delta_{i,f} + \epsilon_i^2} \right) & \text{if } \Delta_{i,f} > 0 \\ \frac{1}{\Delta_{i,f}} \left(\frac{(\Delta_{i,\min}^2 + \epsilon_i^2)\Delta_{i,f} + 2\Delta_{i,f}^2 \Delta_{i,\min}}{\Delta_{i,\min}^2 + 2\Delta_{i,f}^2 + \Delta_{i,\min}\Delta_{i,f} + \epsilon_i^2} \right) & \text{if } \Delta_{i,f} < 0 \\ 1 & \text{if } \Delta_{i,f} = 0 \end{cases} \tag{6.14}$$

where

$$\epsilon_i^2 = K^3 V_i \tag{6.15}$$

and V_i is the cell volume (Blazek, 2005). The parameter K is a threshold that marks the largest size of oscillations untouched by the limiter.

If the gradient is small, then $\epsilon_i^2 \gg \Delta_{i,\min}^2, \Delta_{i,\max}^2, \Delta_{i,\min}\Delta_{i,\max}$ and so $\Phi_i \to 1$, which means there is no limiting. This stops residuals stalling due to numerical noise. By a similar argument, increasing K increases Φ_i and so decreases the amount of limiting, and while this is more conducive to convergence to a steady-state, it also increases the potential for Gibbs-type oscillations and thus instability. The more convergent options are more unstable, that is, there is a trade-off between convergence and stability.

6.1.3 Michalak and Ollivier-Gooch (2008)

While Barth-Jespersen and Venkatakrishnan are the two standard limiters (Blazek, 2005), there are more options in the literature. For example, Michalak and Ollivier-Gooch (2008) replaced the function (6.5) with

$$\phi(y) = \begin{cases} P(y) & \text{if } y < y_t \\ 1 & \text{if } y \geq y_t \end{cases} \tag{6.16}$$

where $P(y)$ is the cubic polynomial with

$$P\Big|_0 = 0 \tag{6.17}$$

$$P\Big|_{y_t} = 1 \tag{6.18}$$

$$\frac{dP}{dy}\Big|_0 = 1 \tag{6.19}$$

$$\frac{dP}{dy}\Big|_{y_t} = 0 \tag{6.20}$$

and $1 \leq y_t \leq 2$ is a threshold. The polynomial itself is not explicitly stated in Michalak and Ollivier-Gooch (2008), but a simple derivation gives

$$P(y) = ay^3 + by^2 + y \tag{6.21}$$

$$a = \frac{1}{y_t^2} - \frac{2}{y_t^3} \tag{6.22}$$

$$b = -\frac{3}{2}ay_t - \frac{1}{2y_t}. \tag{6.23}$$

As shown in Figure 6.1, the function ϕ is differentiable everywhere and does not exceed 1.

Like Venkatakrishnan, Michalak and Ollivier-Gooch also propose switching off the limiter in uniform regions of flow. However, they use a different method for this, smoothly transitioning to switching off the limiter when

$$(\Delta_{i,\max} - \Delta_{i,\min})^2 < K^3 V_i. \tag{6.24}$$

This is achieved by defining

$$\tilde{\phi}_i = \sigma_i + (1 - \sigma_i)\phi \tag{6.25}$$

with ϕ from (6.16) and $\tilde{\phi}_i$ now plugged into (6.4). The indicator-type function is given by

$$\sigma_i = \begin{cases} 1 & \text{if } (\Delta_{i,\max} - \Delta_{i,\min})^2 \le K^3 V_i \\ s\left(\frac{(\Delta_{i,\max}-\Delta_{i,\min})^2 - K^3 V_i}{K^3 V_i}\right) & \text{if } K^3 V_i < (\Delta_{i,\max} - \Delta_{i,\min})^2 < 2K^3 V_i \\ 0 & \text{if } (\Delta_{i,\max} - \Delta_{i,\min})^2 \ge 2K^3 V_i \end{cases} \tag{6.26}$$

with the smooth transition function s given by

$$s(y) = 2y^3 - 3y^2 + 1. \tag{6.27}$$

6.1.4 Limiters in OpenFOAM

The above three limiters are already implemented in OpenFOAM, and can be easily accessed by specifying `cellLimited`, `cellLimited<Venkatakrishnan>`, or `cellLimited<cubic>` respectively alongside the gradient calculation in `fvSchemes`. However, there are major problems with the limiters as implemented in the OpenFOAM source code.

First, `cellLimited<Venkatakrishnan>` is not the same as the limiter put forward by Venkatakrishnan (1995), and so cannot be expected to have the same convergence properties shown in the original study. One difference is that it uses the unmodified version (6.13) instead of (6.14), and so may still be active in uniform regions of flow. Another difference is that, regardless of the limiter, OpenFOAM always clips Φ so that it never exceeds 1. As indeed noted in the source code documentation, this clipping makes this particular limiter non-differentiable, and so it "no longer conforms to the basic principles of this kind of limiter function" (`VenkatakrishnanGradientLimiter.H`, lines 53–54).

The whole reason Venkatakrishnan developed the limiter was so it could be differentiable, therefore it seems that `cellLimited<Venkatakrishnan>` is of little practical use.

Second, `cellLimited<cubic>` is also not the same as the limiter put forward by Michalak and Ollivier-Gooch (2008), and so cannot be expected to have the same convergence properties shown in the original study either. One difference is that it does not use (6.25), and therefore does not stop activation in uniform regions of flow. Another difference is that the limiter uses an incorrect cubic polynomial, one with a very large slope discontinuity at $\phi(y_t)$, not the one stated in (6.21)–(6.23).[1]

In this chapter, we do not attempt to fix `cellLimited<Venkatakrishnan>`; the automatic clipping of Φ makes this too problematic. However, we do implement an improved version of `cellLimited<cubic>` that corresponds to the limiter in Michalak and Ollivier-Gooch (2008), both by using the correct cubic polynomial and by stopping activation in uniform regions of flow. This improved version is called `cellLimited<Michalak>`.[2]

Note that there are three other limited gradient schemes in OpenFOAM: `cellMDLimited`, `faceLimited`, and `faceMDLimited`. In the `cellLimited` limiters outlined above, one scalar limiter is applied to the x, y, and z components of the gradient equally, irrespective of which neighbouring cells have the minimum and maximum values. This can lead to excessive limiting. The `cellMDLimited` limiter takes an alternative approach. Consider a cell with faces $f = 1, 2, ..., F$. Set $(\nabla W)_{i,0} = (\nabla W)_i$ to be the gradient before limiting. Loop through the faces f, calculating the extrapolate

$$\Delta_{i,f} = (\nabla W)_{i,f-1} \cdot \mathbf{d}_{i,f} \tag{6.28}$$

[1]As of June 2021, the cubic polynomial has been fixed in the official development branches due to the work carried out in this thesis. See the bug reports at https://develop.openfoam.com/Development/openfoam/-/issues/2113 [Accessed: 14 June 2021] and https://bugs.openfoam.org/view.php?id=3684 [Accessed: 28 June 2021] for details. However, these bug fixes do not stop activation in uniform regions of flow.

[2]Thus `cellLimited<Michalak>` with $K = 0$ is equivalent to `cellLimited<cubic>` in the official development branches from June 2021.

and then clipping the gradient in the direction from the cell centre to that face centre:

$$(\nabla W)_{i,f} = \begin{cases} (\nabla W)_{i,f-1} + \mathbf{d}_{i,f} \frac{\Delta_{\text{i,max}} - \Delta_{i,f}}{|\text{d}_{i,f}|^2} & \text{if } \Delta_{i,f} > \Delta_{\text{i,max}} \\ (\nabla W)_{i,f-1} + \mathbf{d}_{i,f} \frac{\Delta_{\text{i,min}} - \Delta_{i,f}}{|\text{d}_{i,f}|^2} & \text{if } \Delta_{i,f} < \Delta_{\text{i,min}} \\ (\nabla W)_{i,f-1} & \text{otherwise} \end{cases} \tag{6.29}$$

for $f = 1, 2, ..., F$. The final iteration gives us the `cellMDLimited` gradient. Therefore, the limiter "is applied to the gradient in each face direction separately" (`cellMDLimitedGrad.H`, lines 36–38) not, as is sometimes suggested, to each coordinate direction separately. Meanwhile, the `faceLimited` and `faceMDLimited` limiters are like the `cellLimited` and `cellMDLimited` limiters but clip the gradient between the face-neighbour values rather than the cell-neighbour values. None of these allow differentiable functions as a run-time selectable option like `cellLimited` does. However, simulations using them are included for comparison purposes in the following section, alongside `cellLimited`, `cell-Limited<Venkatakrishnan>`, `cellLimited<cubic>`, and the newly implemented `cell-Limited<Michalak>`.

6.2 Application

The new solver was put through four benchmark tests: a simple 2D dam break to illustrate the importance of limiter choice, a more complicated 2D dam break to compare the new solver against `interFoam`, a 3D dam break to compare the new solver against `interFoam` in 3D, and a hydrostatic case to demonstrate that the solver can be run on arbitrary 3D unstructured meshes.

6.2.1 Convergence of limiters

First, we replicate from Section 5.6.4 the first real-time step of a dam break on a uniform 2D Cartesian mesh. The mesh is a metre length in each direction, divided into 20×20 cells. Due to its simplicity, the mesh is useful for isolating the effect of different limiters on convergence. Initial conditions are $\rho = 1000$ in the water and $\rho = 1$ in the air, with $\mathbf{u} = 0$ and $p = 0$ everywhere. The solver is run for one real-time step of size $\Delta t = 0.01$

seconds with the following settings: three Runge-Kutta stages, an artificial compressibility coefficient of $\beta = 1100$, and a Courant number for pseudo-time of 0.48 to satisfy the explicit stability constraint. The Green-Gauss gradient calculation is used for ρ and $\mathbf{u}$, and the corresponding limiter changed for each simulation to show its effect on convergence.

Recall that the solution at each real-time step is reached when the solution converges in pseudo-time. This is measured with residuals, calculated throughout this chapter as the maximum absolute difference between a conserved variable in the current and previous pseudo-time steps. Figure 6.2 shows that, of all the limiters available in OpenFOAM as standard, only `faceLimited` converges in this case, and not very smoothly. It might seem like the convergence problem is because all the OpenFOAM limiters are non-differentiable, as discussed in Section 6.1.4. However, Figure 6.2 shows that, despite being differentiable, `cellLimited<Michalak>` ($K = 0$) does not converge either. It is only when the limiter is switched off in areas of uniform flow that convergence is achieved ($K = 1$). Clearly, when $K = 1$, the newly implemented limiter `cellLimited<Michalak>` has improved convergence properties compared to the options already present in OpenFOAM.

It is important to note that, although the residuals stall for nearly all of the other limiters, there is no instability and therefore the simulations do not blow up. Only a few cells fail to converge, and they alternate between very similar states, which means the final results do not differ substantially between the limiters. Consider Figure 6.3. The residuals only stall in the top-left corner above the water column. Once K is changed to 1 for `cellLimited<Michalak>`, the limiter switches off in this area of uniform flow.

6.2.2 Comparison with `interFoam` (2D)

The new solver is tested using OpenFOAM's `damBreak` tutorial to compare its performance with `interFoam`. This tutorial simulates a dam break on a 2D mesh featuring an obstacle at the base. Initial conditions are $\rho = 1000$ in the water and $\rho = 1$ in the air, with $\mathbf{u} = 0$ and $p = 0$ everywhere. The simulations are run until $t = 1.0$ or until the residuals stall, whichever is sooner, with the following settings: one Runge-Kutta stage, an artificial compressibility coefficient of $\beta = 1100$, and a Courant number for pseudo-time of 0.48 to satisfy the explicit stability constraint. At each real-time step, the absolute tolerances for

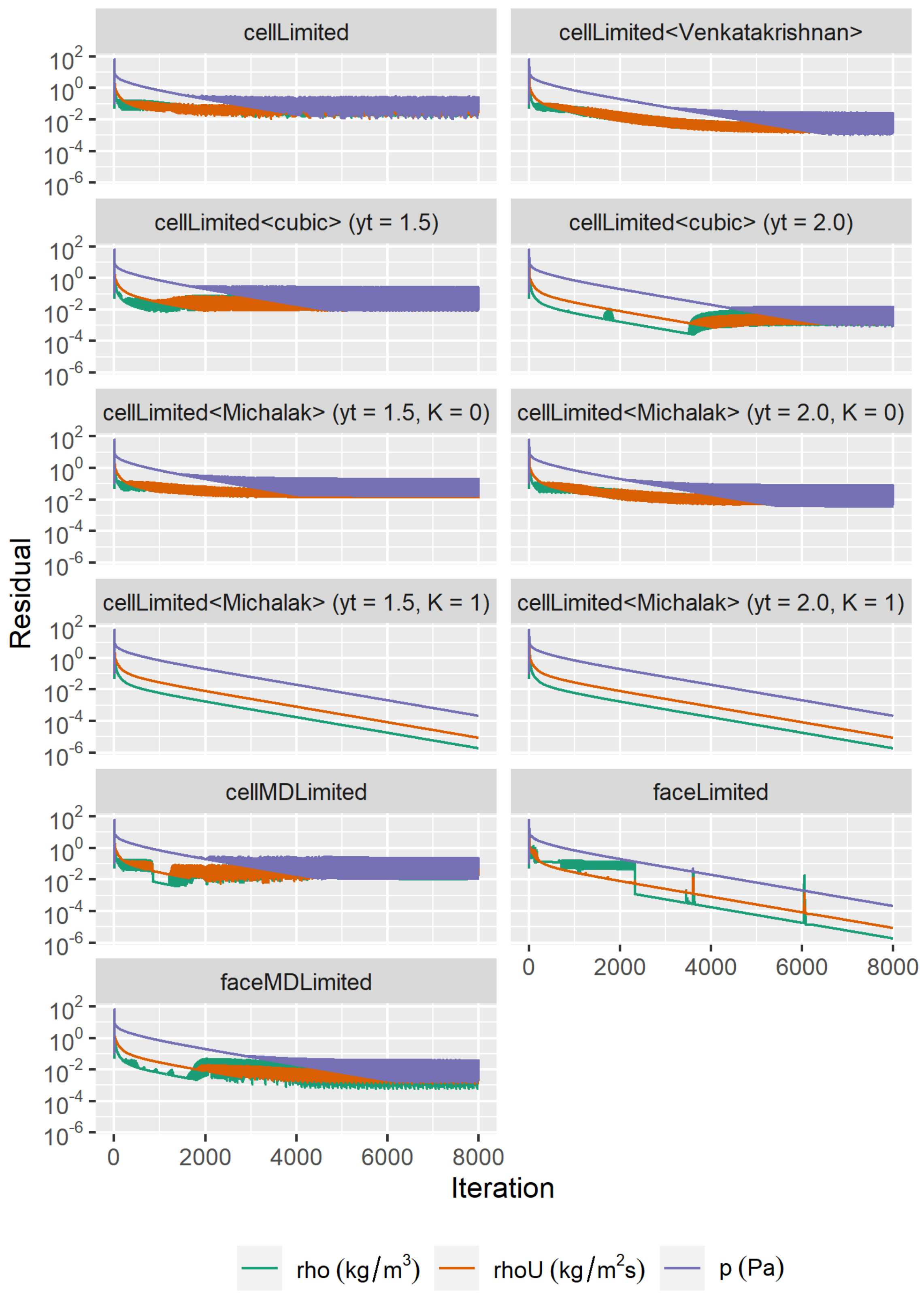

Figure 6.2: Convergence history of absolute residuals for different gradient limiters.

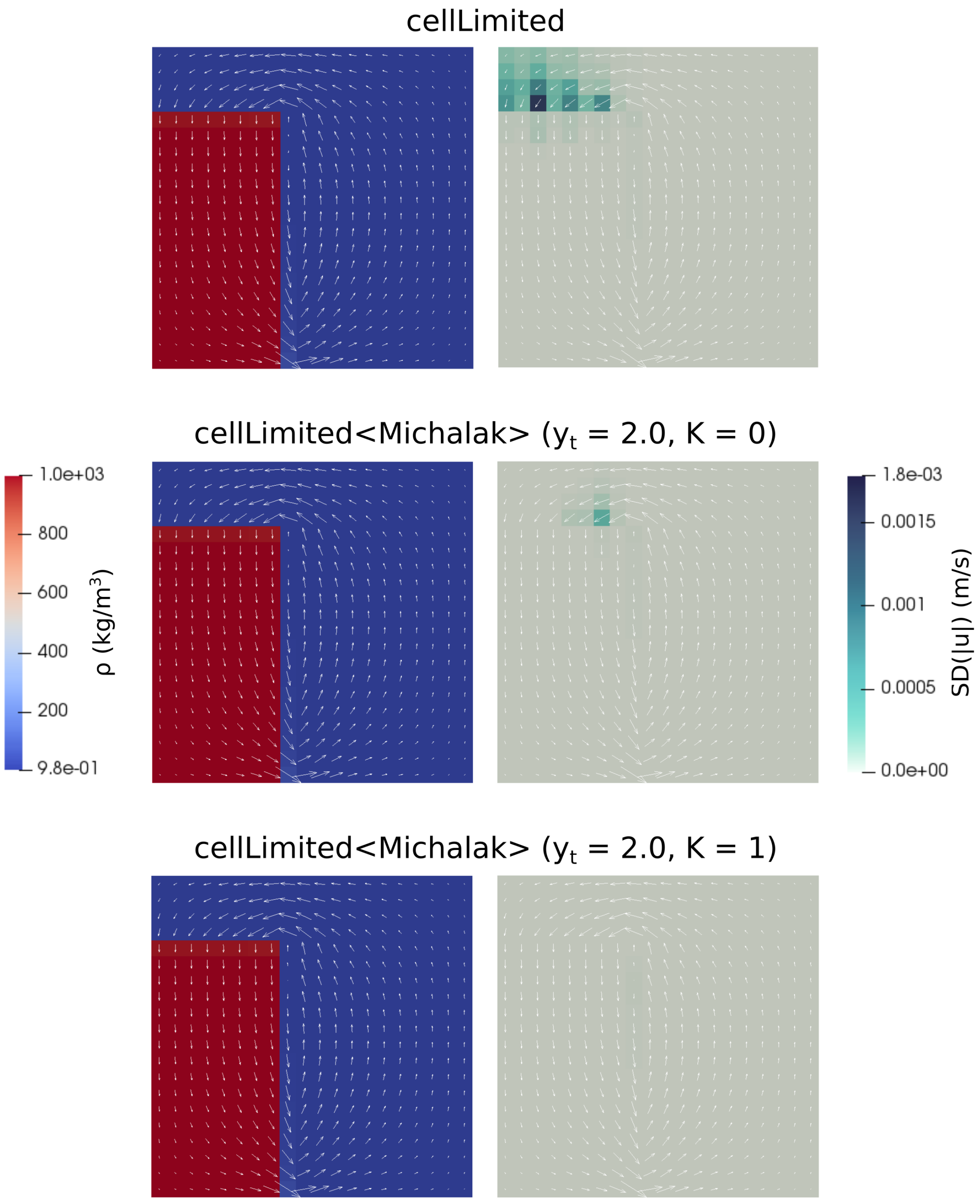

Figure 6.3: Density field after 8000 pseudo-time iterations, and standard deviation of velocity magnitude over pseudo-time iterations 7000–8000 (results saved every 100 iterations).

pseudo-time convergence are 0.0001 for ρ, 0.01 for $\rho\mathbf{u}$, and 0.01 for p. The Green-Gauss gradient calculation is used for ρ and $\mathbf{u}$, and the corresponding limiter is changed along with the Courant number for real-time. Recall that real-time is treated point-implicitly, and so there is no explicit stability constraint for the real-time Courant number.

Figure 6.4 shows how far each simulation can go. The residuals are prone to stalling when the tail of the column of water hit either the obstacle or the far wall and, as expected, `cellLimited<Michalak>` converges better than `cellLimited`. However, sometimes `cellLimited<Michalak>` stalls, and not always with the same parameters. In this particular benchmark, when the real-time Courant number was 1, the choice $y_t = 1.5$ converged but $y_t = 2.0$ stalled, and it was the other way around if the Courant number was 2 or 5. Both choices of y_t stall if the Courant number was 0.5, perhaps due to the increased number of real-time steps meaning there are more chances to stall. Moreover, the effect of switching off the limiter in uniform regions of flow (that is, setting $K > 0$) is not noticeable. This suggests that the standard slope limiters for unstructured meshes are not sufficiently robust for this solver, whether the limiters already available in OpenFOAM or the limiter implemented in this chapter. Again, this is nothing to do with stability. At no point do the simulations blow up or crash; they simply stop converging.

Despite this lack of convergence, we can use parameters that worked for this particular benchmark to compare the new solver, as it stands, with `interFoam`. Figures 6.5–6.6 show the results for `cellLimited<Michalak>` with $y_t = 2.0$, $K = 1$, and real-time Courant numbers of 2 and 5. Figure 6.5 includes results for the `interFoam` simulations, where the settings are as in the standard laminar `damBreak` tutorial, but with zero viscosity and surface tension, and the `atmosphere` patch given a wall boundary condition. Now, `interFoam` keeps the free surface sharp using a compression term[3] with coefficient `cAlpha`. Although the compression term has a physical basis (Rusche, 2002, p.117), in practice `cAlpha` is usually arbitrarily set to 1. Setting `cAlpha` to 0 is equivalent to removing the interface compression term—not a practice acceptable for practical simulations but useful nonetheless for this comparison. Figure 6.5 shows that the new solver with a real-time Courant number of 2 exhibits very similar behaviour to `interFoam` with a `cAlpha` of 0.

[3]Often called artificial compression, completely unrelated to artificial compressibility.

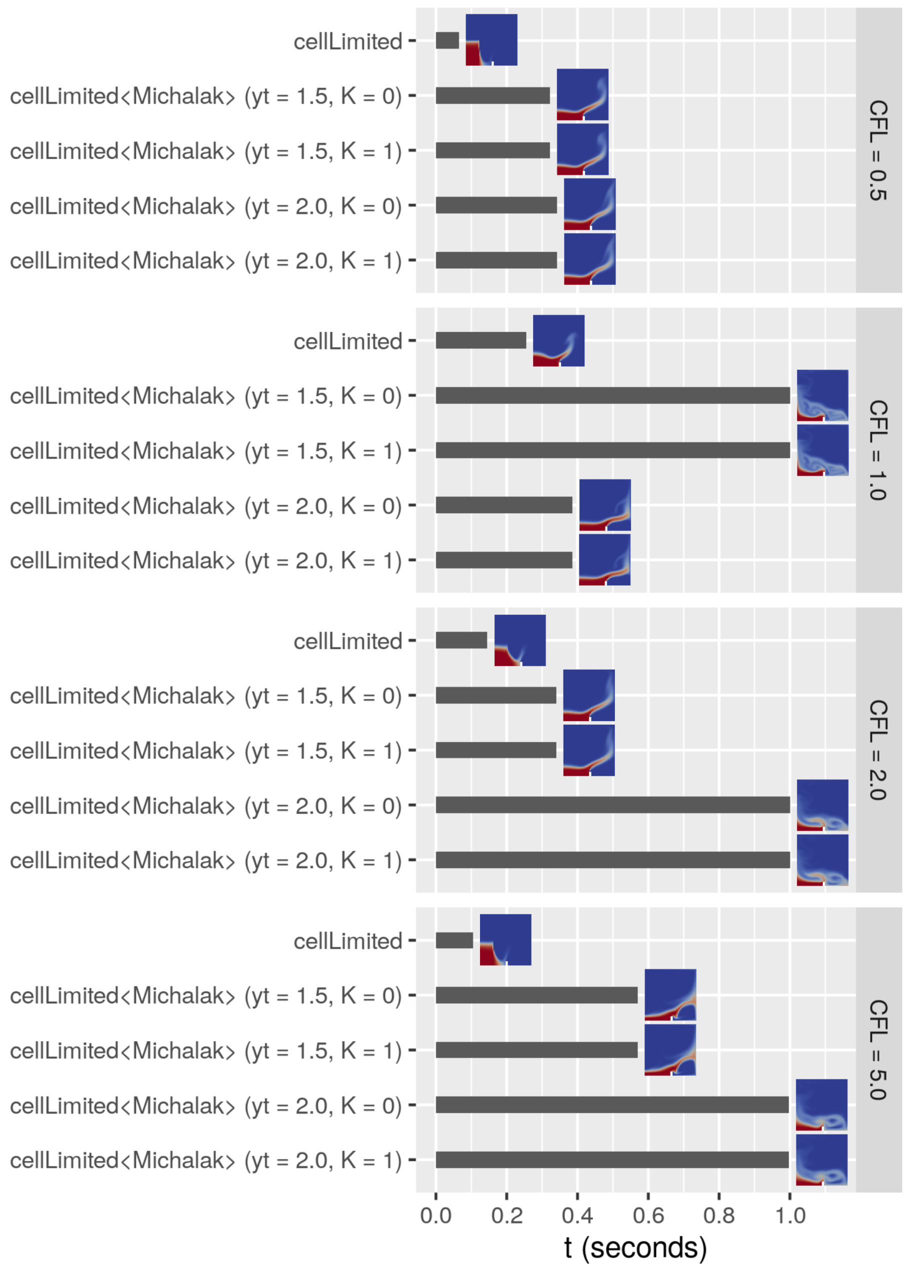

Figure 6.4: Effect of limiter and real-time Courant number on how far the simulation gets and, if convergence is not achieved, when the stall occurs in real-time. Density field shown to highlight problematic moments of the simulation.

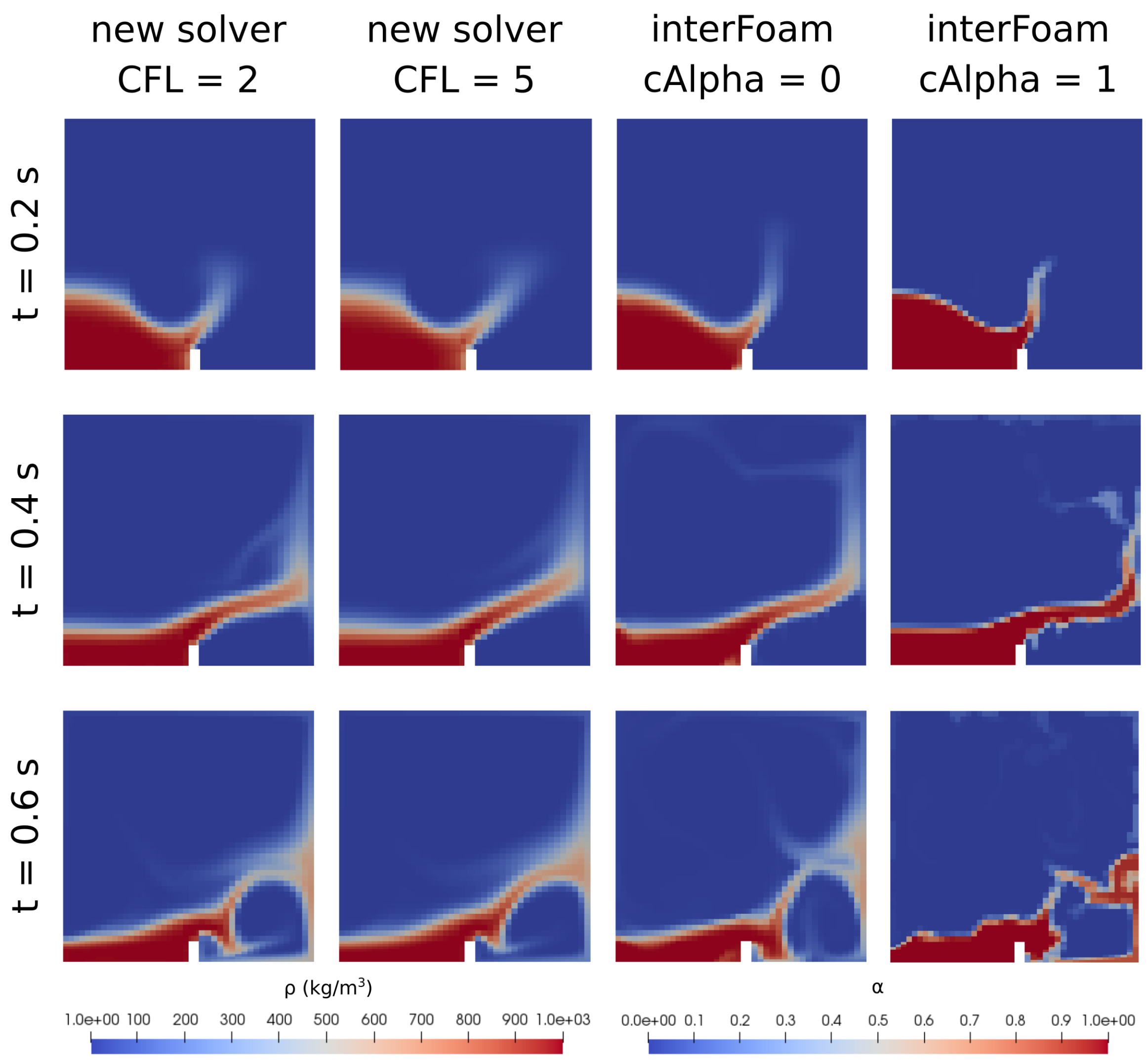

Figure 6.5: Density field for new solver compared with water phase field for `interFoam` in `damBreak` case. The Courant number is for real time.

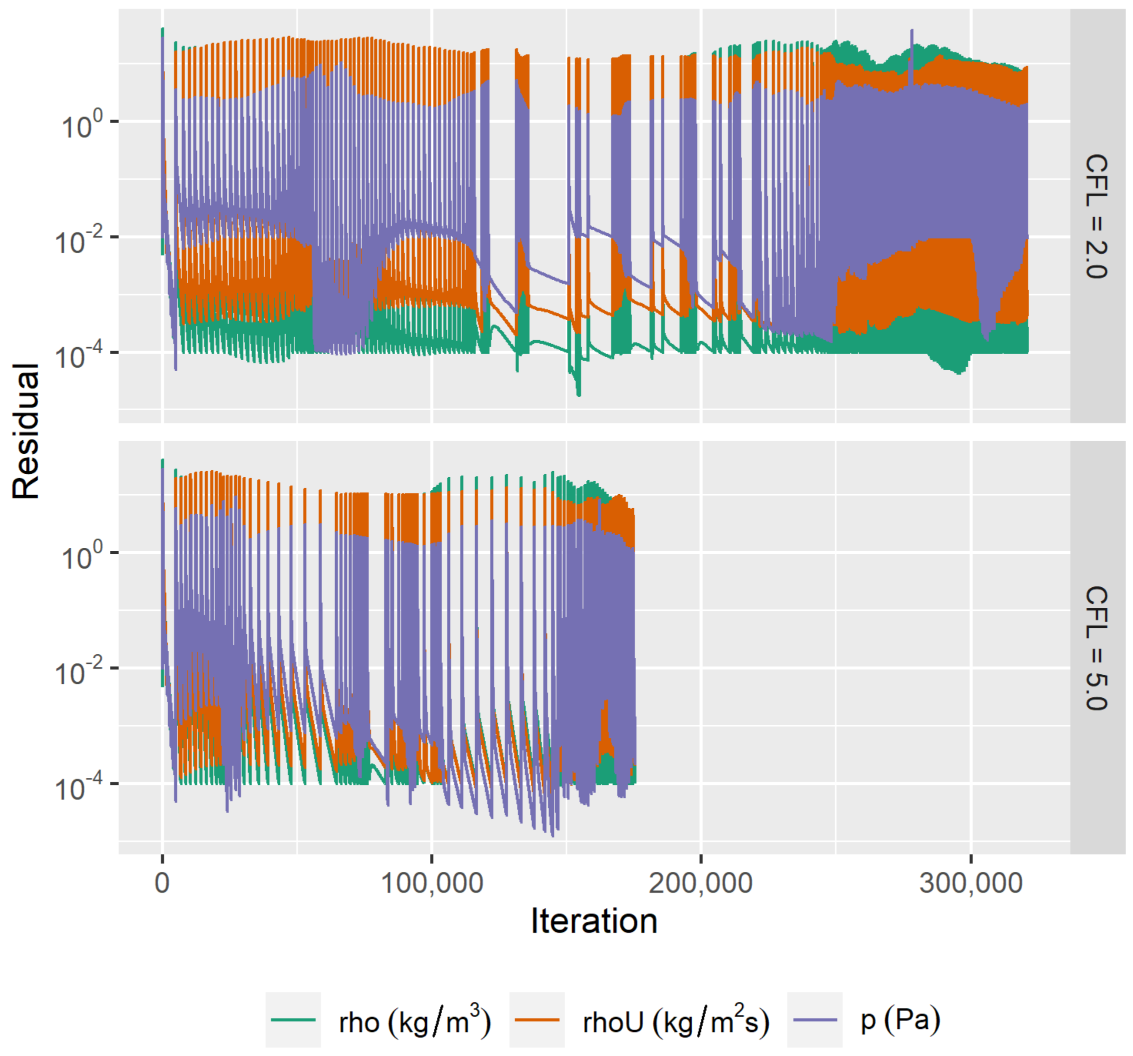

Figure 6.6: Absolute residuals for each pseudo-time iteration required throughout full `damBreak` simulation to $t = 1$.

Surprisingly, when the real-time Courant number is increased to 5, the new solver still manages to capture the highly transient behaviour of the dam break, with only slightly less detail. This is encouraging because fewer total iterations are required for this larger real-time Courant number, as shown by Figure 6.6.

While the solver has potential, Figure 6.5 shows that the use of a high-resolution Godunov-type scheme is not sufficient by itself to keep the interface sharp. Activating the interface compression term does indeed keep the interface sharp when using `interFoam`, but this comes at the cost of a deformed free surface (see the $t = 0.6$ frame). It was

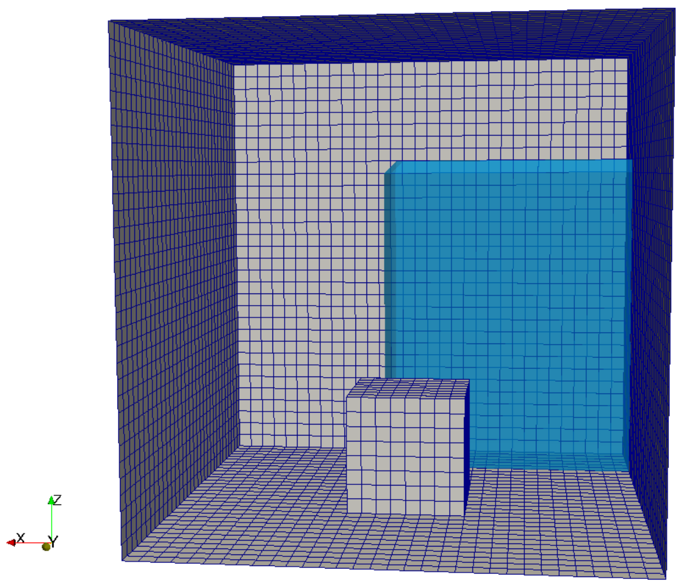

Figure 6.7: Initial column of water for `damBreakWithObstacle` case.

hoped that a Godunov-type scheme might circumvent this problem, but clearly that is not the case, and something like the interface compression term in `interFoam` is required to keep the interface sharp. This has been done before in the context of Godunov-type schemes for artificial compressibility coupled with VOF (Pan and Chang, 2000; Bhat and Mandal, 2019b), and would be a good next step once the convergence problem has been solved.

6.2.3 Comparison with `interFoam` (3D)

The new solver is tested with OpenFOAM's `damBreakWithObstacle` tutorial to compare its performance with `interFoam` on a simple 3D mesh. This tutorial simulates a dam break on a 3D mesh featuring an obstacle at the base. The initial column of water is shown in Figure 6.7. As before, the initial conditions for velocity and pressure are set to zero. The simulations are run until $t = 1.0$ or until the residuals stall, whichever is sooner, with the following settings: one Runge-Kutta stage, an artificial compressibility coefficient of $\beta = 1100$, a Courant number for pseudo-time of 0.2 to satisfy the explicit stability constraint, and a Courant number for real-time of 1.0. At each real-time step, the absolute tolerances for pseudo-time convergence are 0.01 for ρ, 0.01 for $\rho\mathbf{u}$, and 0.1 for p. The Green-Gauss gradient calculation is used for ρ and $\mathbf{u}$, and the corresponding limiter set to `cellLimited<Michalak>` with $y_t = 1.5$ and $K = 1$. Recall that real-time is treated point-implicitly, and so there is no explicit stability constraint for the real-time Courant number.

Figure 6.8 shows how the simulation compares with `interFoam` simulation for the `damBreakWithObstacle` tutorial case, where the `interFoam` solver is run with the same settings as the original laminar tutorial but with zero viscosity and surface tension, and the `atmosphere` patch given a wall boundary condition. The residuals for the new solver are shown in Figure 6.9. Unfortunately, the residuals stall at $t = 0.9$ seconds, and so there is no result at $t = 1.0$ to compare with `interFoam`. However, the results before the solver stall compare well with the `interFoam` simulation when `cAlpha` is zero, as in the previous benchmark. Note that generating the visualisation by clipping at $\rho = 500$ hides the interface diffusion problem established for the new solver in the previous benchmark. Regardless, the present benchmark shows that the general behaviour on a simple 3D mesh is captured well by the solver when it converges.

6.2.4 Mesh from `snappyHexMesh`

Although the meshes in the previous benchmarks look like structured meshes, they are stored as unstructured meshes in OpenFOAM. Therefore, to demonstrate that the solver can be run on arbitrary meshes, it is also tested on a much less uniform mesh generated by `snappyHexMesh`. A simple hydrostatic case is chosen to avoid the convergence problem and because the solution is known, allowing us to test the impact of the pressure gradient calculation on the simulation.

The mesh from the `iglooWithFridges` tutorial is used. While the tutorial is designed to model different physics from those considered in this project (the temperature in an igloo containing two fridges), the mesh is ideal for our purposes as it is very irregular. All the boundary conditions are changed to walls, and the domain filled with water up to $z = 2$, as shown in Figure 6.10. As before, the initial conditions for velocity and pressure are set to zero. The solver is run for one real-time step of size $\Delta t = 0.01$ seconds with

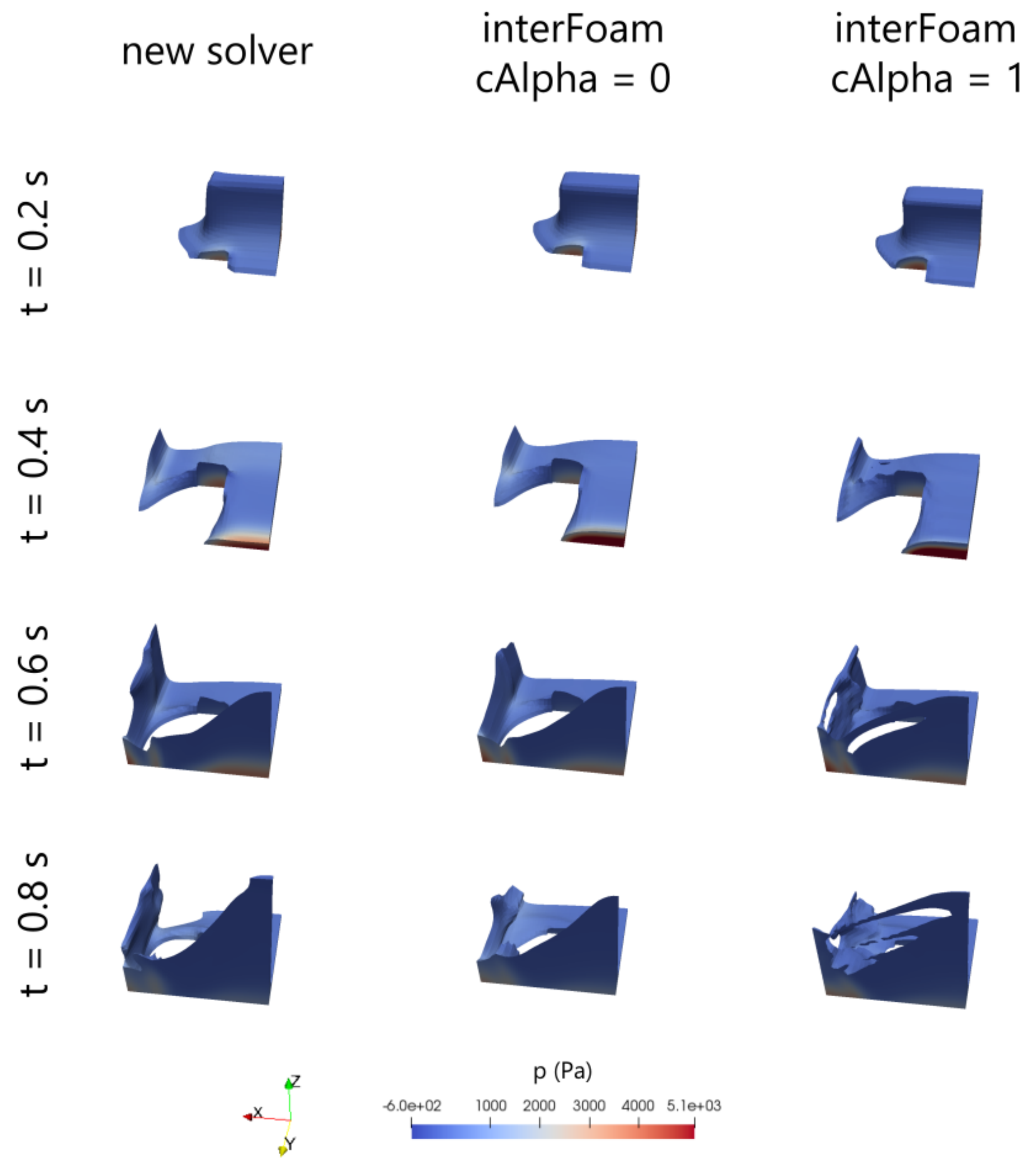

Figure 6.8: Pressure in water phase throughout `damBreakWithObstacle` case for new solver and `interFoam`.

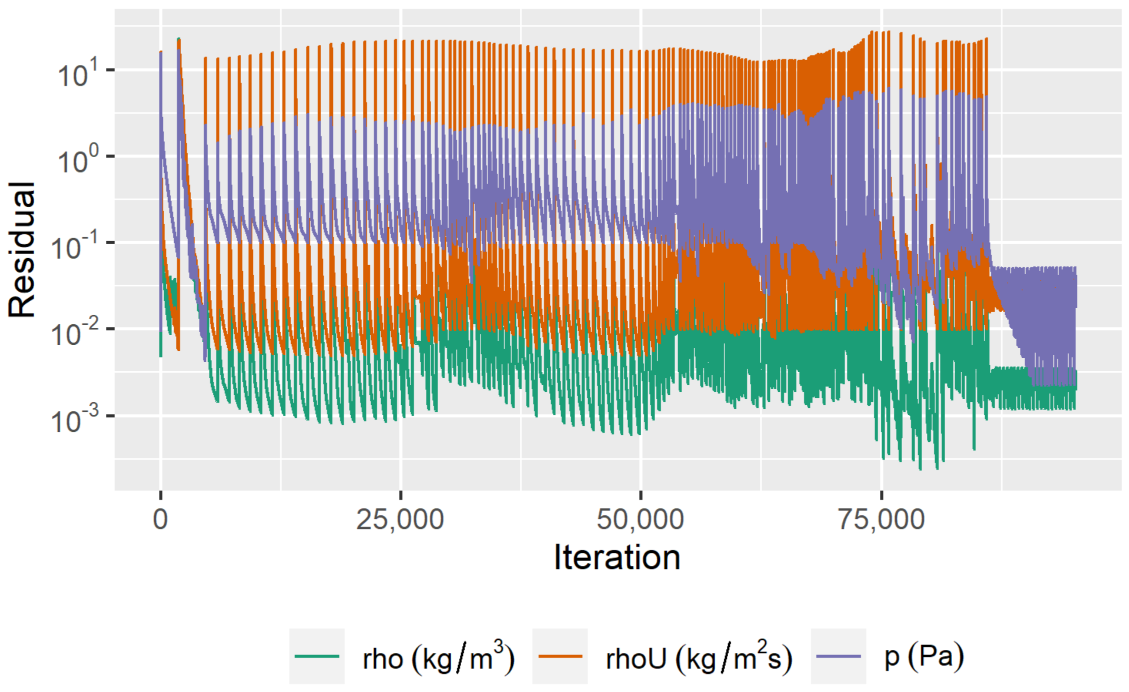

Figure 6.9: Absolute residuals for each pseudo-time iteration throughout `damBreakWithObstacle` simulation.

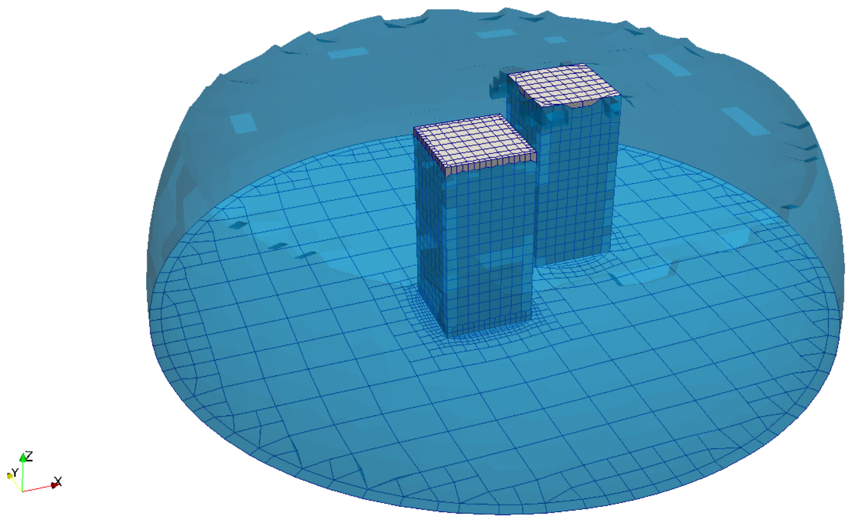

Figure 6.10: Water partially filling the igloo in hydrostatic case (converged solution with ∇p from Chapter 5).

the following settings: one Runge-Kutta stage, an artificial compressibility coefficient of $\beta = 1100$, and a Courant number for pseudo-time of 0.48 to satisfy the explicit stability constraint. The absolute tolerances for pseudo-time convergence are 0.01 for ρ, 0.01 for $\rho\mathbf{u}$, and 0.01 for p. The Green-Gauss gradient calculation is chosen for ρ and $\mathbf{u}$, and the corresponding limiter set to `cellLimited<Michalak>` with $y_t = 1.5$ and $K = 1$.

Recall that the pressure gradient calculation throughout this chapter uses the new method developed in Chapter 5. It is based on a rearrangement of the momentum equation, and so it is automatically well-balanced, that is, it balances exactly with the gravity source term when the velocities are zero. While there are other well-balanced methods (Qian et al., 2006; Ntouras and Papadakis, 2020; Kruisbrink et al., 2018; Vukčević et al., 2018; Queutey and Visonneau, 2007), this one is very easy to implement on unstructured meshes. Therefore, we investigate its effectiveness on unstructured meshes by comparing it to when the pressure gradient is calculated instead by `cellLimited<Michalak>` with $y_t = 1.5$ and $K = 1$.

Figure 6.11 shows that the new pressure gradient calculation from Chapter 5 indeed performs better along the direction of gravity than the limiter `cellLimited<Michalak>`, which is too limiting. The resulting difference in the pressure and velocity in the direction of gravity is notable. The small but non-zero velocities in the other directions can be attributed to the free surface not starting out entirely flat due to the irregular shape of the mesh. Figure 6.12 shows that the improved performance of the well-balanced limiter comes at the cost of slower convergence, but this is primarily due to the fact that it takes longer for the pressure field to converge to the correct magnitude from $p = 0$. In any case, this test demonstrates that the new solver can be run on 3D unstructured meshes and that the pressure gradient calculation developed in Chapter 5 retains its well-balancedness, although not as cleanly as on a 2D Cartesian mesh.

6.3 Compression near obstacle

There is a problem that is not immediately apparent in the figures shown so far. In the simulations of dam breaks around obstacles, water close to the obstacle can have a density

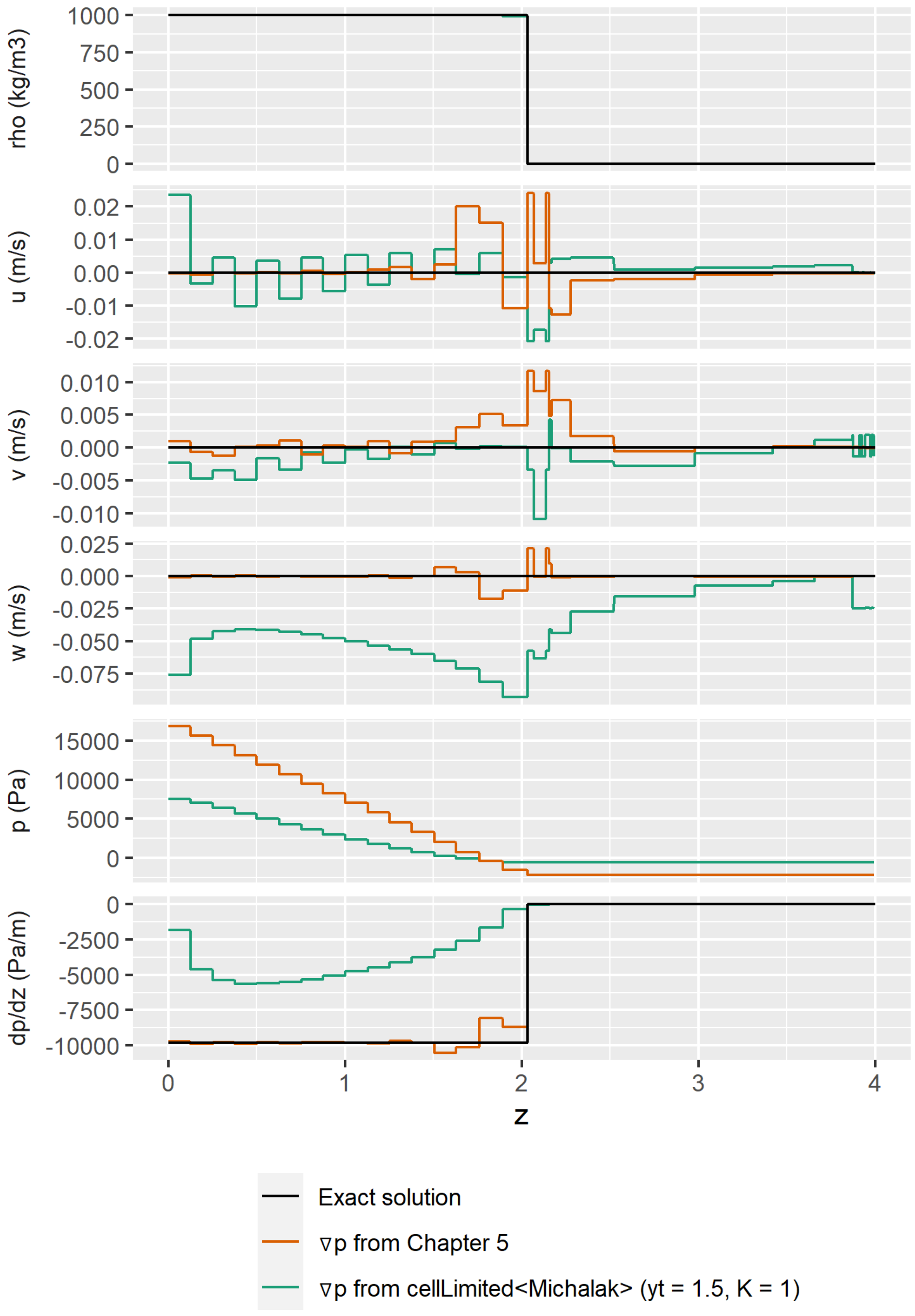

Figure 6.11: Cross section of cell-centred values along the z-axis near the centre of the igloo at $x = 3.2, y = 3$ for different ∇p calculations.

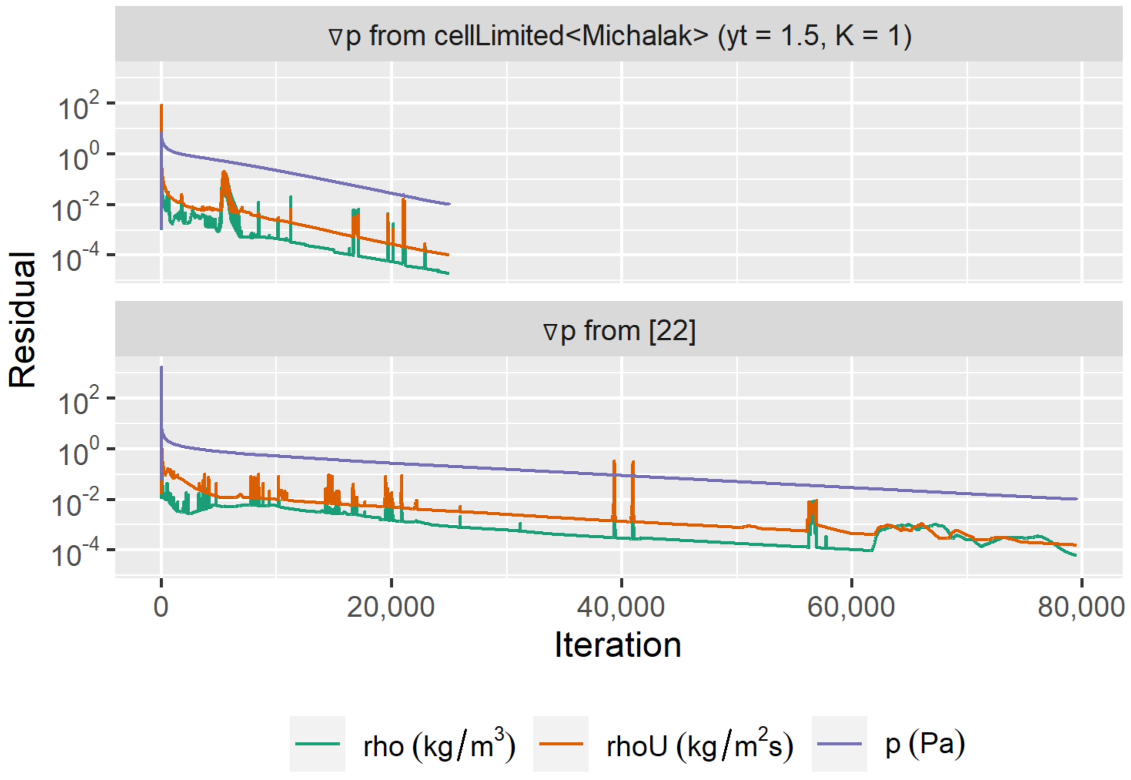

Figure 6.12: Absolute residuals for each pseudo-time iteration required throughout hydrostatic simulation.

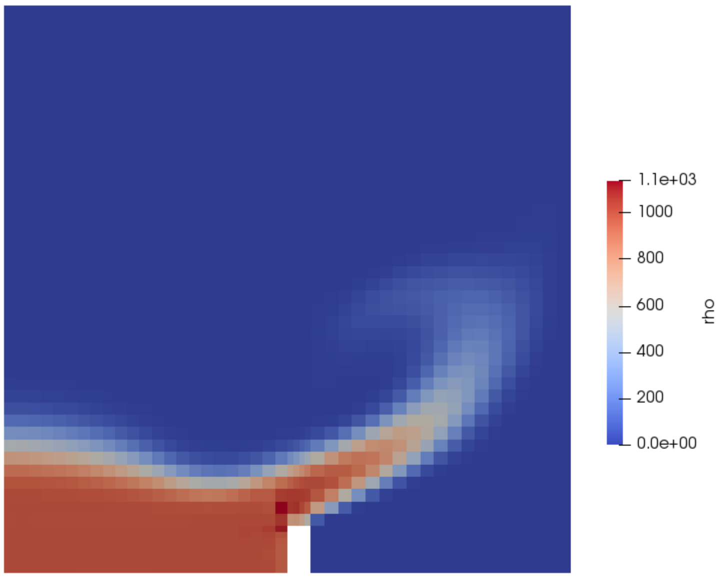

Figure 6.13: Density field at $t = 0.28$s in 2D dam break simulation for CFL = 2.

higher than 1000kg/m^3, which is what it should be at the incompressible limit. However, this problem does not occur throughout the whole simulation or in many cells. To illustrate, Figure 6.13 shows the density field at a particular time step during the 2D case. Cells containing only water should have $\rho = 1000$, those with only air $\rho = 1$, and cells with a mixture $0 < \rho < 1000$. However, it can be seen that there are a few cells close to the obstacle for which $\rho \approx 1100$. Figure 6.14 shows a similar effect in the 3D case, but with $\rho \approx 1400$. Given the discussion in Section 5.4 of how pressure gradients can affect the density field, this is probably caused by the pressure gradient. To check that it was not caused by the complexities of the 3D implementation, we modified the 2D prototype from Chapter 5 to include simple horizontal obstacles, but the problem appeared there also. This requires further investigation.

6.4 Conclusion

In OpenFOAM, the traditional way to model free-surface flows is with the VOF solver `interFoam`, which uses the PIMPLE algorithm based on matrix inversion. However, there are other ways to deal with the lack of a pressure equation in the incompressible Navier-Stokes equations. This chapter focuses on the method of artificial compressibility, which

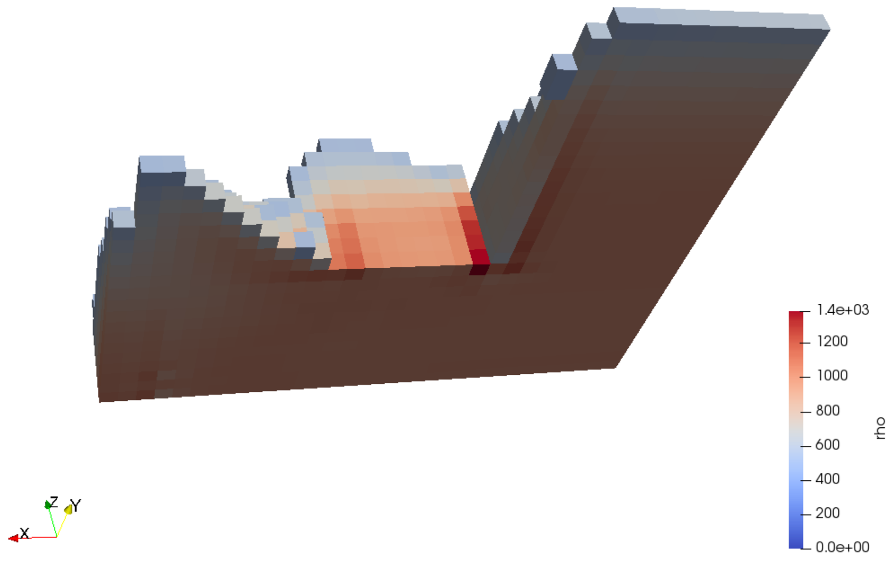

Figure 6.14: Density field at $t = 0.36$s in 3D dam break simulation. Only cells with $\rho > 500\text{kg/m}^3$ shown for clarity.

scales better than matrix inversion as it requires less communication between processors. Indeed, although artificial compressibility is not new, it is currently very relevant due to its suitability for implementation on massively parallel architectures (Hodges, 2020; Nourgaliev et al., 2004; Loppi, 2019).

Despite its potential for efficient parallelisation, artificial compressibility has not been investigated extensively for variable density flows (including free-surface flows), having only been implemented on 2D and structured meshes so far (Kelecy and Pletcher, 1997; Qian et al., 2006; Bassi et al., 2018). The present chapter harnesses OpenFOAM to implement artificial compressibility for variable density incompressible flows on 3D unstructured meshes. We found that, since pseudo-time convergence is required at every real-time step, the slope limiter used in the MUSCL reconstruction step is critical, and the limiters currently in OpenFOAM are not fit for this purpose. Therefore, we implemented the Michalak and Ollivier-Gooch (2008) limiter fully to avail of its improved convergence properties. When the results converge, they both compare well with `interFoam` and are well-balanced, but even the improved limiter is not sufficiently robust to ensure convergence. While the solver requires a mechanism to keep the free surface sharp as in `interFoam`, and the compression near obstacles needs to be eliminated, the convergence problem is more critical.

Without convergence, there is no solution.

Consequently, further research should first focus on alternative routes to second-order accuracy than the Barth and Jespersen (1989) framework for MUSCL. This could involve smoothing out the multidimensional limiter `cellMDLimited` rather than `cellLimited`, or implementing the WENO (Liu et al., 1994) method. Of course, convergence is necessary but not sufficient for a good solution, and so more work would need to be done after achieving robust convergence: a mechanism could be employed to keep the interface sharp, the problem with the density exceeding 1000kg/m^3 could be investigated, the solver could be compared thoroughly with `interFoam`, and the solver could then be applied to practical cases such as in Chapter 4. Ultimately, this will make it possible to use Riemann solvers to capture the free surface automatically and in a way that is well suited to the massively parallel architectures of today.

Chapter 7. Conclusion

Leaky barriers mimic naturally occurring obstructions to channels such as accumulations of Large Woody Debris (LWD) or beaver dams, slowing down flows, creating temporary storage in channels, and forcing water onto floodplains. Aiming to reduce flood hazard and pollutant transport without adversely affecting wildlife, they are an increasingly popular nature-based solution in the UK and globally. However, researchers have struggled to quantify the effects of leaky barriers in the field. Consequently, we do not know which barrier designs work best for different locations, or even if they are worth installing at all. This thesis sets out to address the evidence gap using hydrodynamic modelling.

7.1 Motivation for hydrodynamic modelling

Modelling is one of the key ways hydrologists investigate and predict the behaviour of rivers. However, as the fundamental behaviour of leaky barriers is not well understood, researchers have had to guess how the barriers behave individually to model them as part of a river network. Chapter 1 outlined the five main ways this has been done within both hydrological and hydraulic models: indirect comparison (Slowing the Flow Partnership, 2016), the storage volume method (Nisbet et al., 2015; Norbury et al., 2018), changing model parameters (Kitts, 2010; Odini and Lane, 2010; Dixon et al., 2016), changing the channel geometry (Thomas and Nisbet, 2012; Valverde, 2013), and the hydraulic structure method (Milledge et al., 2015; Metcalfe et al., 2017; Cabaneros et al., 2018). These existing leaky barrier modelling methods are, at best, data-based rather than physics based. Ultimately, there is a need to understand how individual barriers behave before modelling them as part of a network.

Since measuring the impact of leaky barriers in the field is inherently difficult, Computational Fluid Dynamics (CFD) modelling is a promising method to use instead. There have been two CFD studies of leaky barriers in the literature (Allen and Smith, 2012; Xu and Liu, 2017), but these both used the fixed-lid method to deal with the free surface, where the water elevation is an input to the model, rather than an output. Such a method cannot be used to investigate the backwater effect. We need a method that models the free surface to predict how a leaky barrier will perform.

This thesis sets out to incorporate the physical behaviour of leaky barriers into modelling practice. There are two focuses. First, to address the lack of dedicated leaky barrier modelling tool, we investigate the hydraulic structure method for including leaky barriers in hydraulic models, as recommended by Addy and Wilkinson (2019) in their review paper. Second, to address the lack of understanding in how individual leaky barriers behave, we set out to simulate leaky barriers with non-fixed-lid 3D CFD modelling. Both threads of research include high-resolution shock-capturing Godunov-type schemes (Godunov, 1959), the building blocks of which are Riemann solvers. This work on Godunov-type schemes is where the majority of the technical novelty of the thesis lies.

7.2 Simplified modelling with the shallow water equations

Chapter 2 investigates the hydraulic structure method for inserting leaky barriers into hydraulic models. It does this by drawing on the wider hydraulic modelling literature (Zhao et al., 1994; Cozzolino et al., 2015) and representing leaky barriers as internal boundary conditions in a 1D Godunov-type scheme written from scratch in Python. The mass and momentum fluxes at the barrier come from steady-state formulae that depend on the operating stage, worked out dynamically based on the Riemann states. The results compare well to 55 steady-state flume experiments across a range of operating stages. The work in this chapter has been published in Leakey et al. (2020).

Replacing the Riemann solver flux with a steady-state discharge equation, while relatively straightforward to implement, captures the full range of behaviour at the barrier. Moreover, unlike previous studies in the literature, we tested the model behaviour against hydraulic flume data for a number of different barrier configurations and operating stages, providing confidence in the model results and the wider applicability of the method.

A major advantage of the method is its shock-capturing capabilities. Since the base numerical scheme is Godunov-type, it can capture shocks such as flood waves, hydraulic jumps, and bores. Traditional hydraulic models struggle to replicate these extremes of physical behaviour due to their reliance on the method of finite differences, so a Godunov-type scheme is a good framework in which to insert leaky barriers if creating a model from scratch. Some relatively new models do use Godunov-type schemes for the shallow water equations, for example, CityCAT (Glenis et al., 2018), Iber (Bladé et al., 2014), JFLOW SWE (Crossley et al., 2010), and TUFLOW FV (BMT, 2013), and therefore this method could be implemented in existing software too. Whether as part of a bespoke numerical scheme or existing software, embedding leaky barriers in a Godunov-type scheme for the shallow water equations means that the all the impacts of the barriers on the hydraulics can be captured. For example, our transient experiments are able to capture the propagation of a flood wave through different configurations of barrier, both the bore that travels back upstream and the bore that travels downstream.

Further research is required before incorporating different leaky barrier designs than the simple weir/gate combination (Samani and Mazaheri, 2009) considered here. This is what the remainder of the thesis sets out to address, using CFD modelling combined with flume experiments.

7.3 Detailed modelling with the Navier-Stokes equations

The majority of the thesis is about 3D CFD modelling. Chapter 3 outlines the theory behind CFD and Chapter 4 investigates using OpenFOAM's VOF solver `interFoam` to model individual leaky barriers systematically. This involves developing a new boundary condition and post-processing utility in OpenFOAM, creating high-quality meshes in Ansys, running the CFD simulations in OpenFOAM, and then comparing the results to new flume data. This work is significant as it is the first non-fixed-lid CFD simulation of leaky barriers in the literature, and so is able to capture the backwater effect induced by different barrier configurations. However, while OpenFOAM has many good aspects, its numerical methods are not the most up to date. Indeed, being based on semi-implicit pressure-based algorithms, the VOF solver `interFoam` theoretically has the decreased accuracy and increased computational effort of implicit schemes without the advantage of very large time steps.

Dissatisfied with this theoretical snag, in Chapters 5–6, we develop a new free-surface solver based on the method of artificial compressibility (Chorin, 1997). Artificial compressibility allows methods for hyperbolic conservation laws, such as Godunov-type schemes, to be used for the incompressible Navier-Stokes equations. This means that the free surface can be captured automatically by the contact wave in the Riemann solver, without having to rely on complicated surface-tracking or surface-capturing algorithms (Kelecy and Pletcher, 1997; Qian et al., 2006; Bassi et al., 2018). The new solver is explicit in pseudo-time and point-implicit in real-time (Vermeire et al., 2019), and so has the parallelisation potential of an explicit scheme without the requirement for small real-time steps. We also develop a new way to calculate the MUSCL pressure gradient, based on a rearrangement of the momentum equation, and implement an Osher Riemann solver that has not been used for these governing equations before (Dumbser and Toro, 2011).

In Chapter 5, we implement the solver from scratch in Python for 2D Cartesian meshes, using this prototype to investigate the effect of different Riemann solvers and pressure gradient calculations used in the MUSCL reconstruction step. We show that the Osher Riemann solver (Dumbser and Toro, 2011) is a more robust alternative to the Bassi (Bassi et al., 2018) or Roe Riemann solvers (Roe, 1981). Moreover, the new pressure gradient calculation—a completely novel method developed in this thesis—is able to capture automatically the pressure gradient discontinuity at the free surface. While these improvements do not have much of an effect in the Martin and Moyce (1952) experimental dam break case, the pressure gradient calculation does have the advantage of being easy to implement on both structured and unstructured meshes as it uses information that has already been calculated. The work in this chapter has been published in Leakey et al. (2022b).

In Chapter 6, we implement the new solver in OpenFOAM, the first implementation of the variable-density artificial-compressibility equations on 3D unstructured meshes. Generalising the solver this way is primarily a programming challenge. However, one aspect of this work is more than a generalisation, requiring a completely different method to structured meshes—the MUSCL reconstruction—and this leads to problems with convergence. This trouble with convergence is a well-known problem, and there are ways to overcome it, at least partially (Barth and Jespersen, 1989; Venkatakrishnan, 1995; Michalak and Ollivier-Gooch, 2008). Unfortunately, despite creating extra functionality in OpenFOAM to improve convergence and fixing a bug in the official code, the new solver's convergence problem on unstructured methods is yet to be completely resolved and is recommended for future work. In any case, the results compare well with results from the `interFoam` solver in 2D and 3D. The work in this chapter has been published in Leakey et al. (2022a).

The new solver is applicable to any incompressible variable density flows, and so it has a wide area of application beyond leaky barriers. For example, it can be used to simulate other flows of water, like fluvial flows around bridges or coastal flows at groynes, and more general multiphase flows, including that of multiple liquids. The OpenFOAM implementation is critical for these applications as 3D unstructured meshes give the required flexibility to capture complex geometries of practical interest. Moreover, not only can meshes created within OpenFOAM be used in the solver, but meshes can be converted from other meshing software (e.g. Ansys) into the OpenFOAM format, providing even greater flexibility. Additionally, as the solver is implemented in OpenFOAM, it has the advantage of being compatible with the wealth of pre- and post-processing utilities contained in the software. This flexibility is the benefit derived from overcoming the programming and theoretical challenges of generalising the solver from 2D Cartesian meshes to 3D unstructured meshes.

7.4 Recommendations for future work

The validation data for the 1D model in Chapter 2 are limited to steady-state leaky barrier experiments. For further confidence in the model, it must be tested against transient leaky barrier experiments too. These data could come from the new high-resolution cameras that have been installed in the Novak Lab at Newcastle University. This would provide confidence that the model can capture accurately how leaky barriers slow down flows and

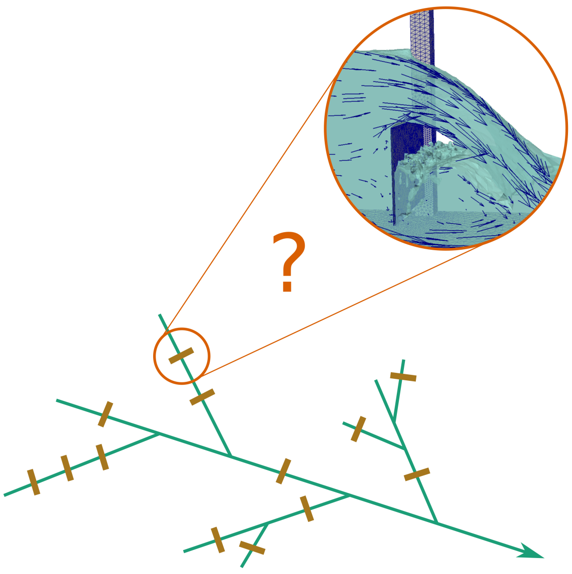

Figure 7.1: Multiple scales of modelling... how do we connect them?

reduce flood peaks. Further work could then focus on the interaction of multiple barriers across a network, making it possible to investigate the synchronisation problem as well as optimisation of barrier location and design. Such an investigation, if carried out in full, could involve a large number of simulations, and so an optimisation algorithm could be used to cut them down to a manageable number.

The new CFD solver developed in Chapters 5–6 requires further work before use on practical applications such as leaky barriers. The first priority is to ensure more reliable convergence; without convergence, there are no results. It would be worthwhile to then implement a method that keeps the free surface sharp. The problem of compression near obstacles also requires further investigation. A major advantage of the solver is its favourable parallelisation properties, allowing efficient implementation on GPUs, as in Loppi (2019) and Kajzer and Pozorski (2020). This would facilitate modelling many different configurations of leaky barrier. However, this would require an implementation outside the OpenFOAM framework due to its dependence on the method of domain decomposition.

Questions remain about how to incorporate the results of the 3D CFD modelling into 1D hydraulic modelling, as illustrated in Figure 7.1. Do we extract empirical relationships from systematic CFD modelling before running the hydraulic model, or dynamically link between the two scales of model at run time, or use the CFD modelling as validation for a more efficient method? These questions are relevant for other hydraulic structures that are also currently not well-understood, for example, bridges.

7.5 Multi-scale modelling

To address the evidence gap related to leaky barriers, we need to move beyond simple methods like changing the value of Manning's *n*—useful as they were for a first approximation—towards numerical methods that represent the physics accurately. To achieve this, modelling is required at both the small and large scales. Small-scale CFD modelling is needed to understand how individual barriers behave, and large-scale hydraulic modelling to understand how networks of barriers behave.

At both scales, explicit Godunov-type schemes, such as those developed in this thesis, provide high-resolution results and the opportunity for efficient parallelisation on massively parallel hardware architectures. Ultimately, this parallelisation potential provides the ability to investigate many barrier configurations, both in form and location, and thus contribute to answering fundamental questions such as the practical problems surrounding leaky barrier placement and design.

References

Addy, S. and Wilkinson, M. E. (2019). Representing natural and artificial in-channel large wood in numerical hydraulic and hydrological models. *WIREs Water*, 6(6):e1389.

Aftosmis, M., Gaitonde, D., and Tavares, T. S. (1995). Behavior of linear reconstruction techniques on unstructured meshes. *AIAA Journal*, 33(11):2038–2049.

Akan, A. O. (2006). *Open Channel Hydraulics*. Butterworth-Heinemann, Oxford.

Allen, J. B., Smith, D. L., Eslinger, O. J., and Valenciano, M. A. (2008). A new approach to streambed modeling and simulation using CFD. In *2008 DoD HPCMP Users Group Conference*, pages 3–8.

Allen, J. D. and Smith, D. L. (2012). Characterizing the impact of geometric simplification on large woody debris using CFD. *International Journal of Hydraulic Engineering*, 1(2):1–14.

Almeland, S. K. (2018). Implementation of an air-entrainment model in interFoam. In Nilsson, H., editor, *Proceedings of CFD with OpenSource Software*. Chalmers University of Technology.

Anderson, J. D. (1995). *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill.

Anderson, W. K., Thomas, J. L., and Van Leer, B. (1986). Comparison of finite volume flux vector splittings for the Euler equations. *AIAA Journal*, 24(9):1453–1460.

Ansys (2009). Overview of flow solvers. `https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node360.htm`. Accessed: 18 May 2022.

Barlow, J., Moore, F., and Burgess-Gamble, L. (2014). Working with natural processes to reduce flood risk: Science report. Technical report, Environment Agency.

Barth, T. and Jespersen, D. (1989). The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*, Reno.

Bassi, F., Massa, F., Botti, L., and Colombo, A. (2018). Artificial compressibility Godunov fluxes for variable density incompressible flows. *Computers and Fluids*, 169:186–200.

Bates, P. D., Lane, S. N., and Ferguson, R. I. (2005). *Computational Fluid Dynamics: Applications in Environmental Hydraulics*. Wiley.

Bayon-Barrachina, A. and Lopez-Jimenez, P. A. (2015). Numerical analysis of hydraulic jumps using OpenFOAM. *Journal of Hydroinformatics*, 17(4):662–678.

Bennett, S. J., Ghaneeizad, S. M., Gallisdorfer, M. S., Cai, D., Atkinson, J. F., Simon, A., and Langendoen, E. J. (2015). Flow, turbulence, and drag associated with engineered log jams in a fixed-bed experimental channel. *Geomorphology*, 248(C):172–184.

Berberović, E., van Hinsberg, N. P., Jakirlić, S., Roisman, I. V., and Tropea, C. (2009). Drop impact onto a liquid layer of finite thickness: Dynamics of the cavity evolution. *Physical Review E*, 79(3):036306.

Berger, M., Aftosmis, M., and Muman, S. (2005). Analysis of slope limiters on irregular grids. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno.

Bhat, S. and Mandal, J. C. (2019a). Contact preserving Riemann solver for incompressible two-phase flows. *Journal of Computational Physics*, 379:173–191.

Bhat, S. P. and Mandal, J. C. (2019b). Modified HLLC-VOF solver for incompressible two-phase fluid flows. Archived at: `https://arxiv.org/abs/1911.11234`.

Bladé, E., Cea, L., Corestein, G., Escolano, E., Puertas, J., Vázquez-Cendón, E., Dolz, J., and Coll, A. (2014). Iber: herramienta de simulación numérica del flujo en ríos. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 30(1):1–10.

Blazek, J. (2005). *Computational Fluid Dynamics: Principles and Applications*. Elsevier, 2nd edition.

BMT (2013). TUFLOW FV science manual. https://downloads.tuflow.com/_archive/TUFLOW_FV/Manual/FV_Science_Manual_2013.pdf. Accessed: 23 February 2022.

Bogoni, M., Canestrelli, A., and Lanzoni, S. (2015). Finite volume modelling of a stratified flow with the presence of submerged weirs. *Journal of Applied Water Engineering and Research*, 3:43–52.

Bokhove, O., Kelmanson, M., and Kent, T. (2018). On using flood-excess volume to assess natural flood management, exemplified for extreme 2007 and 2015 floods in Yorkshire. Archived at: https://eartharxiv.org/87z6w/.

Botta, N., Klein, R., Langenberg, S., and Lützenkirchen, S. (2004). Well balanced finite volume methods for nearly hydrostatic flows. *Journal of Computational Physics*, 196(2):539–565.

Brackbill, J., Kothe, D., and CA, Z. (1992). A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354.

Brewer, C. A. (2022). ColorBrewer 2.0. https://colorbrewer2.org. Accessed: 13 May 2022.

Brown, D. L. (1995). Performance of under-resolved two-dimensional incompressible flow simulations. *Journal of Computational Physics*, 122(1):165–183.

Burgess-Gamble, L. (2020). Working with natural processes—the evidence behind natural flood management. Seminar, NERC-NFM.

Cabaneros, S., Danieli, F., Formetta, G., Gonzalez, R., Grinfield, M., Hankin, B., Hewitt, I., Johnstone, T., Kamilova, A., Kovacs, A., Kretzschmar, A., Kiradjiev, K., Pegler, S., Sander, G., and Wong, C. (2018). JBA Trust challenge: A risk-based analysis of small scale, distributed, “nature-based” flood risk management measures deployed on river networks. Technical report, Maths Foresees.

Chappell, N. (2018). NERC ATSC in natural flood risk management. Lancaster University.

Chen, F. and Hagen, H. (2011). A survey of interface tracking methods in multi-phase fluid visualization. In Middel, A., Scheler, I., and Hagen, H., editors, *Vizualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*, volume 19, pages 11–19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chorin, A. J. (1997). A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 135(2):118–125.

Cifani, P., Michalek, W. R., Priems, G. J. M., Kuerten, J. G. M., van Der Geld, C. W. M., and Geurts, B. J. (2016). A comparison between the surface compression method and an interface reconstruction method for the VOF approach. *Computers and Fluids*, 136:421–435.

Cozzolino, L., Cimorelli, L., Covelli, C., Della Morte, R., and Pianese, D. (2014a). Boundary conditions in finite volume schemes for the solution of shallow-water equations: The non-submerged broad-crested weir. *Journal of Hydroinformatics*, 16(6):1235–1249.

Cozzolino, L., Cimorelli, L., Covelli, C., Della Morte, R., and Pianese, D. (2015). The analytic solution of the shallow-water equations with partially open sluice-gates: The dam-break problem. *Advances in Water Resources*, 80(C):90–102.

Cozzolino, L., Morte, R. D., Cimorelli, L., Covelli, C., and Pianese, D. (2014b). A broad-crested weir boundary condition in finite volume shallow-water numerical models. *Procedia Engineering*, 70:353–362.

Cozzolino, L., Pepe, V., Morlando, F., Cimorelli, L., and Pianese, D. (2017). Exact solution of the dam-break problem for constrictions and obstructions in constant width rectangular channels. *Journal of Hydraulic Engineering*, 143(11).

Crank, J. and Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50—67.

Crossley, A., Lamb, R., and Waller, S. (2010). Fast solution of the shallow water equations using GPU technology. `http://www.jflow.co.uk/sites/default/files/Crossley%20Lamb%20Waller%20-%20BHS%202010.pdf`. Accessed: 23 February 2022.

Cui, Y., Liang, Q., Wang, G., Zhao, J., Hu, J., Wang, Y., and Xia, X. (2019). Simulation of hydraulic structures in 2D high-resolution urban flood modeling. *Water*, 11(10):2139.

Dadson, S. J., Hall, J. W., Murgatroyd, A., Acreman, M., Bates, P., Beven, K., Heathwaite, L., Holden, J., Holman, I. P., Lane, S., O'Connell, E., Penning-Rowsell, E., Reynard, N., Sear, D., Thorne, C., and Wilby, R. (2017). A restatement of the natural science evidence concerning catchment-based 'natural' flood management in the UK. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473.

Dal Maso, G., LeFloch, P. G., and Murat, F. (1995). Definition and weak stability of nonconservative products. *Journal de Mathematiques Pures et Appliquées*, 74:483–548.

Damián, S. (2013). *An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces*. PhD thesis, Universidad Nacional Del Litoral.

Daniels, M. D. and Rhoads, B. L. (2003). Influence of a large woody debris obstruction on three-dimensional flow structure in a meander bend. *Geomorphology*, 51(1):159–173.

Daniels, M. D. and Rhoads, B. L. (2004). Effect of large woody debris configuration on three-dimensional flow structure in two low-energy meander bends at varying stages. *Water Resources Research*, 40(11). W11302.

Darwish, M. and Moukalled, F. (2003). TVD schemes for unstructured grids. *International Journal of Heat and Mass Transfer*, 46(4):599–611.

Defra (2005). Making space for water: Taking forward a new government strategy for flood and coastal erosion risk management in England. Technical report, Defra.

DeGroot, C. T. and Straatman, A. G. (2011). A finite-volume model for fluid flow and nonequilibrium heat transfer in conjugate fluid-porous domains using general unstructured grids. *Numerical heat transfer. Part B, Fundamentals*, 60(4):252–277.

Delestre, O., Lucas, C., Ksinant, P.-A., Darboux, F., Laguerre, C., Vo, T. N. T., James, F., and Cordier, S. (2013). SWASHES: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for Numerical Methods in Fluids*, 72(3):269–300.

Deshpande, S. S., Anumolu, L., and Trujillo, M. F. (2012). Evaluating the performance of the two-phase flow solver interFoam. *Computational Science and Discovery*, 5(1):014016.

Dixon, S. J., Sear, D. A., Odoni, N. A., Sykes, T., and Lane, S. N. (2016). The effects of river restoration on catchment scale flood risk and flood hydrology. *Earth Surface Processes and Landforms*, 41(7):997–1008.

Drew, D. A. (1983). Mathematical modeling of two-phase flow. *Annual Review of Fluid Mechanics*, 15(1):261–291.

Drikakis, D. and Smolarkiewicz, P. K. (2001). On spurious vortical structures. *Journal of Computational Physics*, 172(1):309–325.

Drikakis, D. D. and Rider, W. (2005). *High-Resolution Methods for Incompressible and Low-Speed Flows*. Computational fluid and solid mechanics. Springer-Verlag, New York.

Dudley, S. J., Fischenich, J. C., and Abt, S. R. (1998). Effect of woody debris entrapment on flow resistance. *JAWRA Journal of the American Water Resources Association*, 34(5):1189–1197.

Dumbser, M. and Toro, E. (2011). A simple extension of the Osher Riemann solver to non-conservative hyperbolic systems. *Journal of Scientific Computing*, 48(1):70–88.

Duru, A. (2014). Numerical modelling of contracted sharp crested weirs. Master's thesis, Middle East Technical University.

Echeverribar, I., Morales-Hernández, M., Brufau, P., and García-Navarro, P. (2019). Use of internal boundary conditions for levees representation: application to river flood management. *Environmental Fluid Mechanics*.

Economon, T. D. (2020). Simulation and adjoint-based design for variable density incompressible flows with heat transfer. *AIAA Journal*, 58(2):757–769.

Efremov, V. R., Kozelkov, A. S., Kornev, A. V., Kurkin, A. A., Kurulin, V. V., Strelets, D. Y., and Tarasova, N. V. (2017). Method for taking into account gravity in free-surface flow simulation. *Computational Mathematics and Mathematical Physics*, 57(10):1720–1733.

Elman, H. C., Howle, V. E., Shadid, J. N., and Tuminaro, R. S. (2003). A parallel block multi-level preconditioner for the 3D incompressible Navier–Stokes equations. *Journal of Computational Physics*, 187(2):504—523.

Elsworth, D. T. and Toro, E. F. (1992). Riemann solvers for solving the incompressible Navier-Stokes equations using the artificial compressibility method. NASA STI/Recon Technical Report N.

Environment Agency (2012). Greater working with natural processes in flood and coastal erosion risk management—a response to Pitt review recommendation 27. Technical report, Environment Agency.

Environment Agency (2018). Beavers to return to Essex for the first time in 400 years. `https://www.gov.uk/government/news/beavers-to-return-to-essex-for-the-first-time-in-400-years`. Accessed: 17 January 2019.

Evstigneev, N. (2008). Integration of 3D incompressible free surface Navier-Stokes equations on unstructured tetrahedral grid using distributed computation on TCP/IP networks. In Rahman, M. and Brebbia, C., editors, *Advances in Fluid Mechanics VII*, WIT Transactions on Engineering Sciences, pages 65–77. WIT.

Evtushok, G. Y., Boiko, A. V., Yakovenko, S. N., Takovenko, E. E., and Chang, K. C. (2021). Modification and verification of numerical algorithms for dam-break flow over a horizontal bed. *Journal of Applied Mechanics and Technical Physics*, 62:255–261.

Ferguson, C. and Fenner, R. (2020a). Evaluating the effectiveness of catchment-scale approaches in mitigating urban surface water flooding. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2168):20190203.

Ferguson, C. and Fenner, R. (2020b). How natural flood management helps downstream urban drainage in various storm directions. *Proceedings of the Institution of Civil Engineers—Water Management*, pages 1–21.

Fernández-Pato, J., Morales-Hernández, M., and García-Navarro, P. (2018). Implicit finite volume simulation of 2D shallow water flows in flexible meshes. *Computer Methods in Applied Mechanics and Engineering*, 328:1–25.

Ferzinger, J. H. and Perić, M. (2002). *Computational Methods for Fluid Dynamics*. Springer, 3rd edition.

Flora, K. (2012). Setting BCs for Riverine Flows using Interfoam [post 17]. `https://www.cfd-online.com/Forums/openfoam-solving/97987-setting-bcs-riverine-flows-using-interfoam.html`. Accessed: 29 April 2020.

Follett, E., Schalko, I., and Nepf, H. (2020). Momentum and energy predict the backwater rise generated by a large wood jam. *Geophysical Research Letters*, 47(17):e2020GL089346.

Follett, E., Schalko, I., and Nepf, H. (2021). Logjams with a lower gap: Backwater rise and flow distribution beneath and through logjam predicted by two-box momentum balance. *Geophysical Research Letters*. e2021GL094279.

Forbes, H., Ball, K., and McLay, F. (2015). Natural flood management handbook. Technical report, Scottish Environmental Protection Agency.

Forest Research (2007). The Robinwood Robinflood report: Evaluation of large woody debris in watercourses. Technical report, Forest Research.

Franzini, F., Hoedenaeken, D., and Soares-Frazão, S. (2018). Modeling the flow around islands in rivers using a one-dimensional approach. In Gourbesville, P., Cunge, J., and Caignaert, G., editors, *Advances in Hydroinformatics*, pages 127–143. Springer.

Friedrich, H., Ravazzolo, D., Ruiz-Villanueva, V., Schalko, I., Spreitzer, G., Tunnicliffe, J., and Weitbrecht, V. (2021). Physical modelling of large wood (LW) processes relevant for river management: Perspectives from New Zealand and Switzerland. *Earth Surface Processes and Landforms*, pages 1–26.

Fuchs, F., McMurry, A., Mishra, S., Risebro, N., and Waagan, K. (2010). High order well-balanced finite volume schemes for simulating wave propagation in stratified magnetic atmospheres. *Journal of Computational Physics*, 229(11):4033–4058.

Gallisdorfer, M. S., Bennett, S. J., Atkinson, J. F., Ghaneeizad, S. M., Brooks, A. P., Simon, A., and Langendoen, E. J. (2014). Physical-scale model designs for engineered log jams in rivers. *Journal of Hydro-environment Research*, 8(2):115–128.

Gao, F., Ingram, D. M., Causon, D. M., and Mingham, C. G. (2007). The development of a Cartesian cut cell method for incompressible viscous flows. *International Journal for Numerical Methods in Fluids*, 54(9):1033–1053.

Gao, F., Ingram, D. M., Causon, D. M., and Mingham, C. G. (2009). Numerical modelling of wave interaction with porous structures. In Zhuang, F. G. and Li, J. C., editors, *New Trends in Fluid Mechanics Research*, pages 514–517, Berlin, Heidelberg. Springer Berlin Heidelberg.

Glenis, V., Kutija, V., and Kilsby, C. (2018). A fully hydrodynamic urban flood modelling system representing buildings, green space and interventions. *Environmental Modelling and Software*, 109:272–292.

Godunov, S. K. (1959). A difference scheme for numerical computation of discontinuous solutions of equations of fluid dynamics. *Matematičeskij Sbornik*, 47(89):271–306.

Gopala, V. R. and van Wachem, B. G. M. (2008). Volume of fluid methods for immiscible-fluid and free-surface flows. *Chemical Engineering Journal*, 141(1):204–221.

Goudarzi, S., Milledge, D. G., Holden, J., Evans, M. G., Allott, T. E. H., Shuttleworth, E. L., Pilkington, M., and Walker, J. (2021). Blanket peat restoration: Numerical study of the underlying processes delivering natural flood management benefits. *Water Resources Research*, 57(4):e2020WR029209.

Gove, M. (2018). No such thing as too much information—the science and politics of climate change. `https://www.gov.uk/government/speeches/michael-gove-speech-on-uk-climate-change-projections`. Accessed: 27 November 2018.

Grabowski, R. C., Gurnell, A. M., Burgess-Gamble, L., England, J., Holland, D., Klaar, M. J., Morrissey, I., Uttley, C., and Wharton, G. (2019). The current state of the use of large wood in river restoration and management. *Water and Environment Journal*, 33:366–377.

Greenshields, C. (2020). OpenFOAM v8 user guide. `https://cfd.direct/openfoam/user-guide`. Accessed: 9 September 2020.

Greenshields, C. and Weller, H. (2022). *Notes on Computational Fluid Dynamics: General Principles*. CFD Direct Ltd, Reading.

Gregory, K. J., Gurnell, A. M., and Hill, C. T. (1985). The permanence of debris dams related to river channel processes. *Hydrological Sciences Journal*, 30(3):371–381.

Gschaider, B. (2011). interFoam Average velocity of water only! [post 2]. `https://www.cfd-online.com/Forums/openfoam-post-processing/92180-interfoam-average-velocity-water-only.html`. Accessed: 29 April 2020.

Gschaider, B. (2014). [swak4Foam] Averaging along an axis.. swak4foam? [post 3]. https://www.cfd-online.com/Forums/openfoam-community-contributions/142505-averaging-along-axis-swak4foam.html. Accessed: 29 April 2020.

Guerrero, J. (2019). Finite volume method: a crash introduction. Technical report, Wolf Dynamics.

Guillard, H. and Viozat, C. (1999). On the behaviour of upwind schemes in the low Mach number limit. *Computers and Fluids*, 28(1):63–86.

Hafs, A. W., Harrison, L. R., Utz, R. M., and Dunne, T. (2014). Quantifying the role of woody debris in providing bioenergetically favorable habitat for juvenile salmon. *Ecological Modelling*, 285:30–38.

Hankin, B. (2020). Natural flood management Eddleston Water modelling. Seminar, CIWEM.

Hankin, B., Burgess-Gamble, L., Bentley, S., and Rose, S. (2016). How to model and map catchment processes when flood risk management planning. Technical report, Environment Agency.

Hankin, B., Metcalfe, P., Beven, K., and Chappell, N. A. (2019). Integration of hillslope hydrology and 2D hydraulic modelling for natural flood management. *Hydrology Research*, 50(6):1535–1548.

Hankin, B., Page, T. J. C., Chappell, N. A., Beven, K. J., Smith, P. J., Kretzschmar, A., and Lamb, R. (2021). Using micro-catchment experiments for multi-local scale modelling of nature-based solutions. *Hydrological Processes*, 35(11):e14418.

Hart, J., Rubinato, M., and Lavers, T. (2020). An experimental investigation of the hydraulics and pollutant dispersion characteristics of a model beaver dam. *Water*, 12(9):2320.

Harten, A., Lax, P. D., and van Leer, B. (1983). On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61.

Hashemi, M. R., Ryzhakov, P. B., and Rossi, R. (2020). An enriched finite element/level-set method for simulating two-phase incompressible fluid flows with surface tension. *Computer Methods in Applied Mechanics and Engineering*, 370:113277.

Henry, H. R. (1950). Discussion on 'Diffusion of submerged jets'. *Transactions of the American Society of Civil Engineers*, 115:639–697.

Hewett, C., Mark E. Wilkinson, M., Jonczyk, J., and Quinn, P. (2020). Catchment systems engineering: An holistic approach to catchment management. *WIREs Water*, e1417.

Hino, T. (1999). An interface capturing method for free surface flow computations on unstructured grids. *Journal of The Society of Naval Architects of Japan*, 1999(186):177–183.

Hirt, C. (2014). CFD-101. `https://www.flow3d.com/resources/cfd-101/`. Accessed: 10 September 2020.

Hirt, C. W. and Nichols, B. D. (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225.

Hodges, B. R. (2020). An artificial compressibility method for 1D simulation of open-channel and pressurized-pipe flow. *Water*, 12(6):1727.

Holzmann, T. (2019). *Mathematics, Numerics, Derivations and OpenFOAM*. Holzmann CFD.

Hu, Z. Z., Causon, D. M., Mingham, C. G., and Qian, L. (2013). A Cartesian cut cell free surface capturing method for 3D water impact problems. *International Journal for Numerical Methods in Fluids*, 71(10):1238–1259.

Hyman, J. M. (1984). Numerical methods for tracking interfaces. *Physica D: Nonlinear Phenomena*, 12(1):396–407.

Iacob, O., Rowan, J., Brown, I., and Ellis, C. (2014). Evaluating wider benefits of natural flood management strategies: an ecosystem-based adaptation perspective. *Hydrology Research*, 45(6):774–787.

Ingram, D., Gao, F., Causon, D., Mingham, C., and Troch, P. (2009). Numerical investigations of wave overtopping at coastal structures. *Coastal Engineering*, 56(2):190–202.

Innovyze (2021). Blockages. `https://help2.innovyze.com/infoworksicm/Content/HTML/ICM_ILCM/Blockage.htm`. Accessed: 25 January 2021.

Iserles, A. (2008). *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2nd edition.

Issa, R. (1986). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65.

Jaafar, H. H. and Merkley, G. P. (2010). High-resolution method for modeling hydraulic regime changes at canal gate structures. *Journal of Irrigation and Drainage Engineering*, 136(12):795–808.

Jacobs (2021). Blockage unit. `https://help.floodmodeller.com/floodmodeller/Technical_Reference/1D_Nodes_Reference/Blockage_unit.htm`. Accessed: 25 January 2021.

Jameson, A. (1991). Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. In *AIAA 10th Computational Fluid Dynamics Conference*, Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics.

Jamshed, S. (2015). *Using HPC for Computational Fluid Dynamics—A Guide to High Performance Computing for CFD Engineers*. Elsevier.

Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, Imperial College London.

Jasak, H. (2012). Handling parallelisation in OpenFOAM. In *Cyprus Advanced HPC Workshop*, Nicosia.

Jasak, H. (2014). dbnsfoam, foam-extend 4.0. `http://www.foam-extend.org`. Accessed: 20 May 2022.

Jasak, H., Weller, H., and Gosman, A. (1999). High resolution NVD differencing scheme for arbitrarily unstructured meshes. *International Journal for Numerical Methods in Fluids*, 31(2):431–449.

Kajzer, A. and Pozorski, J. (2020). A weakly compressible, diffuse-interface model for two-phase flows. *Flow, Turbulence and Combustion*, 105(2):299–333.

Kelecy, F. and Pletcher, R. (1997). The development of a free surface capturing approach for multidimensional free surface flows in closed containers. *Journal of Computational Physics*, 138(2):939–980.

Ketabdari, M. J. (2016). Free surface flow simulation using VOF method. In Lopez-Ruiz, R., editor, *Numerical Simulation*. IntechOpen.

Keys, T. A., Govenor, H., Jones, C. N., Hession, W. C., Hester, E. T., and Scott, D. T. (2018). Effects of large wood on floodplain connectivity in a headwater Mid-Atlantic stream. *Ecological Engineering*, 118:134–142.

Kitts, D. (2010). *The Hydraulic and Hydrological Performance of Large Wood Accumulation in a Low-order Forest Stream*. PhD thesis, University of Southampton.

Krause, G. (2019). Hydrostatic equilibrium preservation in MHD numerical simulation with stratified atmospheres: Explicit Godunov-type schemes with MUSCL reconstruction. *Astronomy and Astrophysics (Berlin)*, 631:A68.

Kruisbrink, A., Pearce, F., Yue, T., and Morvan, H. (2018). An SPH multi-fluid model based on quasi buoyancy for interface stabilization up to high density ratios and realistic wave speed ratios. *International Journal for Numerical Methods in Fluids*, 87(10):487–507.

Kwak, D. and Kiris, C. (2011). *Computation of Viscous Incompressible Flows*. Scientific computation. Springer, New York.

Kärrholm, F. P. and Tao, F. (2008). On performance of advection schemes in the prediction of diesel spray and fuel vapour distributions. In *22nd European Conference on Liquid Atomization and Spray Systems*, Como Lake, Italy.

Lai, Y. G. and Bandrowski, D. J. (2014). Large wood flow hydraulics: A 3D modelling approach. In *7th International Congress on Environmental Modelling and Software*, San Diego.

Lane, S. N. (2017). Natural flood management. *WIREs Water*, 4(3):e1211.

Launder, B. and Spalding, D. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269–289.

Leakey, S. (2018). Developing a numerical model of a leaky barrier in a flume. Master's thesis, Newcastle University.

Leakey, S., Glenis, V., and Hewett, C. J. (2022a). Artificial compressibility with Riemann solvers: Convergence of limiters on unstructured meshes. *OpenFOAM Journal*, 2:31–47.

Leakey, S., Glenis, V., and Hewett, C. J. (2022b). A novel Godunov-type scheme for free-surface flows with artificial compressibility. *Computer Methods in Applied Mechanics and Engineering*, 393:114763.

Leakey, S., Hewett, C. J. M., Glenis, V., and Quinn, P. F. (2020). Modelling the impact of leaky barriers with a 1D Godunov-type scheme for the shallow water equations. *Water*, 12(2).

Lee, B. J., Toro, E. F., Castro, C. E., and Nikiforakis, N. (2013). Adaptive Osher-type scheme for the Euler equations with highly nonlinear equations of state. *Journal of Computational Physics*, 246:165–183.

LeVeque, R. J. (2004). *Finite-Volume Methods for Hyperbolic Problems*. Cambridge University Press.

L'Hommedieu, W., Tullos, D., and Jones, J. (2020). Effects of an engineered log jam on spatial variability of the flow field across submergence depths. *River Research and Applications*, 36(3):383–397.

Li, G., Bhatia, D., and Wang, J. (2020). Compressive properties of min-mod-type limiters in modelling shockwave-containing flows. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 42(6):290.

Liang, Q. and Marche, F. (2009). Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in Water Resources*, 32(6):873–884.

Lin, C., Yen, J., and Tsai, C. (2002). Influence of sluice gate contraction coefficient on distinguishing condition. *Journal of Irrigation and Drainage Engineering*, 128(4):249–252.

Linstead, C. and Gurnell, A. (1999). Large woody debris in British headwater rivers: physical habitat role and management guidelines. Technical Report W185, Environment Agency.

Liu, X.-D., Osher, S., and Chan, T. (1994). Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212.

Lo, H. W., Smith, M., Klaar, M., and Woulds, C. (2021). Potential secondary effects of in-stream wood structures installed for natural flood management: A conceptual model. *WIREs Water*, 8(5):e1546.

Loppi, N. A. (2019). *High-Order Incompressible Computational Fluid Dynamics on Modern Hardware Architectures*. PhD thesis, Imperial College London.

Luo, M., Koh, C. G., Gao, M., and Bai, W. (2015). A particle method for two-phase flows with large density difference. *International Journal for Numerical Methods in Engineering*, 103(4):235–255.

Lv, X., Zou, Q., Reeve, D., and Wang, Z. (2008). An Unstructured 3D LES Solver For Free Surface Flow And Breaking Waves. In *International Ocean and Polar Engineering Conference*, Vancouver.

Ma, Z., Qian, L., Causon, D., and Mingham, C. (2011). Simulation of solitary breaking waves using a two-fluid hybrid turbulence approach. In *International Ocean and Polar Engineering Conference*, Maui.

Macchione, F. and Lombardo, M. (2021). Roughness-based method for simulating hydraulic consequences of both woody debris clogging and breakage at bridges in basin-scale flood modeling. *Water Resources Research*, 57(12). e2021WR030485.

Manga, M. and Kirchner, J. (2000). Stress partitioning in streams by large woody debris. *Water Resources Research*, 36(8):2373–2379.

Manners, R. B., Doyle, M. W., and Small, M. J. (2007). Structure and hydraulics of natural woody debris jams. *Water Resour. Res.*, 43(6).

Manzanero, J., Rubio, G., Kopriva, D. A., Ferrer, E., and Valero, E. (2020). An entropy–stable discontinuous Galerkin approximation for the incompressible Navier–Stokes equations with variable density and artificial compressibility. *Journal of Computational Physics*, 408:109241.

Marek, M., Aniszewski, W., and Bogusławski, A. (2008). Simplified volume of fluid method (SVOF) for two-phase flows. *Task Quarterly*, 12(3):255–265.

Marić, T., Marschall, H., and Bothe, D. (2013). voFoam—a geometrical volume of fluid algorithm on arbitrary unstructured meshes with local dynamic adaptive mesh refinement using OpenFOAM. Archived at: `https://arxiv.org/abs/1305.3417`.

Marić, T., Marschall, H., and Bothe, D. (2015). lentFoam—a hybrid level set/front tracking method on unstructured meshes. *Computers and Fluids*, 113(C):20–31.

Maricar, M. F. and Maricar, F. (2020). Flume experiments on woody debris accumulation at the bridge pier during flood. *IOP Conference Series: Earth and Environmental Science*, 575:012188.

Martin, J. C. and Moyce, W. J. (1952). Part IV. An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 244(882):312–324.

Massa, F. C., Bassi, F., Botti, L., and Colombo, A. (2020). An implicit high-order discontinuous Galerkin approach for variable density incompressible flows. In *Droplet Interactions and Spray Processes*, volume 121 of *Fluid Mechanics and Its Applications*, pages 191–202. Springer International Publishing, Cham.

Menter, F. and Esch, T. (2001). Elements of industrial heat transfer predictions. In *16th Brazilian Congress of Mechanical Engineering*, volume 20, pages 117–127.

Menter, F., Kuntz, M., and Langtry, R. B. (2003). Ten years of industrial experience with the SST turbulence model. *Heat and Mass Transfer*, 4:625–632.

Menter, F. R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32(8):1598–1605.

Metcalfe, P., Beven, K., and Freer, J. (2015). Dynamic TOPMODEL: A new implementation in R and its sensitivity to time and space steps. *Environmental Modelling and Software*, 72(C):155–172.

Metcalfe, P., Beven, K., Hankin, B., and Lamb, R. (2017). A modelling framework for evaluation of the hydrological impacts of nature-based approaches to flood risk management, with application to in-channel interventions across a 29-km^2 scale catchment in the United Kingdom. *Hydrological Processes*, 31(9):1734–1748.

Metcalfe, P., Beven, K., Hankin, B., and Lamb, R. (2018). A new method, with application, for analysis of the impacts on flood risk of widely distributed enhanced hillslope storage. *Hydrology and Earth System Sciences*, 22:2589–2605.

Metcalfe, P., Hankin, B., Page, T., Johnson, D., Craigen, I., and Chappell, N. (2016). Rivers Trust Life-IP project: Strategic investigation of natural flood management in Cumbria. Technical report, JBA Consulting.

Michalak, K. and Ollivier-Gooch, C. (2008). Limiters for unstructured higher-order accurate solutions of the Euler equations. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno.

Milledge, D., Odoni, N., Allott, R., Evans, M., Pilkington, M., and Walker, J. (2015). Annex 6. Flood risk modelling. In Pilkington, M., editor, *Restoration of Blanket Bogs; Flood Risk Reduction and Other Ecosystem Benefits*. Moors for the Future Partnership.

Mirjalili, S., Jain, S. S., and Dodd, M. S. (2017). Interface-capturing methods for two-phase flows: an overview and recent developments. *Center for Turbulence Research Annual Research Briefs*, pages 117–135.

Morales-Hernández, M., Murillo, J., and García-Navarro, P. (2013). The formulation of internal boundary conditions in unsteady 2-D shallow water flows: Application to flood regulation. *Water Resources Research*, 49(1):471–487.

Morales-Hernández, M., Hubbard, M., and García-Navarro, P. (2014). A 2D extension of a large time step explicit scheme (CFL>1) for unsteady problems with wet/dry boundaries. *Journal of Computational Physics*, 263:303–327.

Moukalled, F., Mangani, L., and Darwish, M. (2016). *The Finite Volume Method in Computational Fluid Dynamics*. Springer.

Muhawenimana, V., Follett, E., and Wilson, C. (2021). Flow dynamics, accretion and sediment transport of instream wood barriers. In *AGU Fall Meeting 2021*. EP51A-03.

Muhawenimana, V., Wilson, C. A., Nefjodova, J., and Cable, J. (2020). Flood attenuation hydraulics of channel-spanning leaky barriers. *Journal of Hydrology*. 125731.

Mukha, T., Almeland, S. K., and Bensow, R. E. (2020). LES of a classical hydraulic jump: Influence of modelling parameters on the predictive accuracy. Archived at: `https://arxiv.org/abs/2007.01729`.

Müller, S., Wilson, C. A. M. E., Ouro, P., and Cable, J. (2021a). Experimental investigation of physical leaky barrier design implications on juvenile rainbow trout (Oncorhynchus mykiss) movement. *Water Resources Research*. e2021WR030111.

Müller, S., Wilson, C. A. M. E., Ouro, P., and Cable, J. (2021b). Leaky barriers: leaky enough for fish to pass? *Royal Society Open Science*, 8(3):201843.

Negm, A.-A. M., Al-Brahim, A. M., and Alhamid, A. A. (2002). Combined-free flow over weirs and below gates. *Journal of Hydraulic Research*, 40(3):359–365.

Negm, A.-A. M., Al-Brahim, A. M., Alhamid, A. A., Altan Sakarya, B., Aydin, I., and Ger, A. M. (2004). Combined-free flow over weirs and below gates. *Journal of Hydraulic Research*, 42(5):559–562.

Nesshöver, C., Assmuth, T., Irvine, K. N., Rusch, G. M., Waylen, K. A., Delbaere, B., Haase, D., Jones-Walters, L., Keune, H., Kovacs, E., Krauze, K., Külvik, M., Rey, F., van Dijk, J., Vistad, O. I., Wilkinson, M. E., and Wittmer, H. (2017). The science, policy and practice of nature-based solutions: An interdisciplinary perspective. *Science of the Total Environment*, 579(C):1215–1227.

Ngai, R., Wilkinson, M., Nisbet, T., Harvey, R., Addy, S., Burgess-Gamble, L., Rose, S., Maslen, S., Nicholson, A., Page, T., Jonczyk, J., and Quinn, P. (2017). Working with natural processes evidence directory: Literature review. Technical report, Environment Agency.

Nicholls, D. and Hankin, B. (2015). Nature-based approaches for catchment flood management. Technical report, JBA Trust.

Nicholson, A. R., Wilkinson, M. E., O' Donnell, G. M., and Quinn, P. F. (2012). Runoff attenuation features: A sustainable flood mitigation strategy in the Belford catchment, UK. *Area*, 44(4):463–469.

Nisbet, T., Roe, P., Marrington, S., Thomas, H., Broadmeadow, S., and Valatin, G. (2015). Defra FCERM multi-objective flood management demonstration project. Technical report, Defra.

Nisbet, T. and Thomas, H. (2008). Restoring floodplain woodland for flood alleviation. Technical report, Defra.

Noh, W. F. and Woodward, P. (1976). SLIC (simple line interface calculation). In van de Vooren, A. I. and Zandbergen, P. J., editors, *Proceedings of the Fifth International*

Conference on Numerical Methods in Fluid Dynamics, pages 330–340. Springer Berlin Heidelberg.

Norbury, M., Phillips, H., Macdonald, N., Brown, D., Boothroyd, R., Wilson, C., Quinn, P., and Shaw, D. (2021). Quantifying the hydrological implications of pre- and post-installation willowed engineered log jams in the pennine uplands, NW England. *Journal of Hydrology*, page 126855.

Norbury, M., Shaw, D., and Jones, P. (2018). Combining hydraulic modelling with partnership working: Towards a practical natural flood management approach. *Proceedings of the Institute of Civil Engineers: Engineering Sustainability*, pages 1–43.

Nourgaliev, R., Dinh, T., and Theofanous, T. (2004). A pseudocompressibility method for the numerical simulation of incompressible multifluid flows. *International Journal of Multiphase Flow*, 30(7):901–937.

Ntouras, D. and Papadakis, G. (2020). A coupled artificial compressibility method for free surface flows. *Journal of Marine Science and Engineering*, 8(8):590.

Nyssen, J., Pontzeele, J., and Billi, P. (2011). Effect of beaver dams on the hydrology of small mountain streams: Example from the Chevral in the Ourthe Orientale basin, Ardennes, Belgium. *Journal of Hydrology*, 402(1):92–102.

Odini, N. A. and Lane, S. N. (2010). Assessment of the impact of upstream land management measures on flood flows in Pickering Beck using OVERFLOW. Technical report, Durham University.

Okamoto, T., Takebayashi, H., Sanjou, M., Suzuki, R., and Toda, K. (2019). Log jam formation at bridges and the effect on floodplain flow: A flume experiment. *Journal of Flood Risk Management*, 13(Suppl. 1):e12562.

Olsen, N. (2015). Four free surface algorithms for the 3D Navier-Stokes equations. *Journal of Hydroinformatics*, 17(6):845–856.

OpenCFD (2019a). OpenFOAM programmer's guide. Technical report, ESI.

OpenCFD (2019b). OpenFOAM user guide. Technical report, ESI.

Osher, S. and Solomon, F. (1982). Upwind difference schemes for hyperbolic systems of conservation laws. *Mathematics of Computation*, 38(158):339–374.

Oxtoby, O. F., Malan, A. G., and Heyns, J. A. (2015). A computationally efficient 3D finite-volume scheme for violent liquid-gas sloshing. *International Journal for Numerical Methods in Fluids*, 79(6):306–321.

Pan, D. and Chang, C. (2000). The capturing of free surfaces in incompressible multi-fluid flows. *International Journal for Numerical Methods in Fluids*, 33(2):203–222.

Park, J. S. and Kim, C. (2012). Multi-dimensional limiting process for finite volume methods on unstructured grids. *Computers and Fluids*, 65:8–24.

Park, J. S., Yoon, S.-H., and Kim, C. (2010). Multi-dimensional limiting process for hyperbolic conservation laws on unstructured grids. *Journal of Computational Physics*, 229(3):788–812.

Patankar, S. and Spalding, D. (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806.

Patankar, S. V. (1980). *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation.

Patil, A. (2018). Numerical investigation of the effect of nappe non-aeration on caisson sliding force during tsunami breakwater over-topping using OpenFOAM. Master's thesis, TU Delft.

Pattison, I., Lane, S. N., Hardy, R. J., and Reaney, S. M. (2014). The role of tributary relative timing and sequencing in controlling large floods. *Water Resources Research*, 50(7):5444–5458.

Pearson, E. G. (2020). *Modelling the Interactions between Geomorphological Processes and Natural Flood Management*. PhD thesis, University of Leeds.

Pelanti, M. (2017). Low Mach number preconditioning techniques for Roe-type and HLLC-type methods for a two-phase compressible flow model. *Applied Mathematics and Computation*, 310:112–133.

Pepe, V., Cimorelli, L., Pugliano, G., Morte, R. D., Pianese, D., and Cozzolino, L. (2019). The solution of the Riemann problem in rectangular channels with constrictions and obstructions. *Advances in Water Resources*, 129:146–164.

Perry, B., Rennie, C., Cornett, A., and Knox, P. (2018). Comparison of large woody debris prototypes in a large scale non-flume physical model. In *River Flow 2018—Ninth International Conference on Fluvial Hydraulics*, Lyon-Villeurbanne.

Pieri, R. (2014). OpenFOAM selected solver. `https://materials.prace-ri.eu/388/3/OpenFOAMselectedSolver-1.pdf`. Accessed: 28 February 2022.

Pinto, C., Ing, R., Browning, B., Delboni, V., Wilson, H., Martyn, D., and Harvey, G. L. (2019). Hydromorphological, hydraulic and ecological effects of restored wood: findings and reflections from an academic partnership approach. *Water and Environment Journal*, 33(3):353–365.

Piton, G., Horiguchi, T., Marchal, L., and Lambert, S. (2020). Open check dams and large wood: head losses and release conditions. *Natural Hazards and Earth System Sciences.*, 20(12):3293–3314.

Pitt, M. (2008). The Pitt review: Lessons learned from the 2007 floods. Technical report, Independent Chair.

Pope, S. B. (2000). *Turbulent Flows*. Cambridge University Press, Cambridge.

Powell, G., Clark, J., and Nisbet, T. (2021). Monitoring leaky barriers for natural flood management (NFM) within a community-led project. In *EGU General Assembly 2021*. EGU21-7508.

Puttock, A., Graham, H. A., Ashe, J., Luscombe, D. J., and Brazier, R. E. (2020). Beaver dams attenuate flow: A multi-site study. *Hydrological Processes*. e14017.

Puttock, A., Graham, H. A., Cunliffe, A. M., Elliott, M., and Brazier, R. E. (2017). Eurasian beaver activity increases water storage, attenuates flow and mitigates diffuse pollution from intensively-managed grasslands. *Science of the Total Environment*, 576:430–443.

Qian, L., Causon, D. M., Mingham, C. G., and Ingram, D. M. (2006). A free-surface capturing method for two fluid flows with moving bodies. *Proceedings: Mathematical, Physical and Engineering Sciences*, 462(2065):21–42.

Queutey, P. and Visonneau, M. (2007). An interface capturing method for free-surface hydrodynamic flows. *Computers and Fluids*, 36(9):1481–1510.

Quinn, P., O'Donnell, G., Nicholson, A., Wilkinson, M., Owen, G., Jonczyk, J., Barber, N., Mardwick, M., and Davies, G. (2013). Potential use of runoff attenuation features in small rural catchments for flood mitigation. Technical report, Newcastle University.

Rahman, M. M. and Siikonen, T. (2008). An artificial compressibility method for viscous incompressible and low mach number flows. *International Journal for Numerical Methods in Engineering*, 75(11):1320–1340.

Rasche, D., Reinhardt-Imjela, C., Schulte, A., and Wenzel, R. (2019). Hydrodynamic simulation of the effects of stable in-channel large wood on the flood hydrographs of a low mountain range creek, Ore Mountains, Germany. *Hydrology and Earth System Sciences*, 23(10):4349–4365.

Rider, W. J. and Kothe, D. B. (1998). Reconstructing volume tracking. *Journal of Computational Physics*, 141(2):112–152.

Roberts, S., Nielsen, O., Gray, D., Sexton, J., and Davies, G. (2019). ANUGA user manual release 2.0.3. Technical report, Geoscience Australia.

Roe, P. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372.

Roenby, J., Bredmose, H., and Jasak, H. (2016). A computational method for sharp interface advection. *Royal Society Open Science*, 3(11):160405.

Roenby, J., Bredmose, H., and Jasak, H. (2018). IsoAdvector: Geometric VOF on general meshes. In Nóbrega, J. and Jasak, H., editors, *OpenFOAM—Selected papers of the 11th Workshop*. Springer International Publishing.

Roenby, J., Larsen, B., Bredmose, H., and Jasak, H. (2017). A new volume-of-fluid method in OpenFOAM. In M. Visonneau, P. Q. and Touzé, D. L., editors, *7th International Conference on Computational Methods in Marine Engineering (Marine 2017)*, Nantes.

Rudman, M. (1997). Volume-tracking methods for interfacial flow calculations. *International Journal for Numerical Methods in Fluids*, 24(7):671–691.

Ruiz-Villanueva, V. and Stoffel, M. (2017). Frederick J. Swanson's 1976–1979 papers on the effects of instream wood on fluvial processes and instream wood management. *Progress in Physical Geography*, 41(1):124–133.

Rusche, H. (2002). *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. PhD thesis, Imperial College London.

Samani, J. M. V. and Mazaheri, M. (2009). Combined flow over weir and under gate. *Journal of Hydraulic Engineering*, 135(3):224–227.

Sambe, A. N., Golay, F., Sous, D., Fraunié, P., Rey, V., Marcer, R., and De Jouette, C. (2011). Two-phase-flow unstructured grid solver: Application to tsunami wave impact. *International Journal of Offshore and Polar Engineering*, 21(03).

Samra, A. P. (2017). The effects of natural flood management in rural catchments: A research and hydraulic modelling study applied within the river thames catchment in the united kingdom. Master's thesis, UNESCO-IHE.

Scardovelli, R. and Zaleski, S. (1999). Direct numerical simulation of free-surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31(1):567–603.

Scardovelli, R. and Zaleski, S. (2003). Interface reconstruction with least-square fit and split Eulerian-Lagrangian advection. *International Journal for Numerical Methods in Fluids*, 41(3):251–274.

Schalko, I., Ruiz-Villanueva, V., Maager, F., and Weitbrecht, V. (2021). Wood retention at inclined bar screens: Effect of wood characteristics on backwater rise and bedload transport. *Water*, 13(16).

Schalko, I., Schmocker, L., Weitbrecht, V., and Boes, R. M. (2018). Backwater rise due to large wood accumulations. *Journal of Hydraulic Engineering*, 144(9).

Sear, D. A., Millington, C. E., Kitts, D. R., and Jeffries, R. (2010). Logjam controls on channel:floodplain interactions in wooded catchments and their role in the formation of multi-channel patterns. *Geomorphology*, 116(3-4):305–319.

Sepulveda, C., Gomez, M., and Rodellar, J. (2009). Benchmark of discharge calibration methods for submerged sluice gates. *Journal of Irrigation and Drainage Engineering*, 135(5):676–682.

Sethian, J. and Smereka, P. (2003). Level set methods for fluid interfaces. *Annual Review of Fluid Mechanics*, 35:341–372.

Shapiro, E. and Drikakis, D. (2005a). Artificial compressibility, characteristics-based schemes for variable density, incompressible, multi-species flows. Part I. Derivation of different formulations and constant density limit. *Journal of Computational Physics*, 210(2):584–607.

Shapiro, E. and Drikakis, D. (2005b). Artificial compressibility, characteristics-based schemes for variable-density, incompressible, multispecies flows: Part II. Multigrid implementation and numerical tests. *Journal of Computational Physics*, 210(2):608–631.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University.

Shields, F. D. and Gippel, C. (1995). Prediction of effects of woody debris removal on flow resistance. *Journal of Hydraulic Engineering*, 121(4):341–354.

Shin, S. (2013). Simulation of two-dimensional internal waves generated by a translating and pitching foil. *Ocean Engineering*, 72:77–86.

Shin, S., Bae, S. Y., Kim, I. C., Kim, Y. J., and Yoon, H. K. (2012). Simulation of free surface flows using the flux-difference splitting scheme on the hybrid Cartesian/immersed boundary method. *International Journal for Numerical Methods in Fluids*, 68(3):360–376.

Sholtes, J. and Doyle, M. (2011). Effect of channel restoration on flood wave attenuation. *Journal of Hydraulic Engineering*, 137(2):196–208.

Shuttleworth, E. L., Evans, M. G., Pilkington, M., Spencer, T., Walker, J., Milledge, D., and Allott, T. E. H. (2019). Restoration of blanket peat moorland delays stormflow from hillslopes and reduces peak discharge. *Journal of Hydrology X*, 2:100006.

Slow the Flow Calderdale (2022). Hardcastle Crags. `http://slowtheflow.net/hardcastle-crags/`. Accessed: 26 January 2022.

Slowing the Flow Partnership (2016). Slowing the flow partnership briefing: Boxing Day 2015 flood event. Technical report, Forest Research.

Spekreijse, S. (1987). Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws. *Mathematics of Computation*, 49(179):135–155.

Spreitzer, G., Tunnicliffe, J., and Friedrich, H. (2020). Porosity and volume assessments of large wood (LW) accumulations. *Geomorphology*, 358:107122.

Starkey, E., Walsh, C., Quinn, P., Stott, K., Jonczyk, J., and Smith, L. (2020). Using dense monitoring networks in urban environments: National Green Infrastructure Facility 'swale'. In *Environmental Monitoring: Meeting Evidence Needs*, Manchester. UKEOF.

Swamee, P. (1992). Sluice-gate discharge equations. *Journal of Irrigation and Drainage Engineering*, 118(1):56–60.

Sweby, P. K. (1984). High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011.

Tabor, G. (2018). A detailed look at fvSchemes and fvSolution. In *13th OpenFOAM Workshop*, Shanghai.

Tanaka, T. (1994). Finite volume TVD scheme on an unstructured grid system for three-dimensional MHD simulation of inhomogeneous systems including strong background potential fields. *Journal of Computational Physics*, 111(2):381–389.

Taylor, M. and Clarke, L. E. (2021). Monitoring the impact of leaky barriers used for natural flood management on three river reaches in the Stroud Frome and Twyver catchments, Gloucestershire, UK. Project report, University of Gloucestershire.

Tennekes, H. and Lumley, J. (1972). *A First Course in Turbulence*. MIT Press, Cambridge, Massachusetts.

Thames21 (2021). Natural flood management—leaky woody barrier installation guide. Technical report.

Thomas, H. and Nisbet, T. (2012). Modelling the hydraulic impact of reintroducing large woody debris into watercourses. *Journal of Flood Risk Management*, 5(2):164–174.

Thomas, H. and Nisbet, T. R. (2007). An assessment of the impact of floodplain woodland on flood flows. *Water and Environment Journal*, 21(2):114–126.

Thomas, H. and Nisbet, T. R. (2016). Slowing the flow in Pickering: Quantifying the effect of catchment woodland planting on flooding using the soil conservation service curve number method. *International Journal of Safety and Security Engineering*, 6(3):466–474.

Toro, E., Spruce, M., and Speares, W. (1994). Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, 4:25–34.

Toro, E. F. (2001). *Shock-Capturing Methods for Free-Surface Shallow Flows*. John Wiley.

Toro, E. F. (2009). *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer.

Tuoi, V. (2008). One-dimensional Saint-Venant system. Master's thesis, University of Orleans.

Turkel, E. (1993). Review of preconditioning methods for fluid dynamics. *Applied Numerical Mathematics*, 12(1):257–284.

Ubbink, O. (1997). *Numerical Prediction of Two Fluid Systems with Sharp Interfaces*. PhD thesis, Imperial College London.

Valverde, R. (2013). Roughness and geometry effects of engineered log jams on 1-D flow characteristics. Master's thesis, Oregon State University.

van Leer, B. (1976). MUSCL, a new approach to numerical gas dynamics. In *Second European Conference on Computational Physics*.

Venkatakrishnan, V. (1995). Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, 118(1):120–130.

Vermeire, B., Loppi, N., and Vincent, P. (2019). Optimal Runge-Kutta schemes for pseudo time-stepping with high-order unstructured methods. *Journal of Computational Physics*, 383:55–71.

Versteeg, H. K. and Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Prentice Hall, 2nd edition.

Villemonte, J. R. (1947). Submerged weir discharge studies. *Engineering News-Record*, 139(26):54–56.

Vukčević, V., Roenby, J., Gatin, I., and Jasak, H. (2018). A sharp free surface finite volume method applied to gravity wave flows. Archived at: `https://arxiv.org/abs/1804.01130`.

Wallerstein, N., Alonso, C., Bennett, S., and Thorne, C. (2001). Distorted Froude-scaled flume analysis of large woody debris. *Earth Surface Processes and Landforms*, 26(12):1265–1283.

Wang, W.-H., Du, Y.-Z., Wang, L.-L., Wang, Y.-Y., and Huang, Y. (2020). Novel numerical method to simulate hydrodynamic characteristic of moving body through free surface and stratified-fluid interface. *Ocean Engineering*, 205:107234.

Wang, W.-H. and Wang, Y.-Y. (2009). An improved free surface capturing method based on Cartesian cut cell mesh for water-entry and -exit problems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2106):1843–1868.

Wang, W.-H. and Wang, Y.-Y. (2010a). Calculation of water entry problem for free-falling bodies using a developed Cartesian cut cell mesh. *AIP Conference Proceedings*, 1233(1):590–595.

Wang, W.-H. and Wang, Y.-Y. (2010b). An essential solution of water entry problems and its engineering applications. *Journal of Marine Science and Application*, 9:268–273.

Wang, W.-H. and Wang, Y.-Y. (2012). Analysis of mechanical energy transport on free-falling wedge during water-entry phase. *Journal of Applied Mathematics*, 738082.

Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631.

Wenzel, R., Reinhardt-Imjela, C., Schulte, A., and Bölscher, J. (2014). The potential of in-channel large woody debris in transforming discharge hydrographs in headwater areas (Ore Mountains, Southeastern Germany). *Ecological Engineering*, 71(C):1–9.

Westbrook, C. J., Ronnquist, A., and Bedard-Haughn, A. (2020). Hydrological functioning of a beaver dam sequence and regional dam persistence during an extreme rainstorm. *Hydrological Processes*, 34(18):3726–3737.

Weston, B. (2000). A marker and cell solution of the incompressible Navier-Stokes equations for free surface flow. Technical report.

Wilcox, D. C. (1988). Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, 26(11):1299–1310.

Wilcox, D. C. (2006). *Turbulence modeling for CFD*. DCW Industries, La Cañada, California, 3rd edition.

Wilhelmsen, K., Sawyer, A. H., Marshall, A., McFadden, S., Singha, K., and Wohl, E. (2021). Laboratory flume and numerical modeling experiments show log jams and branching channels increase hyporheic exchange. *Water Resources Research*, 57(9). e2021WR030299.

Wilkinson, M. E., Quinn, P. F., and Welton, P. (2010). Runoff management during the September 2008 floods in the Belford catchment, Northumberland. *Journal of Flood Risk Management*, 3(4):285–295.

Wimshurst, A. (2019). [CFD] The PISO algorithm. `https://www.youtube.com/watch?v=ahdW5TKacok`. Accessed: 28 February 2022.

Woessner, W. W. (2020). *Groudwater-Surface Water Exchange*. Groundwater Project.

Wohl, E. (2017). Bridging the gaps: An overview of wood across time and space in diverse rivers. *Geomorphology*, 279:3–26.

Wolstenholme, J., Skinner, C., Milan, D., and Parsons, D. (2021). Geomorphological numerical modelling of woody dams in CAESAR-Lisflood. In *EGU General Assembly 2021*. EGU21-9636.

Woolfson, M. M. (2007). Practical SPH models for major planets. *Monthly Notices of the Royal Astronomical Society*, 376(3):1173–1181.

Wu, Y., Wang, W.-H., Huang, Y., and Wang, Y.-Y. (2014). Further application of surface capturing method and Cartesian cut cell mesh on hydroelastic water-entry problems of free-falling elastic wedge. *Mathematical Problems in Engineering*, 545642.

Xu, Y. and Liu, X. (2017). Effects of different in-stream structure representations in computational fluid dynamics models-taking engineered log jams (ELJ) as an example. *Water*, 9(2).

Yeoh, G. H. and Tu, J. (2010). *Computational Techniques for Multiphase Flows: Basics and Applications*. Butterworth-Heinemann.

Youngs, D. (1982). Time-dependent multi-material flow with large fluid distortion. In Morton, K. N. and Baines, M. J., editors, *Numerical Methods for Fluid Dynamics*, volume 24, pages 273–285. Academic Press.

Zhao, D. H., Shen, H. W., Tabios, G. Q., Lai, J. S., and Tan, W. Y. (1994). Finite-volume 2-dimensional unsteady-flow model for river basins. *Journal of Hydraulic Engineering*, 120(7):863–883.

Zhao, Y., Hui Tan, H., and Zhang, B. (2002). A high-resolution characteristics-based implicit dual time-stepping VOF method for free surface flow simulation on unstructured grids. *Journal of Computational Physics*, 183(1):233–273.

Zhou, L., Cai, Z. W., Zong, Z., and Chen, Z. (2016). An SPH pressure correction algorithm for multiphase flows with large density ratio. *International Journal for Numerical Methods in Fluids*, 81(12):765–788.

Zingale, M., Dursi, L. J., ZuHone, J., Calder, A. C., Fryxell, B., Plewa, T., Truran, J. W., Caceres, A., Olson, K., Ricker, P. M., Riley, K., Rosner, R., Siegel, A., Timmes, F. X., and Vladimirova, N. (2002). Mapping initial hydrostatic models in Godunov codes. *The Astrophysical Journal Supplement Series*, 143(2):539–565.

Appendix A. Flume experiments

The flume experiments were conducted in the Novak Lab at Newcastle University, currently home to two hydraulic flumes of different widths. The same kind of barrier was modelled in both the narrow and wide flumes: a simple horizontal barrier as illustrated in Figure A.1.

A.1 Narrow flume

This older flume has width 0.294m, length 17.95m, adjustable slope fixed to 1:1600 for these experiments, and Manning's n of 0.009. A metal leaky barrier was placed at $x = 11.14$m along the length of the flume, using slots on either side of the flume to fix it in place, as shown in Figure A.2. These slots reduced the barrier width to 0.264m. Let the depth of the base of the leaky barrier be a_0 and the top a_1, as in Figure A.1. Four barrier configurations were tested, each for a range of steady-state flow conditions:

(a) No barrier, 1 experiment;

(b) ($a_0 = 0.05$m, $a_1 = 0.15$m), 7 free flow experiments;

(c) ($a_0 = 0.05$m, $a_1 = 0.1$m), 15 free flow experiments; and

(d) ($a_0 = 0.025$m, $a_1 = 0.125$m), 30 free flow and 2 submerged experiments.

With reference to Figure A.1, "free flow" describes stages 1 and 3, or scenarios where the leaky barrier is not touched, and "submerged" describes stages 2, 4 and 5, where the downstream tailwater affects the flow through the barrier.

Depth measurements were taken with a point gauge at six fixed points along the flume, two upstream of the barrier and four downstream: $x = 10, 11, 11.3, 12, 13, 17.8$m. For

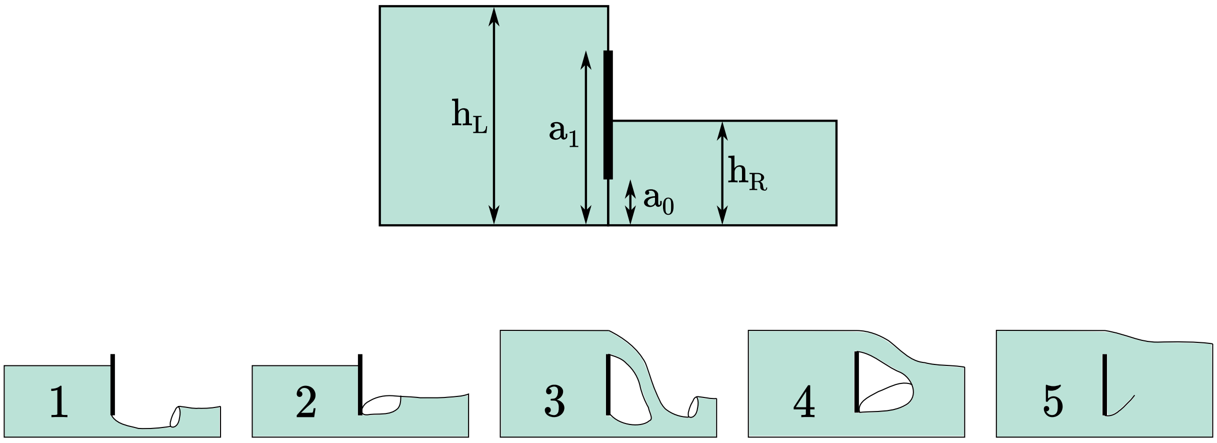

Figure A.1: The leaky barrier model: definition sketch and five conceptual operating stages, repeated from Figure 2.2.

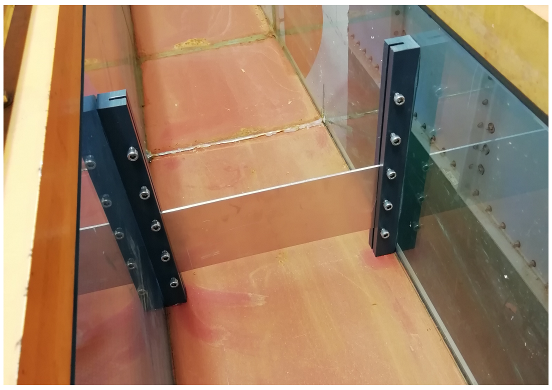

Figure A.2: Leaky barrier model in the narrow flume.

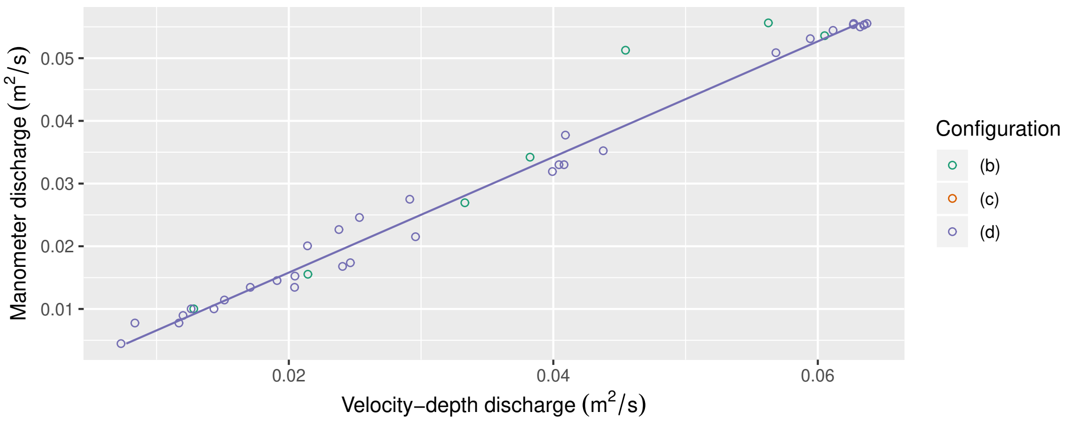

Figure A.3: The relationship between velocity-depth and manometer discharges, with regression for (d).

(c) and (d), two additional depth measurements were taken for the free-flow experiments, one either side of the downstream hydraulic jump.

For each of the 55 flume experiments, the steady-state discharge was measured using a manometer (q_{man}). For (a), (b) and (d), a velocity meter was also used to calculate the discharge (q_{vel}). It was found that the manometer and velocity-meter discharges did not agree. The equation

$$q_{vel} = 0.00286 + 1.08 \cdot q_{man} \tag{A.1}$$

describes the relationship for (d) with $R^2 = 0.99$, as plotted in Figure A.3. In case (c), there were no q_{vel} data. However, the velocity meter was known to give more accurate results than the manometer. Therefore, for consistency, the regression equation (A.1) was used to generate q_{vel} values from measured values of q_{man} for all cases. This transformed discharge is assumed to be an unbiased measurement in the project.

Tables A.1–A.2 summarise the 55 experiments in the narrow flume.

A.2 Wide flume

This newer flume has width 1.0m, length 12.0m, and an adjustable slope fixed to 1:50 for the experiment. A plastic leaky barrier ($a_0 = 0.035$m, $a_1 = 0.235$m) was slotted in at $x = 5.0$m along the length of the flume, shown in Figure A.4. However, unlike in the narrow flume, the slots were more streamlined, built into the sides of the flume.

Experiment	Configuration	q_{man} (m^2/s)	q_{vel} (m^2/s)	q (m^2/s)	$h_{17.8}$ (m)
N1	b	0.01002	0.01281	0.014	0.0340
N2	b	0.01554	0.02144	0.020	0.0430
N3	b	0.02694	0.03331	0.032	0.0550
N4	b	0.05127	0.04546	0.058	0.0610
N5	b	0.03421	0.03824	0.040	0.0585
N6	b	0.05359	0.06051	0.061	0.0750
N7	b	0.05563	0.05626	0.063	0.0845
N8	a	0.05617	0.05921	0.064	0.0770
N9	c	0.00633	-	0.010	0.0340
N10	c	0.01487	-	0.019	0.0365
N11	c	0.01737	-	0.022	0.0415
N12	c	0.01930	-	0.024	0.0450
N13	c	0.02289	-	0.028	0.0490
N14	c	0.02656	-	0.032	0.0515
N15	c	0.03144	-	0.037	0.0550
N16	c	0.03594	-	0.042	0.0595
N17	c	0.03956	-	0.046	0.0610
N18	c	0.04393	-	0.051	0.0650
N19	c	0.04661	-	0.053	0.0650
N20	c	0.04966	-	0.057	0.0710
N21	c	0.05145	-	0.059	0.0710
N22	c	0.05340	-	0.061	0.0740
N23	c	0.05517	-	0.063	0.0715
N24	d	0.01002	0.01434	0.014	0.0375
N25	d	0.01345	0.02044	0.017	0.0445
N26	d	0.01678	0.02406	0.021	0.0470
N27	d	0.01737	0.02465	0.022	0.0480

Table A.1: Summary of experiments N1–N27 in the narrow flume—configuration (a, b, c, or d), manometer discharge, velocity-meter discharge, transformed discharge, and depth at $x = 17.8$m.

Experiment	Configuration	q_{man} (m^2/s)	q_{vel} (m^2/s)	q (m^2/s)	$h_{17.8}$ (m)
N28	d	0.02152	0.02958	0.026	0.0525
N29	d	0.03523	0.04378	0.041	0.0620
N30	d	0.05535	0.06352	0.063	0.0745
N31	d	0.05554	0.06271	0.063	0.0830
N32	d	0.05526	0.06347	0.063	0.0690
N33	d	0.05554	0.06372	0.063	0.0695
N34	d	0.05535	0.06269	0.063	0.0710
N35	d	0.05499	0.06320	0.063	0.0690
N36	d	0.05443	0.06115	0.062	0.0670
N37	d	0.05311	0.05943	0.060	0.0705
N38	d	0.05087	0.05683	0.058	0.0685
N39	d	0.03773	0.04092	0.044	0.0595
N40	d	0.02750	0.02915	0.033	0.0490
N41	d	0.02459	0.02534	0.030	0.0460
N42	d	0.02267	0.02379	0.027	0.0430
N43	d	0.02007	0.02141	0.025	0.0420
N44	d	0.01524	0.02046	0.019	0.0415
N45	d	0.01453	0.01911	0.019	0.0390
N46	d	0.01345	0.01709	0.017	0.0370
N47	d	0.01143	0.01513	0.015	0.0345
N48	d	0.01002	0.01261	0.014	0.0325
N49	d	0.00896	0.01200	0.013	0.0310
N50	d	0.00776	0.01169	0.011	0.0290
N51	d	0.00776	0.00836	0.011	0.0245
N52	d	0.00447	0.00731	0.008	0.0240
N53	d	0.03192	0.03993	0.037	0.0590
N54	d	0.03301	0.04081	0.039	0.1725
N55	d	0.03301	0.04043	0.039	0.2440

Table A.2: Summary of experiments N28–N55 in the narrow flume—configuration (a, b, c, or d), manometer discharge, velocity-meter discharge, transformed discharge, and depth at $x = 17.8$m.

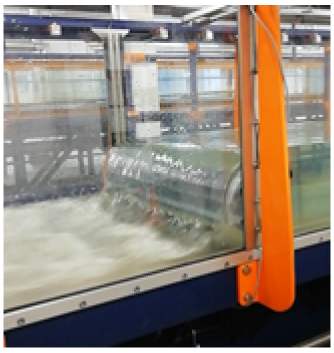

Figure A.4: Leaky barrier model in the wide flume.

Experiment	a_0	a_1	q_{min} (m^2/s)	q_{max} (m^2/s)	$h_{9.0}$ (m)
W	0.035	0.235	0.0632	0.0639	0.0435

Table A.3: Summary of experiment W in the wide flume—leaky barrier dimensions, minimum and maximum discharge, and depth at $x = 9.0$m.

Consequently, even though the barrier was operating in stage 3, the nappe over the top of the barrier was not aerated.

Depth measurements were taken with a point gauge at ten points along the flume, five upstream and five downstream: $x = 2, 3, 4, 4.5, 5, 5.02, 6, 7, 8, 9$m. As the supercritical flow downstream of the barrier was very turbulent, the four most downstream measurements were taken twice, one at each bound of the fluctuating depth. Unlike the narrow flume, the wide flume automatically measures the flow rate in the pipe between the pump and the flume.

Table A.3 summarises the experiment in the wide flume.

Appendix B. OpenFOAM case files

This appendix presents selected OpenFOAM case files used in Chapter 4.

B.1 fvSchemes

B.1.1 Initial stage (quick, stable)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1906                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default                localEuler;
}

gradSchemes
{
    default                cellLimited Gauss linear 1;
}

divSchemes
```

```
{
    default             none;

    div(rhoPhi,U)       Gauss upwind;
    div(phi,k)          Gauss upwind;
    div(phi,omega)      Gauss upwind;

    div(phi,alpha)      Gauss vanLeer;
    div(phirb,alpha)    Gauss linear;

    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default             Gauss linear limited 0.5;
}

interpolationSchemes
{
    default             linear;
}

snGradSchemes
{
    default             limited 0.5;
}

wallDist
{
    method              meshWave;
}

// ************************************************************************* //
```

B.1.2 Final stage (accurate)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1906                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default             Euler;
}

gradSchemes
{
    default             cellLimited Gauss linear 1;
}

divSchemes
{
    default             none;

    div(rhoPhi,U)       Gauss linearUpwind default;
    div(phi,k)          Gauss linearUpwind default;
    div(phi,omega)      Gauss linearUpwind default;

    div(phi,alpha)      Gauss vanLeer;
    div(phirb,alpha)    Gauss linear;

    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default             Gauss linear corrected;
}

interpolationSchemes
{
    default             linear;
}
```

```

snGradSchemes
{
    default                 corrected;
}

wallDist
{
    method                  meshWave;
}

// ************************************************************************* //
```

B.2 fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1906                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        cAlpha          1;
        nLimiterIter    2;
        nAlphaSubCycles 1;
        nAlphaCorr      3;
        alphaApplyPrevCorr  no;
        MULESCorr       yes;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0;
    }

    "pcorr.*"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-5;
        relTol          0;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "(U|k|omega).*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0.01;
    }
}

PIMPLE
{
    momentumPredictor   no;
    nCorrectors         3;
    nOuterCorrectors    1;
    nNonOrthogonalCorrectors 1;
    pRefCell            0;
    pRefValue           0;
```

```
}

// ************************************************************************* //
```