Utrecht University

Master thesis
Mathematical Sciences

Second order velocity reconstruction on unstructured grids

Eveline Visee

Supervised by
Prof. dr. J.E. Frank (UU)
F.W. Platzek (Deltares)
M. Borsboom (Deltares)

May 31, 2016

Abstract

The new shallow water solver D-Flow FM is being developed at the Deltares research institute. It employs a discretization method on unstructured staggered grids, which requires reconstructing a velocity vector in the center of the cell. The currently employed first order method as developed by Perot forces the use of fine and/or regular grids for the modelling of advection and diffusion. To take full advantage of unstructured grids, a second order velocity vector reconstruction method is required. In this thesis, I will derive several second order methods: corrections for Perot, explicit least squares and a hybrid method. We derive discretization methods for advection and diffusion, study all methods in a simplified model in Matlab and determine their convergence behaviour. One of the second order velocity reconstruction methods is also studied in the shallow water solver D-Flow FM. Second order velocity reconstruction turns out to be better than Perot's method but it needs a second order advection method for the best results.

Contents

1 Introduction

1.1 Background

The Shallow Water Equations have numerous applications: if they are used to simulate rivers and seas, it becomes possible to predict floods, study sedimentation and the distribution of nutrients and other chemicals, and, together with atmospheric equations, they are a part of climate models. Accurate simulation software can contribute to the safety of the Netherlands and other coastal plains, river deltas and millions of people. Multiple software packages have been developed to simulate hydrological systems, such as

unTRIM: a semi-implicit finite difference model for the shallow water equations developed by Casulli et al.[5], which is an unstructured-grid version of TRIM (Tidal, Residual, Intertidal Mudflat), developed for modelling in coastal areas. It does not use Perot's reconstruction, but rather the formulation from [6], and the advection term is in Eulerian-Lagrangian form.

SUNTANS (Stanford Unstructured Nonhydrostatic Terrain-following Adaptive Navier-Stokes Simulator), as developed by Fringer [8] et al., is a finite volume solver for coastal oceans. It employs Perot's reconstruction on a Delaunay triangulation and a structured vertical discretization (z-level). Since it works on the Navier Stokes equations rather than the Shallow Water Equations, it is not easy to compare with the current work.

Adcirc: (advanced circulation: adcirc.org) is a package using a finite element method, both available in full-3D version and depth-integrated 2D.

Delft3D: The current shallow water flow solver by the Deltares research institute (Delft, the Netherlands) is the Delft3D software package. It uses curvilinear orthogonal structured grids, a staggered discretization scheme and an implicit time integration scheme. The structured grids cause problems when discretization irregular boundaries and obtaining local grid refinements. Though it uses domain decomposition, the lack of grid flexibility is the main reason for the development of D-Flow FM.

D-Flow FM: Most important for this research is D-Flow Flexible Mesh, which is a successor for Delft3D and uses unstructured grids [13]. It solves the depth-integrated two-dimensional Shallow Water Equations. The workings of D-Flow FM are explained in detail in the following chapters, but the essence is that it solves the Shallow Water Equations on an orthogonal staggered grid with a semi-implicit discretization that can be regulated with the θ-method. The advection term, which is non-linear in the velocity, makes use of a reconstruction at the cell center (Perot's method), which is first-order accurate (but second order accurate on regular grids).

1.2 Motivation and aim

Partial Differential Equation solvers working on square or rectangular grids have been around for a long time, but problems arise if the domain boundary does not align with the grid. Staggered grids are used to decouple the pressure and velocity components, while curvilinear and unstructured grids allow for a domain (especially on the boundary) to be discretized more accurately and add refinements more easily. However, unstructured grids come with their own problems, such as low-order accuracy of many numerical methods, and it is these problems I will address in this thesis.

The choice of the particular discretization in D-Flow FM means we need to reconstruct the velocity vector in the cell center from the normal component of the velocity at the faces of the cell. This is straightforward for rectangular grids but unstructured grids call for more sophisticated methods of velocity reconstruction. Perot's method does well as long as the grid is fairly regular (orthogonal, and the distance between the centroid and the circumcenter is small). Perot's method is, in general, first order accurate, but second order on regular grids. Finding second-order methods, either as correction for Perot or independently, may improve the simulations significantly. We will derive the second order velocity reconstructions, discuss their mathematical properties, and assess their performance (accuracy of water level, discharge, velocity and its gradient, and advection and diffusion), first in a simplified model and later in D-Flow FM.

Second order methods are considered by Vidovic [20], in the form of least squares approximations, Peixoto and Barros [15], who compare different methods, and Boscheri [3].

Vidovic's least squares algorithm reconstructs the velocity at the nodes of triangles, while we will later discuss reconstruction at the circumcenters of triangles. The idea, however, is the same.

Peixoto and Barros [15] consider a velocity reconstruction method with radial basis functions, a method with finite elements and discuss Perot's method (using either normal or tangential velocity components) and least squares methods. They then map several grids to a sphere, analyzing the errors from the velocity reconstruction methods and the mapping on top of it. They conclude that Perot's scheme has the highest L^∞ errors but is comparable to the other methods in L^2 error, though it is still the least accurate.

Boscheri [3] uses a Taylor expansion around the circumcenter of a polygon, using the known velocity on the edges of the polygon to find the velocity in the circumcenter. This expression is normalized and preconditioned using singular value decomposition, and then the overdetermined method is solved using a least squares algorithm. Then the Navier Stokes equations are integrated with a semi-implicit staggered finite volume scheme.

The purpose of this thesis will be to find a general second order velocity reconstruction method that is more accurate than Perot's velocity reconstruction, and does not have a negative effect on the kinetic energy (the energy is dissipated due to the upwind parts in the discretization, but a velocity reconstruction can negate that, causing instabilities. This is outside the scope of this thesis).

2 Governing equations

Fluid dynamics is governed by conservation of mass, momentum and energy. Assuming that the fluid is a continuous medium, physical properties such as velocity and density can be described in a time-dependent fashion. In the Lagrangian formulation, each particle has its physical properties specified as a function of time. In Eulerian formulation, the history of physical properties at each fixed point of the domain is specified.

The Navier-Stokes equations form the basis of computational fluid dynamics. Derived from Newton's laws of motion and the conservation laws of mass and momentum, they describe the movement of a body of water [22].

2.1 The Navier Stokes equations

The mass conservation law states that within a system, the total amount of fluid (water) in a control volume stays the same. The total mass in a control volume Ω can be computed by integrating the density $\rho = \frac{\text{mass}}{\text{volume}}$ over the volume, and the rate of change can be computed by taking its time derivative. This has to be equal to the net flux integrated over the boundary $\partial\Omega$, hence we get

$$\frac{d}{dt}\int_\Omega \rho \, dV = -\int_{\partial\Omega} \rho\vec{u}\cdot\vec{n}\, dA \tag{1}$$

with n the outward normal of the boundary. We use $\vec{u} = (u, v, w)$ for the velocity of the flow in the (x, y, z)-directions. Applying Gauss' theorem to rewrite the right hand side and assuming that the density is a smooth function over Ω we get $-\int_{\partial\Omega} \rho\vec{u}\cdot\vec{n}\, dA = -\int_\Omega \nabla\cdot\rho\vec{u}\, dV$ and we can swap the derivative and integral to obtain the continuity equation in its most general form:

$$\int_\Omega \frac{\partial\rho}{\partial t} + \nabla\cdot\rho\vec{u}\, dV = 0 \tag{2}$$

Since Ω was arbitrary, and assuming there are no shocks or discontinuities, we can drop the integral sign and get the continuity equation $\frac{\partial\rho}{\partial t} + \nabla\cdot\rho\vec{u} = 0$. For incompressible flows, the density will remain the same throughout time and the continuity equation simplifies to $\nabla\cdot\vec{u} = 0$.

Momentum is the product of mass and velocity, so for a volume Ω this is $\int_\Omega \rho\vec{u}\, dV$. To conserve momentum, we need to balance the momentum flux over $\partial\Omega$ with the body forces (such as gravity) and the external stresses like wind. The gravity is given by the gravity acceleration $g = 9.81m/s^2$ times the density, and other body forces such as the Coriolis forces can be introduced at the same part in the equations. The external forces are made up of wind stress, bottom friction and pressure, which we will write as $pI_3 + T$, where the matrix T contains the stress terms we use later and I_3 the 3×3 identity matrix. This gives the momentum balance

$$\frac{d}{dt}\int_\Omega \rho\vec{u}\, dV = -\int_{\partial\Omega} (\rho\vec{u})\vec{u}\cdot\vec{n}\, dA + \int_\Omega \rho g(0,0,1)^T\, dV + \int_{\partial\Omega} (pI_3 + T)\vec{n}\, dV \tag{3}$$

Again, Ω is arbitrary so we can drop the integral sign if there are no shocks or discontinuities. We divide by ρ and we get the momentum equation

$$\frac{d}{dt}\begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} \nabla\cdot u\vec{u} \\ \nabla\cdot v\vec{u} \\ \nabla\cdot w\vec{u} \end{pmatrix} = -\frac{1}{\rho}\nabla\cdot\Big(pI_3 + T\Big) + \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \tag{4}$$

Assuming incompressibility, constant temperature and constant salinity, ρ is constant.

2.2 Shallow water equations

We can derive the shallow water equations (SWE) by depth-integrating the Navier-Stokes equations. The free surface is denoted by ζ, which is the z-coordinate of the water surface dependent on x, y and t. The free surface equation is

$$\zeta_t + u^s \zeta_x + v^s \zeta_y = w^s \tag{5}$$

since vertical flow depends on horizontal flow and change rate of the height (the superscript s is for "surface").

At the bottom $z = -b$ (the total height of the water column becomes $h = \zeta + b$) we assume no movement at the bottom $u = v = 0$ and no pressure at the surface: $p = 0$. We write the bottom shear stress as

$$(\tau_{bx}, \tau_{by}) = (\tau_{xx}\frac{\partial b}{\partial x} + \tau_{xy}\frac{\partial b}{\partial y} + \tau_{xz}, \tau_{yx}\frac{\partial b}{\partial x} + \tau_{yy}\frac{\partial b}{\partial y} + \tau_{yz})$$

and the surface shear stress (this includes wind) as

$$(\tau_{sx}, \tau_{sy}) = (-\tau_{xx}\frac{\partial \zeta}{\partial x} - \tau_{xy}\frac{\partial \zeta}{\partial y} + \tau_{xz}, -\tau_{yx}\frac{\partial \zeta}{\partial x} - \tau_{yy}\frac{\partial \zeta}{\partial y} + \tau_{yz}).$$

There is no normal flow at the bottom, hence the bottom boundary equation is

$$u^b b_x + v^b b_y + w^b = 0 \tag{6}$$

as the bathymetry does not depend on time, and the vertical flow depends only on horizontal flow, bathymetry and height changes (the superscript b is for "bottom"). The free surface equation and bottom boundary equation form the kinematic boundary conditions:

$$u\Big|_b \frac{\partial b}{\partial x} + v\Big|_b \frac{\partial b}{\partial y} + w\Big|_b = 0, \qquad \frac{\partial \zeta}{\partial t} + u\Big|_\zeta \frac{\partial \zeta}{\partial x} + v\Big|_\zeta \frac{\partial \zeta}{\partial y} - w\Big|_\zeta = 0. \tag{7}$$

Note that we assume that the bottom topography of the water body (such as the river bed, sea floor or ocean floor: the bathymetry) does not change in time, which implies

$$\frac{\partial h}{\partial t} = \frac{\partial (b + \zeta)}{\partial t} = \frac{\partial b}{\partial t} + \frac{\partial \zeta}{\partial t} = 0 + \frac{\partial \zeta}{\partial t} = \frac{\partial \zeta}{\partial t}. \tag{8}$$

Now we can integrate the continuity equation (eq. 2) from $z = -b$ to $z = \zeta$, using depth-averaged velocities

$$\bar{u} = \frac{1}{h}\int_{-b}^{\zeta} u\, dz, \qquad \bar{v} = \frac{1}{h}\int_{-b}^{\zeta} v\, dz$$

and applying the kinematic boundary conditions gives (note that the w and z components of $\vec{u}$ and $\vec{x}$ get integrated out, so $\vec{u}$, $\vec{x}$ and the operator ∇ are three-dimensional, while the subscript H denotes their two-dimensional horizontal versions). We use Leibniz' integral rule to change the

order of differentiation and integration:

$$
\begin{aligned}
0 &= \int_{-b}^{\zeta} \nabla \cdot \vec{u}\, dz && (9)\\
&= \nabla \cdot \int_{-b}^{\zeta} \vec{u}\, dz - \big(u\Big|_{\zeta}\frac{\partial \zeta}{\partial x} + u\Big|_{-b}\frac{\partial b}{\partial x}\big) - \big(v\Big|_{\zeta}\frac{\partial \zeta}{\partial y} + v\Big|_{-b}\frac{\partial b}{\partial y}\big) + w\Big|_{\zeta} - w\Big|_{-b} && (10)\\
&= \nabla_H \cdot \big(h\bar{u} + h\bar{v}\big) - \big(u\Big|_{\zeta}\frac{\partial \zeta}{\partial x} + u\Big|_{-b}\frac{\partial b}{\partial x}\big) - \big(v\Big|_{\zeta}\frac{\partial \zeta}{\partial y} + v\Big|_{-b}\frac{\partial b}{\partial y}\big)\\
&\quad + \frac{\partial \zeta}{\partial t} + u\Big|_{\zeta}\frac{\partial \zeta}{\partial x} + v\Big|_{\zeta}\frac{\partial \zeta}{\partial y} - u\Big|_{-b}\frac{\partial b}{\partial x} - v\Big|_{-b}\frac{\partial b}{\partial y} && (11)\\
&= \frac{\partial}{\partial x}(h\bar{u}) + \frac{\partial}{\partial y}(h\bar{v}) + \frac{\partial \zeta}{\partial t} && (12)
\end{aligned}
$$

The momentum equation (eq. 4) in vertical direction collapses to $\frac{\partial p}{\partial z} = \rho g$, as the vertical motion scales are small with respect to the horizontal motion scales, and integrating this over z gives the hydrostatic pressure term $p = \rho g(\zeta - z)$. Then $\frac{\partial p}{\partial x} = \rho g \frac{\partial \zeta}{\partial x}$ and $\frac{\partial p}{\partial y} = \rho g \frac{\partial \zeta}{\partial y}$ and we can use this when integrating the $x-$ and $y-$ momentum equations over depth. The left hand side of the x-momentum equation becomes

$$
\int_{-b}^{\zeta} \frac{\partial u}{\partial t} + \nabla u \vec{u}\, dz = \frac{\partial}{\partial t}(h\bar{u}) + \frac{\partial}{\partial x}(h\bar{u}^2) + \frac{\partial}{\partial y}(h\bar{u}\bar{v}) + \text{ differential advection terms} \tag{13}
$$

and analogous for y. The differential advection terms (abbreviated DAT) originate from the fact that the average of a product of two functions is generally not the same as the product of the averages. For the right hand side of the x-momentum equation, we have

$$
\int_{-b}^{\zeta} -\frac{1}{\rho}\frac{\partial}{\partial x}\Big(p + T\Big)\, dz = \int_{-b}^{\zeta} -\frac{1}{\rho}\Big(\rho g h \frac{\partial \zeta}{\partial x} + \frac{\partial T}{\partial x}\Big)\, dz = -gh\frac{\partial \zeta}{\partial x} - \frac{h}{\rho}\Big(\frac{\partial T}{\partial x}\Big|_{\zeta} - \frac{\partial T}{\partial x}\Big|_{-b} + F_x\Big) \tag{14}
$$

and analogous for y. Here F_x, F_y can include external forces such as the Coriolis force $\sin(\phi) h\Omega \times \vec{u}$ where ϕ indicates the latitude, Ω the earth rotational speed, and only the third coordinate is non-zero. Hence we get the two-dimensional Shallow Water equations

$$
\begin{aligned}
\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x}(h\bar{u}) + \frac{\partial}{\partial y}(h\bar{v}) &= 0 && (15)\\
\frac{\partial}{\partial t}(h\bar{u}) + \frac{\partial}{\partial x}(h\bar{u}^2) + \frac{\partial}{\partial y}(h\bar{u}\bar{v}) + \text{ DAT} &= -gh\frac{\partial \zeta}{\partial x} - \frac{h}{\rho}\Big(\frac{\partial T}{\partial x}\Big|_{\zeta} - \frac{\partial T}{\partial x}\Big|_{-b} + F_x\Big) && (16)\\
\frac{\partial}{\partial t}(h\bar{v}) + \frac{\partial}{\partial x}(h\bar{u}\bar{v}) + \frac{\partial}{\partial y}(h\bar{v}^2) + \text{ DAT} &= -gh\frac{\partial \zeta}{\partial y} - \frac{h}{\rho}\Big(\frac{\partial T}{\partial y}\Big|_{\zeta} - \frac{\partial T}{\partial y}\Big|_{-b} + F_y\Big) && (17)
\end{aligned}
$$

The stress term $\frac{h}{\rho}\Big(\frac{\partial}{\partial(x,y)}T\Big|_{\zeta} - \frac{\partial}{\partial(x,y)}T\Big|_{-b} + \vec{F}\Big)$ will be equal to $g\frac{|\vec{u}|\vec{u}}{C^2}$, which represents bottom shear stress. Wind and other stresses will be modeled separately where needed. The term C is the Chézy coefficient in $\sqrt{m}/s$, which describes the roughness of the bottom, we drop the averages again, and get

$$
\frac{\partial \zeta}{\partial t} + \nabla_H \cdot (h\vec{u}_H) = 0, \qquad \frac{\partial}{\partial t}(h\vec{u}_H) + \nabla_H \cdot (h\vec{u}_H\vec{u}_H) = -gh\nabla_H \zeta_H + g\frac{|\vec{u}|\vec{u}}{C^2}. \tag{18}
$$

From here on, all vectors will be two-dimensional, so we drop the subscript H.

Sometimes we will use the momentum equation in non-conservative form (as this form is the one discretized in D-Flow FM [18]); first expand the time derivative

$$\frac{\partial hu}{\partial t} = h\frac{\partial u}{\partial t} + u\frac{\partial h}{\partial t} \qquad \Rightarrow \qquad \frac{\partial u}{\partial t} = \frac{1}{h}\frac{\partial hu}{\partial t} - \frac{1}{h}u\frac{\partial h}{\partial t}$$

(and analogous for v) and substitute in the momentum equation to get

$$\frac{\partial \vec{u}}{\partial t} + \frac{1}{h}\Big(\nabla\cdot(h\vec{u}\vec{u}) - \vec{u}\nabla\cdot(h\vec{u})\Big) + \frac{1}{h}\mathrm{DAT} = -g\nabla\zeta + g\frac{|\vec{u}|\vec{u}}{h\,C^2}. \tag{19}$$

The term $\nabla\cdot(h\vec{u}\vec{u}) = (\frac{\partial huu}{\partial x}+\frac{\partial hvu}{\partial y}, \frac{\partial huv}{\partial x}+\frac{\partial hvv}{\partial y})$ in the momentum equation represents advection, which indicates the rate with which a certain property (in this case: momentum) is transported by the flow.

3 Unstructured staggered grids

This chapter will deal with the necessary properties of the spatial grid before we move on to the discretization of the Shallow Water Equations.

3.1 Finite Volume Discretization

The general idea of the FVD is to integrate the continuity equation over the cell boundary, using Gauss' divergence theorem

$$\iiint_V \nabla \cdot \vec{F}\, dV = \oiint_S \vec{F} \cdot \vec{n}\, dS \tag{20}$$

for some field $\vec{F}$. This means that the flow leaving one cell is gained by the adjoining cell, leading to conservation of mass in the inner cells of the domain, and mass can only change via the boundary conditions. Conservation of momentum and kinetic energy are difficult to prove on irregular grids, but this should be considered from case to case.

In a finite volume discretization, the domain Ω is covered with disjoint polygonal cells, often triangles and quadrilaterals:

$$\Omega = \bigcup_j \Omega_j, \quad \Omega_i \cap \Omega_j = \emptyset, \quad i \neq j. \tag{21}$$

Throughout this thesis we will assume convex cells. Non-convex cells add the need for more correction terms as in [10] and the circumcenter might be outside of the cell, causing more problems. The boundaries between the cells are referred to as edges or faces, while the word boundary is reserved for the domain boundary. The grid nodes ("cell corners") are at the endpoints of the faces, a node cannot be on a face. Using the divergence theorem, we can integrate the continuity equation over the cell, and calculate the conserved variables in the center of the cell as an average over the volume.

3.2 Staggered grids

For a staggered grid, we distinguish between values computed at the cell nodes (corners), cell centers, and edges/faces. Staggered grids were introduced by Harlow and Welch in 1965 for the Navier Stokes equations [12]. They compute the pressure in the cell center, but the normal components of the velocity at the middle of the cell faces. The alternative, where all variables are calculated at the nodes, is a collocated scheme. Staggered grids can be motivated by a simple thought: if a function is interpolated from its known values at a few points, the interpolation will be most accurate at the point exactly in between those values; as the continuity equation contains a waterlevel time-derivative and the momentum equation contains a velocity time-derivative, while the velocity space-derivative is in in the continuity equation and the waterlevel space-derivative in the momentum equation, it makes sense to discretize the equations half a step away from each other. This eliminates the checkerboard patterns that collocated grids tend to have, as there is no pressure-velocity decoupling.

For the center in the cell, many definitions are possible, such as the circumcenter (the point equidistant from all nodes) or the centroid (center of gravity/mass). Some polygons may not have a circumcenter (though it coincides with the centroid if it is a regular polygon), but, if it exists, it is the intersection of the perpendicular bisectors of the faces and this makes the grid orthogonal.

3.3 Determining the circumcenter

For triangles with vertices (x_1, y_1), (x_2, y_2), (x_3, y_3), the circumcenter is calculated using the exact formula

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = \frac{1}{2} \begin{pmatrix} (x_1 + x_2) + a(y_1 - y_2) \\ (y_1 + y_2) - a(x_1 - x_2) \end{pmatrix}, \qquad a = \frac{(x_2 - x_3)(x_3 - x_1) + (y_2 - y_3)(y_3 - y_1)}{(x_1 - x_2)(y_3 - y_1) - (y_1 - y_2)(x_3 - x_1)} \tag{22}$$

Note that if the triangle has an obtuse angle, the circumcenter is outside of the triangle, and if the triangle has a right angle, the circumcenter is in the middle of the longest edge.

For triangles, cycle quadrilaterals and regular polygons, the circumcenter exists, but for irregular polygons with more than three vertices, this is not always the case. Then we can use the iterative formula [23]

$$\vec{x}_c = \vec{x}_c + \gamma \sum_{\text{edges}} \left(\frac{\vec{x}_{e1} - \vec{x}_{e2}}{\|\vec{x}_{e1} - \vec{x}_{e2}\|_2} (\vec{x}_c - \vec{x}_f) \right) \cdot \frac{\vec{x}_{e1} - \vec{x}_{e2}}{\|\vec{x}_{e1} - \vec{x}_{e2}\|_2} \tag{23}$$

with $\vec{x}_e$ the endpoints of edge e, hence $\frac{\vec{x}_{e1} - \vec{x}_{e2}}{\|\vec{x}_{e1} - \vec{x}_{e2}\|_2}$ is a unit vector along the edge. Noting the similarity to the Gram-Schmidt method, we see why every vector $\vec{x}_c - \vec{x}_f$ computed by this algorithm should be orthogonal to the corresponding face. This is not exact, but the convergence is rapid [23]. In D-Flow FM, the parameter γ is set to 0.1 and the first estimate for the circumcenter is the centroid, which is calculated as the average of the polygon corners. For regular polygons, convergence is trivial as the circumcenter and centroid coincide.

3.4 Grid structure

Meandering rivers can be efficiently modeled with curvilinear grids that are aligned with the main flow direction [13]. Since flow gradients in the direction of the main flow are smaller than those orthogonal to the flow, we need a higher resolution in the cross direction than in the main flow direction, but the cells can be long in the main flow direction, such that the grid needs fewer cells than a triangular grid.

However, curvilinear grids have some drawbacks. In the inner bends of meandering rivers, grid lines become focused and cells become very small and the opposite problem happens for outer bends. Staircase representations of coastlines need to be corrected as well.

Curvilinear and triangular grids can be combined successfully [13] in a mixed grid model. When channels are discretized with curvilinear grids, we can use triangles to form a transition to local refinement, or to other water bodies such as the sea or another channel.

In curvilinear grids, orthogonality is hard to achieve, therefore the iteratively defined circumcenters are moved in D-Flow FM to achieve better orthogonality.

3.5 Orthogonality

Reconstructing the flow gradient at the faces causes cross-diffusion in non-orthogonal grids. Let e be the normalized vector between two circumcenters: $e = \frac{x_{c1} - x_{c2}}{\|x_{c1} - x_{c2}\|}$. In an orthogonal grid, e passes through x_f, and in this case e and the face-normal vector n are scalar multiples of each other. Then, the gradient along a normal vector is

$$(\nabla u \cdot n)_f = (\nabla u \cdot e)_f = \frac{u_f - u_c}{\|x_f - x_c\|} \tag{24}$$

On a non-orthogonal grid, e does not pass through x_f, and the normal vector n is a sum of e and a tangential vector t. Then

$$(\nabla u)_f \cdot n = (\nabla u)_f \cdot e + (\nabla u)_f \cdot t. \tag{25}$$

and to compute the gradient along a normal vector in a non-orthogonal grid we need to take into account not only the gradient along e, but also the term t in the discretization.

However, Ham et al. [10] work with a non-orthogonal grid, based on Casulli's work with orthogonal grids, designing an algorithm where the water levels are not corrected with a tangential component, but with a correction term computed from the water levels ζ_c from the surrounding cells. This has mostly influence on the discretization of the waterlevel gradients $\nabla\zeta$, which need to be corrected. While the orthogonal scheme of Casulli [6] yields a symmetric positive definite matrix which is easy to solve with the conjugate gradient method, the non-orthogonal scheme of Ham et al. gives a perturbed symmetric matrix. Still, the authors of [10] claim to encounter no problems with convergence and only a slightly longer running time.

An example of an orthogonal staggered grid (used by Perot in [16]) is the Delaunay mesh (consisting of triangles and quadrilaterals) and its dual, the Voronoi tessellation. A Voronoi tessellation of a plane with given nodes is obtained by partitioning the plane such that each cell consists of one of the nodes (the center) and the points in the plane that are closer to that node than to any other node. The cell faces in the Voronoi tessellation are orthogonal to the faces in the Delaunay mesh. The Delaunay triangles then form cells, and the vertices of the Voronoi cells are the circumcenters of the cells.

3.6 Discrete operators for mimetic schemes

The goal of mimetic schemes is to construct a discretization that mimics the mathematical properties of the original PDE.

The mimetic discretization methods are designed to leave the vector integral and differential identities intact on a discrete level. This often implies conservation properties. The name "mimetic" is chosen because a mimetic scheme accurately mimics the physical properties of the system: conservation of mass, momentum and energy. The scheme employed in D-Flow FM is not mimetic, since the bottom stress, wall friction and other mechanisms decrease the momentum and kinetic energy of the system, and the numerical energy loss only needs to be negligible with respect to the physical energy loss. The main mechanism to build mimetic schemes is to have discretizations of the inner product, divergence, gradient and curl conform to the same vector identites as their continuous analogs.

4 Discretizing the SWE in time and space

This section describes the discretization of the Shallow Water Equations as used in D-Flow FM [1, 13, 18].

4.1 Semi-implicit solution approach

We use an irregular staggered two-dimensional grid with the water level ζ_c in the cell circumcenter $\vec{x}_c$ (all quantities at the circumcenter will have a subscript c) and the face-normal velocities u_f at the midpoints of the faces of the cell (all quantities discretized at the face midpoints $\vec{x}_f$ will have the subscript f, except for l_f which is the (one-dimensional) length of face f). The total water depth $h = \zeta + b$ is discretized both in $\vec{x}_f$ and $\vec{x}_c$, and in the fluxes it is taken in some upwind manner. We will assume a fixed time step Δt and discretize the continuity equation (eq. 15) as integrals over the cell:

$$\frac{\zeta_c^{n+1} - \zeta_c^n}{\Delta t} V_c = -\sum_f s_{fc} l_f h_f^n u_f^{n+1} \tag{26}$$

where V_c is the horizontal cell area [14] and $n, n+1$ are time indices. The parameter $s_{fc} \in \{-1, 0, 1\}$ ensures the correct (counterclockwise) orientation of the integral. If face f belongs to the boundary of cell c, then $s_{fc}\vec{n}_f$ is the outward normal vector for the cell through the face.

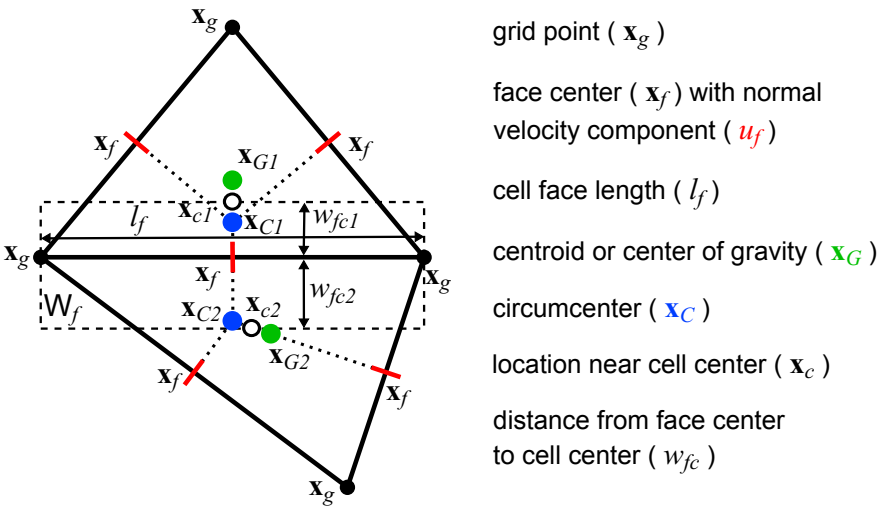


Figure 1: Two grid cells. Courtesy Mart Borsboom [1]

Let h_f be some approximation of the total water depth at the faces, the different choices will be discussed later. We proceed to discretize the momentum equation.

For now, we write $\mathrm{Adv}(\vec{u}^n)$ for the discretization of $\frac{1}{h}\big(\nabla\cdot(h\vec{u}\vec{u}) - \vec{u}\nabla\cdot(h\vec{u})\big)$. The non-conservative momentum equation (eq. 19) at the faces is discretized as

$$\frac{u_f^{n+1} - u_f^n}{\Delta t} + \mathrm{Adv}(\vec{u}^n) + g\frac{\zeta_{cR}^{n+1} - \zeta_{cL}^{n+1}}{||\vec{x}_{cR} - \vec{x}_{cL}||} + g\frac{||\tilde{u}_f^n|| u_f^{n+1}}{h_f^n C^2} = 0 \tag{27}$$

where $||\tilde{u}_f^n|| = \sqrt{(u_f^n)^2 + (\alpha_f v_{cL} + (1-\alpha_f)v_{cR})^2}$ and h in $\frac{1}{hC^2}$ originates from the depth-averaging and division by the depth.

We can extract u_f^{n+1}:

$$u_f^{n+1} = \left(1 + \Delta t \frac{g\|\tilde{u}_f^n\|}{h_f^n C^2}\right)^{-1} \cdot \left(u_f^n - g\Delta t \frac{\zeta_{cR}^{n+1} - \zeta_{cL}^{n+1}}{\|\vec{x}_{cR} - \vec{x}_{cL}\|} - \Delta t \mathrm{Adv}(\vec{u}^n)\right) \tag{28}$$

and substitute it in the continuity equation, allowing us to solve for ζ_c^{n+1}:

$$\zeta_c^{n+1} = \zeta_c^n - \frac{\Delta t}{V_c} \sum_f s_{fc} l_f h_f^n \left(u_f^n - g\Delta t \frac{\zeta_{cR}^{n+1} - \zeta_{cL}^{n+1}}{\|\vec{x}_{cR} - \vec{x}_{cL}\|} - \Delta t \mathrm{Adv}(\vec{u}^n)\right) \left(1 + \Delta t \frac{g\|\tilde{u}_f^n\|}{h_f^n C^2}\right)^{-1} \tag{29}$$

This equation (eq. 29) forms a symmetric positive definite system of equations for the unknown water levels at the new time level $n+1$. This can be solved efficiently by the Conjugate Gradient method [13]. Having ζ_c^{n+1}, we can substitute and directly calculate u_f^{n+1}, which necessarily conserve mass since this is forced by the implicit coupling of the momentum and continuity equation. Note that the momentum and kinetic energy are not necessarily conserved, since they are dependent on the discretization of $\mathrm{Adv}(\vec{u}^n)$.

The discretization in D-Flow FM then uses the θ-method [18]. The parameter θ needs to be in $[\frac{1}{2}, 1]$ for stability [4]. Then ζ_c^{n+1} is replaced by $\theta\zeta^{n+1}+(1-\theta)\zeta^n$ and $h_f^n u_f^{n+1}$ by $\theta h_f^n u_f^{n+1}+(1-\theta)h_f^n u_f^n$ to obtain a semi-implicit discretization. The discretization is second-order accurate for $\theta = \frac{1}{2}$ (Crank-Nicholson method), and first-order accurate for $\frac{1}{2} < \theta < 1$. When θ is close to 1, the first order accuracy in time may lead to excessive damping of running waves. For $\theta = 1$, the pressure gradient is implicit in the momentum equation and the fluxes are implicit in the continuity equation. The advection is always explicit, hence the discretization of the system is semi-implicit.

4.2 Discretizing the advection operator

In [1] it is shown that we need to use $\tilde{h}_f$ in the time derivative of the momentum equation for energy conservation, but $\bar{h}_f$ in the pressure gradient term to ensure momentum and energy conservation, where:

$$\text{(Choices for) discretization of } h: \begin{cases} H_f = \max(0, d_f + \frac{1}{2}(\zeta_{cR} + \zeta_{cL})) & \text{(Casulli [7, eq.41])} \\ \tilde{h}_f = \frac{1}{2}(\zeta_{cR} + \zeta_{cL}) + \alpha_f d_{cL} + (1-\alpha_f) d_{cR} & \text{(Kramer, Stelling [14])} \\ \bar{h}_f = \alpha_f h_{cL} + (1-\alpha_f) h_{cR} & \text{(Borsboom [1])} \end{cases}$$

Let $\vec{u}_f^*$ be the full velocity vector at the faces reconstructed from the cell center velocities $\vec{u}_c$, which are computed (from u_f) by Perot's velocity vector reconstruction or any other method (see paragraph 4.4 and chapter 5). Then define $\alpha_f = \frac{\|\vec{x}_{cL} - \vec{x}_f\|}{\|\vec{x}_{cL} - \vec{x}_{cR}\|}$, where cL is the upwind cell from f and it is important that the line $\vec{x}_{cL} - \vec{x}_{cR}$ passes through $\vec{x}_f$ (this happens automatically if the grid is orthogonal). Then α_f is a dimensionless distance expression and can be used as a weighing factor. Kramer discusses the approaches of Perot and Wenneker in the discretization of the advection term $\nabla \cdot h\vec{u}\vec{u}$ for the conservative momentum equation. Discretize $\vec{q} = h\vec{u}$ as

$$\vec{q}_c = \frac{1}{V_c} \sum_f l_f (\vec{x}_{cL} - \vec{x}_{cR}) \vec{q}_f \vec{n}_f s_{fc} \tag{30}$$

where $\vec{n}_f$ are normal vectors at the faces. This expression forms a balance since $V_c I = \sum_f l_f(\vec{x}_{cL} - \vec{x}_{cR})\vec{n}_f\vec{n}_f^T$. Then [14, eq.21] we can write

$$\int_\Omega \nabla \cdot \vec{q}\vec{u}\, dV = \int_{\partial\Omega} \vec{q}\cdot\vec{n}\vec{u}\, dS = \sum_f \int_{\text{face } f} \vec{q}\cdot\vec{n}_f\vec{u} \approx \sum_f Q_f\vec{u}_f = V_c\vec{q}_c \tag{31}$$

where Q_f is the flux through face f. Kramer and Stelling [14] then use $\tilde{h}_f$ to get the discretization of the momentum equation

$$\frac{dq_f}{dt} + (\alpha_f\vec{q}_{cL} + (1-\alpha)\vec{q}_{cR})\cdot\vec{n}_f + g\tilde{h}_f\frac{\zeta_{cR}-\zeta_{cL}}{\|\vec{x}_{cR}-\vec{x}_{cL}\|} = 0. \tag{32}$$

and they claims conservation, supported by [16]. The advection term is integrated explicitly in D-Flow FM [13].

Kramer and Stelling also present Wenneker's scheme without a proof of conservation, though both their article and Wenneker's [21] claim conservative behaviour in numerical test cases. Where Perot uses just small rectangles (between the circumcenters) as control volume, Wenneker's scheme makes use of the entire cells belonging to a face. For a face f_0 and its two cells cR, cL, take the faces $f_1, f_2, \ldots$ belonging to the cells but not the face f_0 itself, and discretize the momentum equation on f_0 as

$$\frac{d(h\vec{u})_{f_0}}{dt} + \Big(\sum_{k:k\neq 0} Q_{f_k}\vec{u}_f^*\Big)\cdot\vec{n}_{f_0} + g\tilde{h}_{f_0}\frac{\zeta_{cR}-\zeta_{cL}}{\|\vec{x}_{cR}-\vec{x}_{cL}\|} = 0 \tag{33}$$

Here $Q_f = \vec{q}_f\cdot\vec{n}_f$ integrated along face f, i.e. $h_f(\vec{u}_f^*\cdot s_{fc}\vec{n}_f)l_f$.

Perot's advection discretization is the one used in D-Flow FM with the numerical experiments. The main problem now is to find an appropriate approximation of $\vec{u}_f^*$. To find the normal component, we need the entire vector. In order to find this, we first construct the velocity vector for a cell (at the cell circumcenter, as described in section 4.4 and chapter 5) and then reconstruct the velocity vector at the face. The reconstructed $\vec{u}_c$, and, when available, its gradient, can then be used in the different types of advection and diffusion discretizations as described below.

4.3 Discretizing the diffusion operator

In D-Flow FM, the diffusion tensor $\text{Diff}(u^n)$ is discretized per cell, then averaged to the faces and multiplied by the face-normal vector to obtain the diffusion of the momentum equation normal to the faces. For each face f, we consider the other faces of the cells cL, cR , where u_{fx} is the difference of u with respect to x at the face along $\vec{x}_{cR} - \vec{x}_{cL}$, i.e. $\frac{u_{cR}-u_{cL}}{\|\vec{x}_{cL}-\vec{x}_{cR}\|_2}$ (and similar for u_{fy}, v_{fx}, v_{fy}) and u_l is shorthand for the directional derivative of u along a face f: $u_l = \frac{1}{l_f}(\vec{u}_{n1} - \vec{u}_{n2})$.

$$\begin{aligned}\text{Diff}(x_f, y_f) \;=\; & \sum_{\text{neighbour faces}} -\frac{\alpha_f}{h_{cL}V_{cL}} l_f \begin{pmatrix} u_{fx}(1+n_x^2) - n_xn_yu_l & u_{fy}n_xn_y - n_y^2u_l \\ v_{fx}n_xn_y + n_x^2v_l & v_{fy}(1+n_y^2) + n_xn_yv_l \end{pmatrix}\cdot\vec{n}_f \\ & -\frac{1-\alpha_f}{h_{cR}V_{cR}} l_f \begin{pmatrix} u_{fx}(1+n_x^2) - n_xn_yu_l & u_{fy}n_xn_y - n_y^2u_l \\ v_{fx}n_xn_y + n_x^2v_l & v_{fy}(1+n_y^2) + n_xn_yfv_l \end{pmatrix}\cdot\vec{n}_f \end{aligned} \tag{34}$$

where $\vec{n}_f = (n_x, n_y)$.

4.4 Velocity vector reconstruction with Perot's method

For use in the advection and diffusion, we need a vector $\vec{u}_c^n$ in each circumcenter, derived from the scalar u_f^n (face-normal values) on the faces. D-Flow FM uses Perot's velocity vector reconstruction method. The $\vec{u}_c^n$ is then used to compute u_f^{n+1}, i.e. explicitly. Perot developed this reconstruction as part of a fully implicit scheme for the Navier-Stokes equations [16, section 5.4].

The total flow through the cell can be computed by the following integral, using the divergence theorem, and discretized by assuming the cell boundary to consist of straight lines.

$$\int_V \nabla \cdot \vec{u}(\vec{x} - \vec{x}_c)\, dV = \oint_{\partial V} \vec{u} \cdot \vec{n}(\vec{x} - \vec{x}_c)\, ds = V_c \sum_{f \in \partial\Omega} s_{fc} u_f l_f (\vec{x}_f - \vec{x}_c) \tag{35}$$

For a point $\vec{x}_0$ inside the cell, we can then compute the average velocity:

$$\vec{u}_0 = \frac{1}{V_c} \int_{\partial\Omega_c} (\vec{u} \cdot \vec{n})(\vec{x} - \vec{x}_0)\, dA. \tag{36}$$

Assuming the scalar velocities normal to the cell faces u_f are known, we can reconstruct $(u, v)_c$ with Perot's formula

$$\vec{u}_c^P = \frac{1}{V_c} \sum_{f \in \partial\Omega_c : \text{cell faces}} s_{fc} l_f u_f (\vec{x}_f - \vec{x}_c). \tag{37}$$

4.5 Momentum conservation

It is not always possible to prove local momentum conservation, but we prove global momentum conservation according to [1]. We write the momentum equation at each face, using $M = \nabla \cdot (huu) - 2 \sin\phi h \Omega \times \vec{u}$, as $\frac{dh\vec{u}}{dt} + \vec{M} + gh\nabla\zeta = 0$. Analogous to [16], we then multiply with the normal of the face and integrate over the control volume Ω_f which is determined by the face length l_f and the distances from the face center to the cell centers $W = w_{fcL} + w_{fcR}$, where w_{fc} is the distance between the cell center and the face midpoint.

$$\begin{aligned} \int_{\Omega_f} (\frac{dh\vec{u}}{dt} + \vec{M} + gh\nabla\zeta) \cdot \vec{n}_{fcL} &= \int_{\Omega_f} (\frac{dh\vec{u} \cdot \vec{n}_{fcL}}{dt} + \vec{M} \cdot \vec{n}_{fcL} + gh\frac{d\zeta}{d\vec{n}})\, dV & (38) \\ &\approx s_{fcL} l_f W \frac{dh_f u_f}{dt} + l_f (w_{fcL} \vec{M}_{cL} + w_{fcR} \vec{M}_{cR}) \cdot \vec{n}_{fcL} \\ &\quad + l_f g h_f (\zeta_{cR} - \zeta_{cL}) = 0 & (39) \end{aligned}$$

where $h_f = \frac{1}{W}(w_{fcL} h_{cL} + w_{fcR} h_{cR}) = \tilde{h}_f$. The time derivatives can be rewritten as (substituting $\vec{u}_c = \frac{1}{\tilde{h}_c V_c} \sum_{\text{cell faces}} s_{fc} l_f \tilde{h}_f u_f (\vec{x}_f - \vec{x}_c)$)

$$\sum_{\text{all inner faces}} s_{fc} l_f W \frac{dh_f u_f}{dt} (\vec{x}_f - \vec{x}_c) = \sum_{\text{all inner cells}} \frac{dV_c \tilde{h}_c \vec{u}_c}{dt}, \tag{40}$$

with $\tilde{h}_c$ some average of the $\tilde{h}_f$ belonging to its cell (the fact that this definition is not important also implies that the h_f in the time derivative in the momentum equation does not play a role in

momentum conservation), using that V_c, l_f and s_{fc} are fixed properties of the grid. We can sum the discretization over all inner faces (and multiply by W for convenience):

$$\begin{aligned}\sum_{\text{all inner faces}} s_{fcL} l_f W \frac{dh_f u_f}{dt}(\vec{x}_f - \vec{x}_{cL}) + l_f(w_{fcL}\vec{M}_{cL} + w_{fcR}\vec{M}_{cR}) \cdot \vec{n}_{fcL}(\vec{x}_f - \vec{x}_{cL}) \\ + l_f g h_f(\zeta_{cR} - \zeta_{cL})(\vec{x}_f - \vec{x}_{cL}) \\ + s_{fcR} l_f W \frac{dh_f u_f}{dt}(\vec{x}_f - \vec{x}_{cR}) + l_f(w_{fcL}\vec{M}_{cL} + w_{fcR}\vec{M}_{cR}) \cdot \vec{n}_{fcR}(\vec{x}_f - \vec{x}_{cR}) \\ + l_f g h_f(\zeta_{cR} - \zeta_{cL})(\vec{x}_f - \vec{x}_{cR}) = 0 \end{aligned} \tag{41}$$

Now, if $\vec{x}_{cR} - \vec{x}_{cL} = (\vec{x}_{cR} - \vec{x}_f) + (\vec{x}_f - \vec{x}_{cL})$, (the orthogonality criterion, otherwise the ζ-slope has a component tangential to the faces, which can be solved using ζ_c from surrounding cells [10], but then the matrix that solves ζ is not symmetric positive definite anymore), we write $\vec{x}_{cR} - \vec{x}_{cL} = W\vec{n}_{fcL} = -W\vec{n}_{fcR}$ we get that the gradients ζ/W are normal to the faces, and also $\vec{n}_{fcL} = -\vec{n}_{fcR}$, the discretization reduces to

$$\begin{aligned}\sum_{\text{all inner faces}} s_{fcL} l_f \frac{dh_f u_f}{dt}(\vec{x}_f - \vec{x}_{cL}) + l_f w_{fcL}\vec{M}_{cL} \cdot \vec{n}_{fcL}\vec{n}_{fcL} + l_f g h_f(\zeta_{cR} - \zeta_{cL})\vec{n}_{fcL} \\ + s_{fcR} l_f \frac{dh_f u_f}{dt}(\vec{x}_f - \vec{x}_{cR}) + l_f w_{fcR}\vec{M}_{cR} \cdot \vec{n}_{fcR}\vec{n}_{fcR} \end{aligned} \tag{42}$$

$$\begin{aligned} = \sum_{\text{all inner faces}} s_{fcL} l_f \frac{dh_f u_f}{dt}(\vec{x}_f - \vec{x}_{cL}) + l_f w_{fcL}\vec{M}_{cL} \cdot \vec{n}_{fcL}\vec{n}_{fcL} + l_f g h_f(\zeta_{cR} - \zeta_{cL})\vec{n}_{fcL} \\ - s_{fcL} l_f \frac{dh_f u_f}{dt}(\vec{x}_f - \vec{x}_{cL}) - l_f w_{fcL}\vec{M}_{cL} \cdot \vec{n}_{fcL}\vec{n}_{fcL} \end{aligned} \tag{43}$$

$$= 0 \tag{44}$$

Since $V_c I = \sum_{\text{cell faces}} l_f \vec{n}_{fc}(\vec{x}_f - \vec{x}_c) = \sum_{\text{cell faces}} l_f w_{fc}\vec{n}_{fc}\vec{n}_{fc}$ and we can rewrite

$$\sum_{\text{all inner faces}} l_f w_{fc}\vec{M}_c \cdot \vec{n}_{fc}\vec{n}_{fc} = \sum_{\text{all inner cells}} V_c \vec{M}_c \tag{45}$$

Furthermore, we have, for $\bar{h}_f = \frac{1}{2}(h_{cL} + h_{cR})$:

$$\sum_{\text{all inner faces}} l_f g \bar{h}_f(\zeta_{cR} - \zeta_{cL})\vec{n}_{fcL} = \sum_{\text{all inner faces}} l_f g(\frac{1}{2}(h_{cR}^2 - h_{cL}^2) - \bar{h}_f(b_{cR} - b_{cL})\vec{n}_{fc1} \tag{46}$$

$$= \sum_{\text{all inner cells}} g\frac{h_c^2}{2} \sum_{\text{cell faces}} l_f \vec{n}_{fc} - g \sum_{\text{all inner cells}} \sum_{\text{cell faces}} l_f w_{fc} \bar{h}_f \frac{b_{cR} - b_{cL}}{W} \tag{47}$$

Writing $\zeta_c = h_c - b_c$, and using that each inner cell has a closed surface, hence $\sum_{\text{cell faces}} l_f \vec{n}_{fc} = 0$, all terms in the discretization of the momentum equation, except the Coriolis terms, cancel out, and hence we have proven global momentum conservation.

Kinetic and potential energy can be shown to be conservative in most terms of the SWE, but the time derivative does not conserve energy. This means that energy conservation errors are small for solutions that vary slowly in time, but not zero. However, for slowly varying flow and for steady-state solutions, the energy is well conserved [1, Section 5].

Though conservation is important, there are other considerations. The two different choices for h_f which are necessary for conservation: $\tilde{h}_f$ in the time derivative for energy conservation [19] and $\bar{h}_f$ is used in the ζ-slope term for momentum and energy conservation [1, section 3.2], are not consistent with each other and may have an overall negative effect on the accuracy. After all, conservation does not guarantee accuracy, but an accurate discretization necessarily gives a certain amount of conservation.

For details on the specific discretization in D-Flow FM, we study [13], which formulates $\mathrm{Adv}(\vec{u})$ in a not entirely momentum-conserving way, but since all other terms are in finite volume formulation, most of the discretization in D-Flow FM is conservative. Advection is evaluated explicitly (i.e. using $Adv(u_f^n)$) The conservative discretization of $\frac{d\zeta}{dt}$ implies volume conservation. We study the volumetric flow rate $Q = \frac{\partial V}{\partial t}$. Approximate

$$\left.\frac{\partial V}{\partial t}\right|_{total} = \sum Q_{in} - Q_{out} \tag{48}$$

Define the momentum control volume V_{cf} for a face f and a cell c as the volume spanned by the two endpoints of the face and the circumcenter of the cell and let $u_{in} = \vec{u} \cdot \vec{n}_f$. Then the control volume for the entire cell is $V_f = \alpha V_{cL} + (1-\alpha)V_{cR}$ and we look at the momentum conservation

$$\frac{\partial V_f u}{\partial t} = V_f \frac{\partial u}{\partial t} + u\frac{\partial V_f}{\partial t} = V_f \frac{\partial u}{\partial t} + u\sum Q_{in} - Q_{out}. \tag{49}$$

Then

$$\frac{\partial u}{\partial t} = \frac{1}{V_f}\Big(\sum_{in} Q u_{in} - \sum_{out} Q u_{out} - u(\sum Q_{in} - Q_{out})\Big) = \frac{1}{V_f}\Big(\sum_{in} Q(u_{in}-u) - \sum_{out} Q(u_{out}-u)\Big) \tag{50}$$

Now compute u_{in}, $u_{out} = u_c$ using Perot's velocity reconstruction technique, then

$$\frac{\partial u}{\partial t} = \frac{1}{V_f}\Big(\alpha\Big(\sum_{inL} Q(u_{inL}-u) - \sum_{outL} Q(u_{outL}-u)\Big) + (1-\alpha)\Big(\sum_{inR} Q(u_{inR}-u) - \sum_{outR} Q(u_{outR}-u)\Big)\Big) \tag{51}$$

and analogous for $\frac{\partial v}{\partial t}$. Taking $u_{out} = u$, hence treating this as an upwind scheme, the second and fourth term drop out and we have a finite volume discretization formulation for the momentum equation. This is not conservative because of the choice of V_f, and this means there is no strict momentum conservation in D-Flow FM.

4.6 Time step limitations

If the velocity per timestep is larger than the grid spacing, the information would propagate through more than one cell in a time step. This is why we consider the Courant number $(u + \sqrt{gh})\frac{\Delta t}{\Delta x} + \nu\frac{\Delta t}{(\Delta x)^2} < (1 + \text{bottom friction})$. The celerity $\sqrt{gh}$ is the speed of a surface gravity wave in shallow water with depth h. Since the wave part of the equations (this gives the equations for ζ that are solved in every timestep) are implicit, the wave Courant number $C_{\text{wave}} = \sqrt{gh}\frac{\Delta t}{\Delta x}$ has no influence, and we only have to consider the Courant numbers for flow (advection) and diffusion. The Courant number reduces to $u\frac{\Delta t}{\Delta x} + \nu\frac{\Delta t}{(\Delta x)^2} < (1 + \text{bottom friction})$. For a 2D flow that is skewed to the grid, streamlines can cross cell boundaries earlier and it can be shown that a Courant number of 0.7 is necessary for maintaining stability. This is the standard maximum Courant number in D-Flow FM, where the Courant number is fixed, Δx depends on the cell geometry and Δt is then chosen to satisfy the Courant criterion.

5 Accuracy of the velocity reconstruction

5.1 Analysis of Perot's reconstruction

Peixoto and Barros [15] prove that Perot's velocity vector reconstruction (section 4.4) is second-order accurate on regular grids (for example squares, equilateral triangles or hexagons). In other cases, it is only first-order accurate. Consider a linear flow velocity field $\vec{u} = \vec{u}_0 + \nabla\vec{u}(\vec{r})$ in some point $\vec{x}_0$ in the cell Ω_c, where $\nabla\vec{u}$ is a matrix with the (constant) derivatives and $\vec{r} = \vec{x} - \vec{x}_0$. Parametrize $\partial\Omega = \cup_{\text{faces}}(\vec{x}_f + a\vec{t_f})$, $a \in [-l_f/2, l_f/2]$, $\vec{t}_f$ is a unit vector at the point $\vec{x}_f$ along face f.

$$\int_\Omega \vec{u} + (\vec{x}_f - \vec{x}_0)\nabla\cdot u\, dS = \sum_f \int_{\vec{x}_f + a\vec{t_f}, a\in[-l_f/2, l_f/2]} (\vec{x}_f - \vec{x}_0)(u\cdot s_{fc}\vec{n}_f)\, ds \tag{52}$$

Then, using $\vec{r}_f = \vec{x}_f + a\vec{t_f} - \vec{x}_0$:

$$\begin{aligned}
V_c\vec{u}_0 &= \sum_{\text{faces}} \int_{\vec{x}_f - \frac{l_f}{2}\vec{t}_f}^{\vec{x}_f + \frac{l_f}{2}\vec{t}_f} (\vec{x}_f + a\vec{t_f} - \vec{x}_0)(\vec{u}_0 + \nabla\vec{u}_0\cdot(\vec{x}_f + a\vec{t_f} - \vec{x}_0))\cdot\vec{n}_f s_{fc}\, ds && (53)\\
&= \sum_{\text{faces}} \int_{\vec{x}_f - \frac{l_f}{2}\vec{t}_f}^{\vec{x}_f + \frac{l_f}{2}\vec{t}_f} \vec{r}_f(\vec{u}_f\cdot\vec{n}_f s_{fc}) + a\vec{t_f}\vec{u}_f\cdot\vec{n}_f s_{fc} + a\vec{r}_f(\nabla\vec{u}_0\cdot\vec{t_f}\cdot\vec{n}_f s_{fc})\\
&\qquad\qquad + a^2\vec{t}_f(\nabla\vec{u}_0\cdot(\vec{t_f})\cdot\vec{n}_f s_{fc})\, ds && (54)\\
&= \sum_{\text{faces}} \vec{r}_f\vec{u}_f l_f + \vec{t}_f(\nabla\vec{u}_0\cdot(\vec{t_f})\cdot\vec{n}_f s_{fc})\frac{l_f^3}{12} && (55)\\
&= \sum_{\text{faces}} (\vec{x}_f - \vec{x}_0)\vec{u}_f l_f + V_c E_1 && (56)
\end{aligned}$$

where $(\nabla\vec{u})_0 = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}$. We see that we get the error term $E_1 = \frac{1}{V_c}\vec{t_f}(\nabla\vec{u}_0\cdot(\vec{t_f})\cdot\vec{n}_f s_{fc})\frac{l_f^3}{12}$. If Ω is an aligned polygon (opposite edges are parallel and of the same length), we let $\bar{f}$ be the opposite face of f, then $s_{fc}\vec{n}_f = -s_{\bar{f}c}\vec{n}_{\bar{f}}$, $\vec{t_f} = -\vec{t}_{\bar{f}}$, $l_f = l_{\bar{f}}$, and we let $\vec{x}_0$ be the centroid

$$\begin{aligned}
V_c E_1 &= \sum_{\text{faces}} \vec{t_f}(\nabla\vec{u}\cdot\vec{t_f}\cdot s_{fc}\vec{n}_f)\frac{l_f^3}{12} && (57)\\
&= \sum_{\text{half of the faces}} \vec{t_f}(\nabla\vec{u}\cdot\vec{t_f}\cdot s_{fc}\vec{n}_f)\frac{l_f^3}{12} + \vec{t}_{\bar{f}}(\nabla\vec{u}\cdot\vec{t}_{\bar{f}}\cdot s_{\bar{f}c}\vec{n}_{\bar{f}})\frac{l_{\bar{f}}^3}{12} && (58)\\
&= \sum_{\text{half of the faces}} \vec{t_f}(\nabla\vec{u}\cdot\vec{t_f}\cdot s_{fc}\vec{n}_f)\frac{l_f^3}{12} - \vec{t_f}(\nabla\vec{u}\cdot\vec{t_f}\cdot s_{fc}\vec{n}_f)\frac{l_f^3}{12} = 0 && (59)
\end{aligned}$$

so E_1 vanishes on regular grids.

Perot [16] proves that this method conserves momentum for the divergence form of the Navier-Stokes equations, and conserves kinetic energy for both the divergence and the rotational form, but he does not consider the behaviour on the Shallow Water Equations.

5.2 Second order corrections for Perot

5.2.1 Integration over the dual volume

Perot reconstructs the velocity vector in the cell center $\vec{x}_c$ (eq. 37) as

$$V_c \vec{u}_c^P = \int_{\partial\Omega} (\vec{u}\cdot\vec{n})(\vec{x}_c - \vec{x}_f)\, dA \approx \sum_{\text{faces}} s_{fc} l_f \vec{u}_f (\vec{x}_c - \vec{x}_f) \tag{60}$$

Now we rewrite the integral to get a closer approximation, analogous to [17]:

$$\int_{\partial\Omega} (\vec{u}\cdot\vec{n})(\vec{x}-\vec{x}_c)\, dA = \int_\Omega \nabla\cdot(\vec{u}(\vec{x}-\vec{x}_c))\, dV = \int_\Omega (\nabla\cdot\vec{u})(\vec{x}-\vec{x}_c) + (\vec{u}\cdot\nabla)(\vec{x}-\vec{x}_c)\, dV \tag{61}$$

Then $(\vec{u}\cdot\nabla)(\vec{x}-\vec{x}_c) = \vec{u}$ gives us

$$\int_\Omega \vec{u}\, dV = \int_{\partial\Omega} (\vec{u}\cdot\vec{n})(\vec{x}-\vec{x}_c)\, dA - \int_\Omega (\nabla\cdot\vec{u})(\vec{x}-\vec{x}_c)\, dV \tag{62}$$

If we substitute $\vec{u} = \vec{u}_c + (\vec{x}-\vec{x}_c)\cdot(\nabla\vec{u})_c + \frac{1}{2}((\vec{x}-\vec{x}_c)^2 : (\nabla\nabla)\vec{u})_c + ...$ in the left hand side and rearrange, where we use $V_c = \int_\Omega dV$ and $\vec{x}_G = \frac{1}{V_c}\int_\Omega \vec{x}\, dV$ (the centroid), we get a corrected velocity reconstruction

$$V_c\vec{u}_c = \int_{f\in\partial\Omega} (\vec{u}\cdot\vec{n})(\vec{x}-\vec{x}_c) dA - V_c\Big((\nabla\cdot\vec{u})_c(\vec{x}_G-\vec{x}_c) - (\vec{x}_G-\vec{x}_c)\cdot(\nabla\vec{u})_c\Big) + l_f O(\Delta_{xy}^2) \tag{63}$$

We use the error term $E_1 = \frac{1}{V_c}\frac{l_f^3}{12} s_{fc}\vec{t}_f(\nabla\vec{u}(\vec{t}_f)\cdot\vec{n}_f)$ as computed in the previous paragraph. However, since this was only computed for a linear field, the second part of the error was disregarded. Since it is of the form $l_f O(\Delta_{xy}^2)$ it gets absorbed in the last term of the equation above. Then

$$V_c\vec{u}_c = \sum_{\text{faces}} s_{fc} l_f u_f(\vec{x}_f-\vec{x}_c) + \frac{l_f^3}{12} s_{fc}\vec{t}_f(\nabla\vec{u}(\vec{t}_f)\cdot\vec{n}_f) - V_c\Big((\nabla\cdot\vec{u})_c(\vec{x}_G-\vec{x}_c) - (\vec{x}_G-\vec{x}_c)\cdot(\nabla\vec{u})_c\Big) + l_f O(\Delta_{xy}^2) \tag{64}$$

forms the corrected velocity reconstruction, where $\vec{t}_f$ is a unit vector tangential to the face and $s_{fc} \in \{-1, 0, 1\}$ ensures correct (counter-clockwise) orientation of the integral. Define the error term by writing $u_c = u_c^P + E + O(\Delta_{xy}^2)$, so $E = E_1 + (\nabla\cdot\vec{u}_c(\vec{x}_G-\vec{x}_c) - (\vec{x}_G-\vec{x}_c)\cdot(\nabla\vec{u})_c$. Then define $l_f\vec{n}_f = (y_{cR}-y_{cL}, x_{cL}-x_{cR})^T$ and $l_f\vec{t}_f = (x_{cR}-x_{cL}, y_{cR}-y_{cL})^T =: (\Delta x, \Delta y)^T$. Here cL denotes the circumcenter of the cell upstream of the face and cR denotes the circumcenter of the cell downstream of the face under consideration. For ease of notation, we re-define $(\nabla\vec{u})_c := \begin{pmatrix} \tilde{u}_{cx} & \tilde{v}_{cx} \\ \tilde{u}_{cy} & \tilde{v}_{cy} \end{pmatrix}$ in the circumcenter (as opposed to $\nabla\vec{u}_0$ above, which was transposed and defined in an arbitrary point $\vec{x}_0$) in order to be able to interchange $\vec{n}_f$ and $\vec{t}_f$ and write E_1 as follows:

$$\frac{1}{12V_c}\sum_{\text{faces}} (((\nabla\vec{u})_c\cdot l_f\vec{n}_f)\cdot l_f\vec{t}_f) l_f\vec{t}_f = \frac{1}{12V_c}\sum_{\text{faces}} \begin{pmatrix} \tilde{u}_{cx}\Delta x^2\Delta y - \tilde{v}_{cx}\Delta y^3 + \tilde{u}_{cy}\Delta x\Delta y^2 - \tilde{v}_{cy}\Delta x^2\Delta y \\ \tilde{u}_{cx}\Delta x\Delta y^2 - \tilde{v}_{cx}\Delta x^2\Delta y + \tilde{u}_{cy}\Delta y^3 - \tilde{v}_{cy}\Delta x\Delta y^2 \end{pmatrix} \tag{65}$$

and for each cell we get

$$E = \frac{1}{12V_c}\sum_{\text{faces}}\begin{pmatrix}\tilde{u}_{cx}\Delta x^2\Delta y - \tilde{v}_{cx}\Delta y^3 + \tilde{u}_{cy}\Delta x\Delta y^2 - \tilde{v}_{cy}\Delta x^2\Delta y\\ \tilde{u}_{cx}\Delta x\Delta y^2 - \tilde{v}_{cx}\Delta x^2\Delta y + \tilde{u}_{cy}\Delta y^3 - \tilde{v}_{cy}\Delta x\Delta y^2\end{pmatrix} - \begin{pmatrix}-2\tilde{u}_{cx}(x_G - x_c) + \tilde{u}_{cy}(y_G - y_c) + \tilde{v}_{cy}(x_G - x_c)\\ 2\tilde{v}_{cy}(y_G - y_c) + \tilde{u}_{cx}(y_G - y_c) + \tilde{v}_{cx}(x_G - x_c)\end{pmatrix}. \quad (66)$$

We rewrite this as $E = \begin{pmatrix}a_{ux1}\tilde{u}_{cx} + a_{uy1}\tilde{u}_{cy} + a_{vx1}\tilde{v}_{cx} + a_{vy1}\tilde{v}_{cy}\\ a_{ux2}\tilde{u}_{cx} + a_{uy2}\tilde{u}_{cy} + a_{vx2}\tilde{v}_{cx} + a_{vy2}\tilde{v}_{cy}\end{pmatrix}$ with

$$a_{ux1} = \frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x^2\Delta y - 2(x_G - x_c) \quad (67)$$

$$a_{vx1} = -\frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x^3 \quad (68)$$

$$a_{uy1} = \frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x\Delta y^2 - (y_G - y_c) \quad (69)$$

$$a_{vy1} = -\frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x^2\Delta y - (x_G - x_c) \quad (70)$$

$$a_{ux2} = \frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x\Delta y^2 - (y_G - y_c) \quad (71)$$

$$a_{vx2} = -\frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x^2\Delta y - (x_G - x_c) \quad (72)$$

$$a_{uy2} = \frac{1}{12V_c}\sum_{\text{cell faces}}\Delta y^3 \quad (73)$$

$$a_{vy2} = -\frac{1}{12V_c}\sum_{\text{cell faces}}\Delta x\Delta y^2 - 2(y_G - y_c) \quad (74)$$

$$\tilde{u}_{cx} = \frac{1}{2A_{nb}}\sum_{l\in S_l}(u_{c(1,l)} + u_{c(2,l)})(y_{n(2,l)} - y_{n(1,l)}) \quad (75)$$

$$\tilde{u}_{cy} = -\frac{1}{2A_{nb}}\sum_{l\in S_l}(u_{c(1,l)} + u_{c(2,l)})(x_{n(2,l)} - x_{n(1,l)}) \quad (76)$$

$$\tilde{v}_{cx} = \frac{1}{2A_{nb}}\sum_{l\in S_l}(v_{c(1,l)} + v_{c(2,l)})(y_{n(2,l)} - y_{n(1,l)}) \quad (77)$$

$$\tilde{v}_{cy} = -\frac{1}{2A_{nb}}\sum_{l\in S_l}(v_{c(1,l)} + v_{c(2,l)})(x_{n(2,l)} - x_{n(1,l)}) \quad (78)$$

where S_l forms the connection between two neighbouring circumcenters of the cell under consideration. A_{nb} is the area spanned by the neighbouring circumcenters, as seen in figure 2. Now, each neighbour in the link l can be considered clockwise or counterclockwise relative to the cell under consideration, and in the above expressions, the counterclockwise (ccw) neighbour in each calculation provides a positive contribution, while the clockwise (cw) neighbour gives a negative

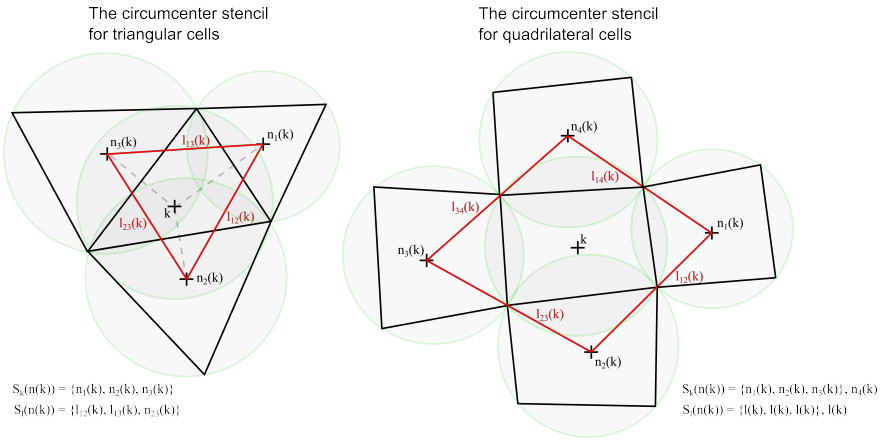


Figure 2: Integration area for second order velocity reconstruction over the dual volume. The point k is the circumcenter of cell c, and A_{nb} is spanned by the points $n_i(k)$. Courtesy Frank Platzek [17]

contribution, and we can rewrite $u_{c(1,l)} + u_{c(2,l)} =: u_l$, rename the $x_{n(1,l)}, \ldots$ to $x_{cw}, x_{ccw}, y_{cw}, y_{ccw}$, E becomes

$$\frac{1}{2A_{nb}} \begin{pmatrix} a_{ux1} \sum_l u_l(y_{ccw} - y_{cw}) - a_{uy1} \sum_l u_l(x_{ccw} - x_{cw}) + a_{vx1} \sum_l u_l(y_{ccw} - y_{cw}) - a_{vy1} \sum_l u_l(x_{ccw} - x_{cw}) \\ a_{ux2} \sum_l u_l(y_{ccw} - y_{cw}) - a_{uy2} \sum_l u_l(x_{ccw} - x_{cw}) + a_{vx2} \sum_l u_l(y_{ccw} - y_{cw}) - a_{vy2} \sum_l u_l(x_{ccw} - x_{cw}) \end{pmatrix} \tag{79}$$

Collecting the calculations for E into a matrix A, we solve $Au_c = u_c^P + Cu_{(\text{boundary conditions})}$ using BiCGstab. Note that the matrix A only needs to be computed once since it only uses the geometry of the grid, and if $\vec{x}_c = \vec{x}_G$, it becomes the identity matrix. Since we need to compute the Perot-velocities in every time step as well as the corrections, it is useful to find out when Perot's method suffices, and when we need to correct it.

As this method will be referenced a lot of times throughout this thesis, and it integrates over the dual volume of a cell, it will from here on be known as "the dual method".

5.2.2 Integration over the cell itself

Instead of using the dual volume defined by neighbouring circumcenters, it seems logical to use the cell itself as the integration volume for the second-order correction. Define the approximation to the faces to be the weighted average of the velocities in the neighbouring circumcenters, i.e. $\tilde{u}_f = (1 - \alpha_f)u_c(i) + \alpha_f u_c(nb)$. Remember $\alpha_f = \frac{||x_{cL} - x_f||}{||x_{cL} - x_{cR}||}$, where cL is the upwind cell from f. Then we write, using Green's theorem, while keeping the coefficients $a_{ux1}, \ldots$ the same, a new

expression for E ($\vec{x}_{n1}, \vec{x}_{n2}$ are the two nodes of face f):

$$\tilde{u}_{cx} = \frac{1}{V_c}\sum_f((1-\alpha_f)u_c(i)+\alpha_f u_c(nb))(y_{n1}-y_{n2}) \quad (80)$$

$$\tilde{u}_{cy} = -\frac{1}{V_c}\sum_f((1-\alpha_f)u_c(i)+\alpha_f u_c(nb))(x_{n1}-x_{n2}) \quad (81)$$

$$\tilde{v}_{cx} = \frac{1}{V_c}\sum_f((1-\alpha_f)u_c(i)+\alpha_f u_c(nb))(y_{n1}-y_{n2}) \quad (82)$$

$$\tilde{v}_{cy} = -\frac{1}{V_c}\sum_f((1-\alpha_f)u_c(i)+\alpha_f u_c(nb))(x_{n1}-x_{n2}) \quad (83)$$

taking care that the integral is performed counterclockwise. The cell gets this contribution from itself and from each of its neighbours. The matrix formed is slightly different from the matrix A from the integration over the dual volume, but the rest of the algorithm is the same.

Looking ahead at the experiments in chapter 6, we see that it performs almost the same as the second order method that integrates over the dual volume. This justifies not programming the method in D-Flow FM and leaving it out of discussion after the Matlab experiments.

5.3 Least Squares solution

This method is, in contrast to the previous section, not a correction for Perot but an altogether different method. The least squares (LSQ) method works by minimizing the L^2-error $\sqrt{\sum_c \frac{V_c}{\sum_c V_c}(\vec{u}_c^{LSQ} - \vec{u}_c^{\text{exact}})^2}$. For each cell the algorithm selects the faces belonging to the cell and the faces adjacent to the cell's own faces, record their u_f, the normal vector $(n_x, n_y)^T$, for which we can just take the cosine and sine of the face, and the distance $(\Delta x, \Delta y)$ of the face center to the cell circumcenter. Each face gets a row in the following matrix, then the algorithm solves the system (this system is constructed and solved for each cell; it is possible to put it all in a big matrix but this may take more time and certainly takes a lot more memory) [20] to compute the velocity vector and its gradient.

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n_x & n_y & n_x\Delta x & n_y\Delta x & n_x\Delta y & n_y\Delta y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} u \\ v \\ \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial y} \end{pmatrix} = \vec{u}_f \quad (84)$$

For the advection vector add the columns $n_x^2\Delta x + n_y n_x \Delta y$ and $n_x n_y \Delta x + n_y^2 \Delta y$ to the matrix, while the second derivatives can be computed with the columns

$$\frac{\partial^2 u}{\partial x^2} \leftarrow n_x\Delta x\Delta x, \qquad \frac{\partial^2 u}{\partial y^2} \leftarrow n_x\Delta y\Delta y, \qquad \frac{\partial^2 v}{\partial x^2} \leftarrow n_y\Delta x\Delta x, \qquad \frac{\partial^2 v}{\partial y^2} \leftarrow n_y\Delta y\Delta y,$$

$$\frac{\partial^2 u}{\partial x\partial y} \leftarrow n_x\Delta x\Delta y, \qquad \frac{\partial^2 v}{\partial x\partial y} \leftarrow n_y\Delta x\Delta y \quad (85)$$

and the diffusion can be computed from the second derivatives: $(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2})$. Of course we assume here that the field is smooth enough, so we have equality of mixed partials. Since we need the system to be over-determined (more rows than columns, so more faces than columns), we do not want too many columns, hence we need to find out which quantities we want to compute in this matrix. The mixed partials help stabilize [20], but the advection and non-mixed second derivatives are not very accurate and have a negative influence on the accuracy of the velocity and gradient. Both advection and diffusion computed by this method are less accurate than the interpolated methods described above, which is why we will only use this method to find the velocity and gradients.

Note that, if the grid is close to regular, the normal vectors of different faces will be aligned with each other, and some rows in the matrix are multiples of each other, causing the matrix to be singular, and the system might not have a unique solution. Hence, in contrast to the other methods described, the least squares interpolation performs best on highly irregular grids.

5.4 Hybrid method

We can combine the derivations of the implicit method with the least squares method. Computing the velocities and first-order derivatives with a least squares method as described above, we can discard the velocities and substitute the derivatives in the expression E and get a corrected approximation for $(u, v)_c$:

$$\vec{u}_c^{\text{corrected}} = \vec{u}_c^{\text{Perot}} + E^{\text{LSQ}}. \tag{86}$$

where E^{LSQ} is just E as in section 5.2.1 but with the least squares derivatives. This is computationally less expensive than the implicit dual method, which needs to solve a larger matrix in each step, but a little more expensive than using Perot or computing everything with the least squares method. However, it might combine the best of both worlds and is hence worth investigating.

5.5 Other options for velocity reconstruction and alignment index

Other ways to design hybrid methods would be to use Perot's reconstruction on aligned cells and a least squares reconstruction on irregular pieces of the grid. For this it is necessary to define an alignment index [15, eqn. 28]. This index sums the differences in x- and y-coordinates of two faces opposite each other and divides them by the face length for all faces of a cell. Peixoto and Barros call a cell badly aligned if this index is bigger than 0.01, which is cause for them to use the least squares solution. They characterize their implementation of a hybrid scheme as 'effective'.

5.6 Boundary conditions

The second-order correction methods use, for each cell, the Perot-reconstructed $\vec{u}_c$ from neighbouring cells. Since those are not always available at the domain boundaries, we construct a layer of extra cells around the boundary called ghost cells, where we can prescribe the boundary conditions. Ghost cells are only defined by their circumcenter, which is defined by mirroring the circumcenter of the boundary cell in the domain wall. More precisely, the face coordinates are halfway between the two circumcenters, i.e. $\vec{x}_f = \frac{1}{2}(\vec{x}_{c(i)} + \vec{x}_{c(g)})$, and halfway between the cell corners: $\vec{x}_f = \frac{1}{2}(\vec{x}_{n1} + \vec{x}_{n2})$, where $c(i)$ is the interior cell and $c(g)$ the ghost cell. Therefore we can compute

$$\vec{x}_{c(g)} = \vec{x}_{n1} + \vec{x}_{n2} - \vec{x}_{c(i)}. \tag{87}$$

At a water-water boundary, we can just prescribe a velocity or water level for the ghost cells. At a land-water boundary, we need to take care of slip conditions. For a no-slip wall, the water in contact with the wall is not moving, and we can prescribe that $\vec{u}_{c(g)} = -\vec{u}_{c(i)}$, for a full-slip wall, there is no shear stress and we can prescribe $\vec{u}_{c(g)} = \vec{u}_{c(i)}$. For a partial slip wall, we apply the logarithmic law of the wall. This can all be programmed to be solved implicitly. More formally, we can look at the tangential velocity at the domain wall $\vec{u}_W$ and consider turbulence.

In a fully developed flow, the velocity has a logarithmic profile horizontally. In the fluid adjacent to the wall, the flow will be laminar (a viscous sublayer). Write the law of the wall as

$$\bar{u} = \frac{\vec{u}^* \cdot \vec{n}}{\kappa} \ln(1 + \frac{\delta_{xy}}{\delta_0}) \tag{88}$$

where $\kappa \approx 0.4$ is the von Karman constant, δ_{xy} is the distance from the wall and δ_0 is the thickness of the viscous sublayer, and we only look relatively close to the wall.

Since an extremely fine discretization in order to discretize the logarithm would be unwieldy, we define $\vec{u}^*$ as a cut-off point of the logarithmic function (at some point, use a tangent to the velocity profile and see where it intersects with $\delta_{xy} = 0$, this is $\vec{u}^*$) and use $\vec{u}_W = \vec{u}^*$ ($\vec{u}_W$ is the velocity at the wall). Now consider the equation

$$\lambda \vec{u}_W + L(1-\lambda)\frac{\partial \vec{u}_W}{\partial \vec{n}} = 0, \quad \lambda \in [0,1] \tag{89}$$

Then define the constant $w_p = \frac{\kappa}{\ln(1+\frac{\delta_{xy}}{\delta_0})}$ as the partial slip coefficient and solve $\vec{u}_{c(\text{boundary})} = (1-2w_p)\vec{u}_{c(\text{interior})}$. We do not actually compute $\vec{u}_{c(\cdot)}$ here, but this expression appears as a diagonal element in the matrix A for cells which are adjacent to the boundary. In summary, we define a general slip coefficient c to deal with all boundary conditions at the same time and compute the diagonal element in A for a cell adjacent to a closed boundary:

$$c = \begin{cases} -1 & \text{no slip (water at wall not moving)} \\ 2w_p - 1 & \text{partial slip} \\ 1 & \text{free slip (no shear stress at wall)} \end{cases} \tag{90}$$

$$\begin{aligned} A(i,i) = \sum_{nbcw,nbccw} & -a_{ux1} \cdot c \cdot (y_{nbccw} - y_{nbcw}) + a_{uy1} \cdot c \cdot (x_{nbccw} - x_{nbcw}) \\ & -a_{vx1} \cdot c \cdot (y_{nbccw} - y_{nbcw}) + a_{vy1} \cdot c \cdot (x_{nbccw} - x_{nbcw}) \\ & -a_{ux2} \cdot c \cdot (y_{nbccw} - y_{nbcw}) + a_{uy2} \cdot c \cdot (x_{nbccw} - x_{nbcw}) \\ & -a_{vx2} \cdot c \cdot (y_{nbccw} - y_{nbcw}) + a_{vy2} \cdot c \cdot (x_{nbccw} - x_{nbcw}) \end{aligned} \tag{91}$$

where x_{nbccw} is the circumcenter of the counterclockwise neighbour to cell i and x_{nbcw} is the circumcenter of the clockwise neighbour to cell i. The sum is taken over all pairs of neigbours of cell i that have each other as neighbour as well.

6 Experiments with a simplified model

6.1 The model

These tests make use of a simplified model of the Shallow Water Equations implemented in Matlab; the model takes files that describe a grid as input, an analytical velocity function which is applied to the u_f, and allows the user to choose a computational method for velocity vector reconstruction (Perot's method, the dual- and cell-integrated method, the least squares method LSQ and the hybrid method) and advection (the first order upwind FOU, second order upwind SOU, central method, α-weighted method and face-integrated using Simpson's rule FIS) are used. Then it reconstructs the velocity, its gradients, the advection and diffusion for the entire grid, but it does not take into account the time integration, water depth, the discretization of the bottom or the boundary conditions. As the stencil of the second order methods needs some boundary conditions, there are ghost cells introduced which consist only of a circumcenter. However, the velocities in the ghost cells are not reconstructed, but are all zero. Each velocity field is scaled with the length of the grid in x-direction $x_{\max} - x_{\min}$ and the length of the grid in y-direction $y_{\max} - y_{\min}$, which are both just the difference between the largest and smallest $x-$ resp. y-coordinate.

The matrix solver BiCGstab is chosen because it is stable for asymmetric sparse matrices, (the matrix A is not symmetric, not skew-symmetric and the sparsity pattern is not symmetric either), converges fast (in iterations; not necessarily in computing time) even when the matrix has a large bandwidth (this is of course dependent on the indexing of the cells, so we might be able to choose a faster solver / have BiCGstab converge faster if we permute the matrix entries such that the u and v are interlaced).

First we will discuss the various discretizations for advection and diffusion in the Matlab scripts, then proceed to the experiments.

6.1.1 First-order upwind advection

Assume cell 1 is upwind from cell 2, that is, there is flow from cell 1 entering cell 2. The cells then contribute to each others advection with the amount of $h_f u_{c1} \cdot u_f \cdot l_f s_{fc}$, which is added to the advection of cell 1 but subtracted from the advection of cell 2. For each cell, a summation over all neighbouring cells is performed, resulting in

$$Adv(\vec{x}_c) = \sum_{\text{faces of cell}} u_{c1} \cdot u_f \cdot l_f s_{fc} \tag{92}$$

and similar for the v-component.

6.1.2 Second-order upwind advection

In inner cells, the advection is computed as the sum over all faces of a first order Taylor approximation:

$$Adv(\vec{x}_c) = \sum_{\text{faces of cell}} (u_c + (x_f - x_c)\frac{du}{dx}\bigg|_c + (y_f - y_c)\frac{du}{dy}\bigg|_c) l_f u_f s_{fc} \tag{93}$$

and analogously for the v component. In boundary/ghost cells it is changed to second order downwind. An alternative would be using the gradients from the neighbouring interior cell with the u_c of the ghost cell, but the downwind choice is more consistent.

6.1.3 α-weighted method for advection

The weighing factor α is computed for each face by the formula $\alpha = \frac{\text{Area}(n_1, n_2, c_1)}{\text{Area}(n_1, n_2, c_1, c_2)}$, in other words: the area spanned by the circumcenter of the cell that has outflow through that face and the vertices of the volume divided by the area of both circumcenters and vertices belonging to that face. In well-behaved grids, $\alpha \in (0, 1)$ for all faces. However, if cells have an obtuse angle, the circumcenter may be outside of the cell, and α for the face opposite the biggest angle will be negative. In the α-method, cell 1 obtains advection in the amount of

$$Adv(\vec{x}_c) = \sum_{\text{faces of cell}} (\alpha \cdot u_{c1} + (1 - \alpha) \cdot u_{c2}) \cdot u_f \cdot l_f s_{fc}, \tag{94}$$

while the same amount is subtracted from the advection of the neighbouring cells, resp. for each face. Again, this is summed over all neighbouring cells and the calculation for the v-component is similar.

6.1.4 Central method for advection

The central method is obtained by setting each α equal to $\frac{1}{2}$ in the previous method. When the grid consists of squares, rectangles or equilateral triangles, this is exactly the same, but as the grid becomes more irregular, the central method deteriorates compared to the α method.

6.1.5 Face-integrated method (Simpson rule) for advection

This method uses a three-point numerical integration method on each face. The contributions in the endpoints n_1 and n_2 of the face are reconstructed using a first order Taylor polynomial (same for the v component)

$$u_n = u_c + (x_n - x_c) \cdot \left.\frac{du}{dx}\right|_c + (y_n - y_c) \cdot \left.\frac{du}{dy}\right|_c \tag{95}$$

while the contribution in the face center is given by

$$\tilde{u}_f = u_c + (x_f - x_c) \cdot \left.\frac{du}{dx}\right|_c + (y_f - y_c) \cdot \left.\frac{du}{dy}\right|_c \tag{96}$$

and again for the v component. The integral over the face is $\frac{1}{6}(u_{n_1} + 4\tilde{u}_f + u_{n_2})$. The advection of the cell is then the sum of the integrals over the faces, each multiplied with $u_f l_f$, hence the computation becomes

$$Adv(\vec{x}_c) = \sum_{\text{faces of cell}} u_f l_f \frac{1}{6}(u_{n_1} + 4\tilde{u}_f + u_{n_2}). \tag{97}$$

If the cell is a virtual cell, the gradients from the neighbouring inner cell are used.

6.1.6 Discrete advection: difficulties

Every one of these methods has its own advantages and drawbacks, though some are common to all methods. When we take the face values and integrate to get cell values, these can be seen as one point Gaussian quadrature, which has second order accuracy. If the face values are second order as well, we get a product of second order errors in the circumcenter, which has a cumulative effect

and the result is that, with a linear velocity field, we do not get exact results anymore. Also, we take the value $u_f = u(\vec{x}_f) \cdot n_f$, i.e. a point value. For linear velocity fields, the velocity actually varies along a face so this gives us a second order error (the first order error gets integrated out with the circle integral). This also assumes that the advection does not change sign within the cell. Taking the velocity along the entire face, though, would not be compatible with the discretization of the continuity equation. This means that we can use the gradients in x_c, but not those in x_f.

6.1.7 Analytical advection using Gauss quadrature

In the Matlab experiments we can specify a formula for the velocity field, allowing us to compare the reconstructed velocities and their gradients to the analytical value. This gives us several methods to compute the analytical advection: as point values, as integral over the cell, or as integral over the faces interpolated to the cell. Matlab can evaluate symbolic expressions and differentiate them, so if $(u(x,y),\, v(x,y))$ is given, Matlab can symbolically differentiate and then evaluate $Adv(\vec{x}_c) = \left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y},\, u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right)_c$ in the point $\vec{x_c}$.

Another method is using Gauss quadrature over the faces. First we write, using the divergence theorem (and a unit vector $\vec{n} = (\cos f,\, \sin f)$ (where the sine and cosine are of the angle between the face f and the x-axis) normal to the face f), that

$$V_c Adv(\vec{x}_c) = \iint_{\Omega_c} (\nabla \cdot \vec{u}\vec{u})\, d\Omega_c = \int_{\partial\Omega_c} \vec{u}\vec{u} \cdot \vec{n}\, ds = \sum_{\text{faces of } \Omega} \int_{\text{face}} \begin{pmatrix} u \\ v \end{pmatrix} \left(u\cos f + v \sin f\right) ds. \tag{98}$$

If the face has endpoints $(a_1,\, b_1)$ and $(a_2,\, b_2)$, we parametrize it with the curve

$$\vec{c}(t) = \frac{1}{2}\begin{pmatrix} a_1 + a_2 \\ b_1 + b_2 \end{pmatrix} + \frac{t}{2}\begin{pmatrix} a_2 - a_1 \\ b_2 - b_1 \end{pmatrix}, \quad -1 \le t \le 1. \tag{99}$$

Note that the point $\frac{1}{2}(a_1 + a_2, b_1 + b_2)$ is the midpoint $\vec{x}_f$ of the face. We then use the change of coordinates formula where $|\vec{c}'(t)| = \frac{1}{2}\sqrt{(a_2 - a_1)^2 + (b_2 - b_1)^2} = \frac{1}{2} l_f$ and the integral becomes

$$\sum_{\text{faces of } \Omega} |\vec{c}'(t)| \int_{\text{face}} \begin{pmatrix} u(\vec{c}(t)) \\ v(\vec{c}(t)) \end{pmatrix} \left(u(\vec{c}(t))\cos f + v(\vec{c}(t)) \sin f\right) ds. \tag{100}$$

We will approximate this last expression with a quadrature rule. An n-point Gaussian quadrature yields an exact result for polynomials of degree $2n - 1$ or less. The 3-point Gaussian quadrature over $[-1,\, 1]$ is defined on the points $0, \pm\sqrt{\frac{3}{5}}$ with weights $\frac{8}{9}, \frac{5}{9}$, respectively. The integral of the polynomial $g(x)$ with degree 5 or less is then equal to

$$\int g(x)\, dx = \int_{-1}^{1} g(x)\, dx = \frac{5}{9} g(-\sqrt{\frac{3}{5}}) + \frac{8}{9} g(0) + \frac{5}{9} g(\sqrt{\frac{3}{5}}). \tag{101}$$

Applying this to the advection integral gives

$$\begin{aligned} V_c Adv(\vec{x}_c) \;=\; & \sum_{\text{faces of cell}} \frac{l_f}{2} \left(\frac{8}{9} \begin{pmatrix} u(x_f) \\ v(y_f) \end{pmatrix} \left(u(x_f)\cos f,\, v(y_f)\sin f\right) \right. \\ & \left. + \frac{5}{9} \begin{pmatrix} u(\frac{1}{2}(a_1 + a_2) \pm \frac{1}{2}\sqrt{\frac{3}{5}}(a_2 - a_1)) \\ v(\frac{1}{2}(b_1 + b_2) \pm \frac{1}{2}\sqrt{\frac{3}{5}}(b_2 - b_1)) \end{pmatrix} \begin{bmatrix} u\left(\frac{1}{2}(a_1 + a_2) \pm \frac{1}{2}\sqrt{\frac{3}{5}}(a_2 - a_1)\right)\cos f + ... \\ ... + v\left(\frac{1}{2}(b_1 + b_2) \pm \frac{1}{2}\sqrt{\frac{3}{5}}(b_2 - b_1)\right)\sin f \end{bmatrix} \right) \end{aligned} \tag{102}$$

Now the right hand side is fifth order accurate, but dividing by the cell surface costs two orders, hence $Adv(x_c)$ is third order accurate.

A third possible approximation is using a Gaussian quadrature in two dimensions, hence over the entire cell. An n-simplex S_n has $n+1$ vertices x_i (so for a triangle, $n=2$). First find the centroid $G = \sum_{i=1}^{n+1} \frac{x_i}{n+1}$ and let V be the volume of S_n. Then define the points $y_i = \frac{2}{n+3}x_i + \frac{n+1}{n+3}G$ and compute the coefficients $a_n = \frac{(n+3)^2}{4(n+1)(n+2)}V$ and $c_n = \frac{-(n+1)^2}{4(n+2)}V$. For a cubic polynomial g, the following quadrature rule is exact [11]:

$$\int_{S_n} g\,dV = c_n g(G) + a_n \sum_{i=1}^{n+1} g(y_i). \tag{103}$$

The second method (3-point method on the faces) is the one that was used in the Matlab experiments of chapter 6 in order to find the analytical values of the advection and determine the errors in the different advection discretizations.

6.1.8 Diffusion

The diffusion is given by the Laplacian $\nabla^2\vec{u} = \nabla\cdot(\nabla\vec{u}) = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y})^T = \frac{\partial^2 u}{dx^2} + \frac{\partial^2 u}{dy^2}$. In inner cells, the net flux through the boundary is zero and we get, using Green's thm,

$$\int_{\partial V} \nabla\vec{u}\cdot\vec{n}\,ds = 0 \qquad \Leftrightarrow \qquad \int_V \nabla^2\vec{u}\,dV = 0 \tag{104}$$

Since diffusion spreads out equally in all directions, we can use a central approximation for inner cells if the gradients of u are available. Then the diffusion is the sum over all faces:

$$\frac{1}{2}\sum_{\text{faces of cell}} s_{fc}\left(\left.\frac{\partial u}{\partial x}\right|_c + \left.\frac{\partial u}{\partial x}\right|_{nb} + \left.\frac{\partial u}{\partial y}\right|_c + \left.\frac{\partial u}{\partial y}\right|_{nb}\right) l_f \tag{105}$$

and similar for the v-component. The sign indicates that what is added to the diffusion of one cell via face f, will be subtracted from the cell neighbouring at face f, because the cells have opposite orientations (*not* because of mass conservation, which is the mechanism behind the same result at the advection calculation). This discretization will be employed in the Matlab experiments.

6.2 Test description

We run experiments on two small grids (see figure 3), trying all velocity reconstruction methods: Perot, second order over the dual volume, second order over the cell itself, least squares reconstruction and the hybrid method. We can compute the gradients for Perot's method in the exact same way as $\tilde{u}_x$ etc in the dual method. The least squares method computes only u_c, v_c, u_{cx}, v_{cx}, u_{cy}, v_{cy}, but not advection or second derivatives, as this requires adding extra columns to the matrix, which decreases accuracy. The velocity fields are the constant field $(u,v) = (2,3)$, the linear field $(u,v) = -2(y-y_{\min})/(y_{\max}-y_{\min}), 3(x-x_{\min})/(x_{\max}-x_{\min})$ and the quadratic field $u = -3((y-y_{\min})^2)/((y_{\max}-y_{\min})^2)$, $v = 2(x-x_{\min})/(x_{\max}-x_{\min})$, on a fully triangular grid and a grid consisting of triangles and rectangles (named "mixed" in the table).

All matrix equations are solved by the BiCGStab method with a tolerance of 10^{-12}. For least squares, it needs to be mentioned that the method cannot reconstruct the velocities in the four

corner cells of the triangular grid: the matrix-vector equation does not have a solution and these cells are left out of the error calculation. This is due to the matrix being row-deficient, which may be caused by the corner cells having a right angle.

Table 1: Matlab L^∞ errors for three fields on two maps

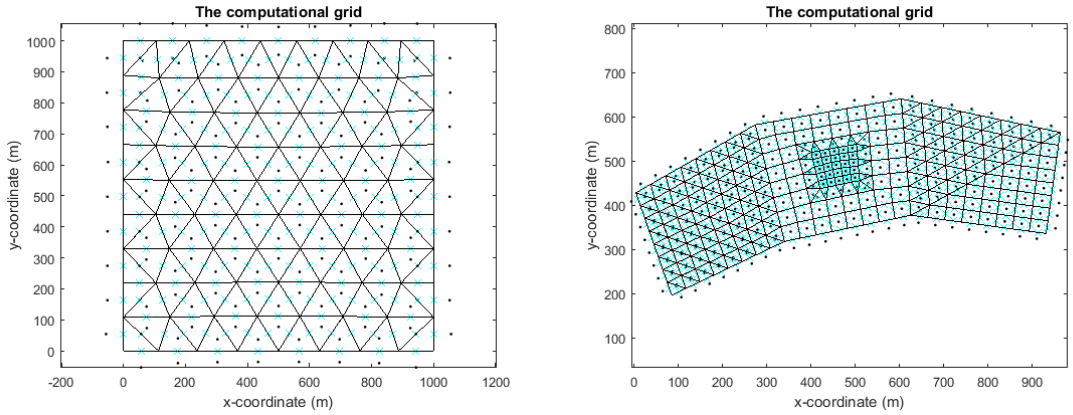
Triangular grid						Mixed grid					
Constant velocities:						Constant velocities:					
	Perot	Dual	Cell	LSQ	Hybrid		Perot	Dual	Cell	LSQ	Hybrid
u_c	$8{\cdot}10^{-15}$	$3{\cdot}10^{-11}$	$2{\cdot}10^{-10}$	$5{\cdot}10^{-11}$	$8{\cdot}10^{-15}$	u_c	$1{\cdot}10^{-1}$	$4{\cdot}10^{-2}$	$1{\cdot}10^{-1}$	$4{\cdot}10^{-10}$	$1{\cdot}10^{-1}$
v_c	$8{\cdot}10^{-15}$	$2{\cdot}10^{-11}$	$8{\cdot}10^{-10}$	$1{\cdot}10^{-10}$	$8{\cdot}10^{-15}$	v_c	$2{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$2{\cdot}10^{-10}$	$2{\cdot}10^{-1}$
$\frac{\partial u}{\partial x}$	$8{\cdot}10^{-17}$	$4{\cdot}10^{-13}$	$1.5{\cdot}10^{-12}$	$9{\cdot}10^{-14}$	0	$\frac{\partial u}{\partial x}$	$4{\cdot}10^{-3}$	$4{\cdot}10^{-3}$	$4{\cdot}10^{-3}$	$1{\cdot}10^{-11}$	0
$\frac{\partial v}{\partial x}$	$1{\cdot}10^{-16}$	$2.5{\cdot}10^{-13}$	$8{\cdot}10^{-12}$	$1{\cdot}10^{-12}$	0	$\frac{\partial v}{\partial x}$	$6{\cdot}10^{-3}$	$6{\cdot}10^{-3}$	$6{\cdot}10^{-3}$	$1{\cdot}10^{-10}$	0
$\frac{\partial u}{\partial y}$	$6{\cdot}10^{-17}$	$3{\cdot}10^{-13}$	$2{\cdot}10^{-12}$	$6{\cdot}10^{-13}$	0	$\frac{\partial u}{\partial y}$	$4{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$1{\cdot}10^{-11}$	0
$\frac{\partial v}{\partial y}$	$8{\cdot}10^{-17}$	$2{\cdot}10^{-13}$	$6{\cdot}10^{-12}$	$3{\cdot}10^{-13}$	0	$\frac{\partial v}{\partial y}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$4{\cdot}10^{-12}$	0
Linear velocities:						Linear velocities:					
	Perot	Dual	Cell	LSQ	Hybrid		Perot	Dual	Cell	LSQ	Hybrid
u_c	$2{\cdot}10^{-2}$	$2{\cdot}10^{-12}$	$10{\cdot}10^{-11}$	$2{\cdot}10^{-11}$	$2{\cdot}10^{-2}$	u_c	$3{\cdot}10^{-2}$	$2{\cdot}10^{-2}$	$3{\cdot}10^{-2}$	$1{\cdot}10^{-11}$	$3{\cdot}10^{-2}$
v_c	$4{\cdot}10^{-3}$	$2{\cdot}10^{-12}$	$4{\cdot}10^{-10}$	$8{\cdot}10^{-11}$	$4{\cdot}10^{-3}$	v_c	$4{\cdot}10^{-2}$	$4{\cdot}10^{-2}$	$6{\cdot}10^{-2}$	$2{\cdot}10^{-11}$	$4{\cdot}10^{-2}$
$\frac{\partial u}{\partial x}$	$1.5{\cdot}10^{-4}$	$2{\cdot}10^{-14}$	$8{\cdot}10^{-13}$	$7{\cdot}10^{-14}$	0	$\frac{\partial u}{\partial x}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$1{\cdot}10^{-13}$	0
$\frac{\partial v}{\partial x}$	$3{\cdot}10^{-5}$	$2.5{\cdot}10^{-14}$	$4{\cdot}10^{-12}$	$1{\cdot}10^{-12}$	0	$\frac{\partial v}{\partial x}$	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$6{\cdot}10^{-13}$	0
$\frac{\partial u}{\partial y}$	$2{\cdot}10^{-4}$	$2{\cdot}10^{-14}$	$1{\cdot}10^{-12}$	$1{\cdot}10^{-12}$	0	$\frac{\partial u}{\partial y}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$6{\cdot}10^{-13}$	0
$\frac{\partial v}{\partial y}$	$3{\cdot}10^{-5}$	$3{\cdot}10^{-14}$	$3{\cdot}10^{-12}$	$8{\cdot}10^{-14}$	0	$\frac{\partial v}{\partial y}$	$1.5{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$4{\cdot}10^{-13}$	0
Quadratic velocities:						Quadratic velocities:					
	Perot	Dual	Cell	LSQ	Hybrid		Perot	Dual	Cell	LSQ	Hybrid
u_c	$4{\cdot}10^{-2}$	$9{\cdot}10^{-3}$	$7{\cdot}10^{-3}$	$4{\cdot}10^{-3}$	$6{\cdot}10^{-2}$	u_c	$8{\cdot}10^{-2}$	$8{\cdot}10^{-2}$	$8{\cdot}10^{-2}$	$1{\cdot}10^{-3}$	$8{\cdot}10^{-2}$
v_c	$1.5{\cdot}10^{-2}$	$8{\cdot}10^{-4}$	$3{\cdot}10^{-3}$	$7{\cdot}10^{-3}$	$1.5{\cdot}10^{-2}$	v_c	$6{\cdot}10^{-2}$	$6{\cdot}10^{-2}$	$7{\cdot}10^{-2}$	$2{\cdot}10^{-3}$	$6{\cdot}10^{-2}$
$\frac{\partial u}{\partial x}$	$4{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$6{\cdot}10^{-5}$	0	$\frac{\partial u}{\partial x}$	$2.5{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$1{\cdot}10^{-4}$	0
$\frac{\partial v}{\partial x}$	$1{\cdot}10^{-4}$	$6{\cdot}10^{-6}$	$3{\cdot}10^{-5}$	$1{\cdot}10^{-4}$	0	$\frac{\partial v}{\partial x}$	$1.5{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	0
$\frac{\partial u}{\partial y}$	$6{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$4{\cdot}10^{-3}$	$6{\cdot}10^{-3}$	$\frac{\partial u}{\partial y}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	$5{\cdot}10^{-4}$	$12{\cdot}10^{-3}$
$\frac{\partial v}{\partial y}$	$1{\cdot}10^{-4}$	$6{\cdot}10^{-6}$	$3{\cdot}10^{-5}$	$5{\cdot}10^{-5}$	0	$\frac{\partial v}{\partial y}$	$2{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$1{\cdot}10^{-4}$	0

The table 1 gives the largest numerical error L^∞ for each test case. The advection and diffusion will be deferred to section 6.4.

If we interpolate u_f back from u_c again (constructing the u_f for the next timestep, but without accounting for ζ and other quantities), using the formula $\tilde{u}_f = (\alpha u_{c1} + (1-\alpha)u_{c2}) \cdot \vec{n}$, we expect the difference with the original u_f to be small. For a linear velocity field, this difference is near machine precision for Perot's method, but the higher-order reconstructions do not form a conservative field. In contrast to the u_f, the $\tilde{u}_f$ do not form a conservative field. In figure 4, the difference between u_f and $\tilde{u}_f$ is plotted for the linear field on the mixed grid.

6.3 Results

We can see from the table 1 that any higher-order reconstruction is an improvement on Perot's velocity reconstruction. For a constant or linear field on the fully triangular grid, the dual and cell-



(a) Triangular grid (137 cells)

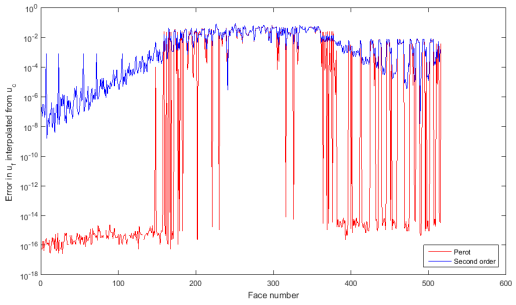
(b) Mixed grid (329 cells)

Figure 3: Grids for the Matlab experiments, black dots represent cell circumcenters $\vec{x}_c$, cyan crosses represent face centers $\vec{x}_f$

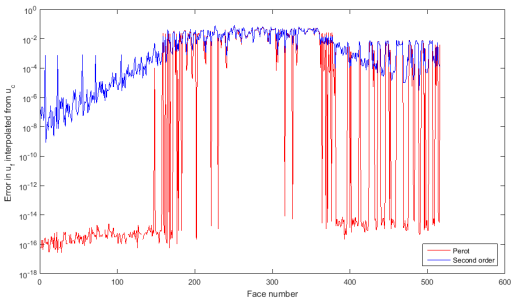
centered methods come close to the tolerance for the matrix solver, while Perot still has errors of $O(10^{-2})$. For the mixed grid, the differences are not as spectacular, but the second order methods still improve on Perot. As the velocity field is of order 1, the maximum error in table 1 is 0.2 for u_c for the constant field with Perot's reconstruction, where $u = 2$, which means that the error is around 10%. However, most errors are only fractions of percentages.

Comparing the errors over the whole grid, it is clear that the methods that integrate over the dual volume and the cell itself are almost the same, and smaller than solver precision. This justifies programming only one of them in Fortran for the tests in D-Flow FM. The least squares reconstruction seems to be the best choice for a constant or linear field, though its quality declines when choosing a higher order field. It also becomes unstable when time-stepping is used in D-Flow FM, see section 7. Furthermore, all methods seem to perform best when only triangles are involved, this may be caused by the fact that the circumcenter is not uniquely defined for quadrilaterals (though it is unique for rectangles). It is also possible that the transition between two kinds of cells causes an extra disturbance. It is noteworthy that the gradients for the second order methods are more accurate than the velocity field; a possible explanation for this is the fact that the BiCGStab method, while it is only employed to solve for (u_c, v_c), it uses in its algorithm the residual which is based on a gradient.

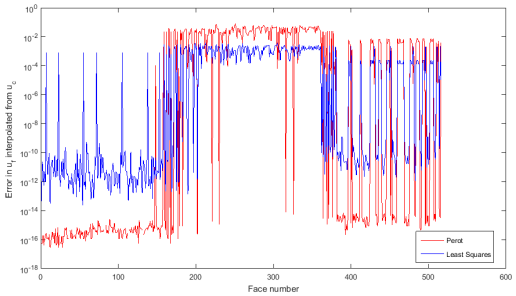
Since the second order methods are meant to improve the velocity, advection and diffusion discretization, we will proceed with a convergence study. As the dual- and cell-integrated methods perform equally well, only one of these needs to be included. The full least squares method is also added, as it performs better than the hybrid method where the velocities are concerned.



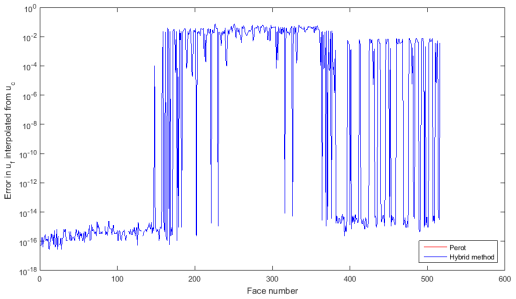
(a) Dual



(b) Cell



(c) Least squares



(d) Hybrid (note that the line for hybrid method coincides with the line for Perot)

Figure 4: Error in $\tilde{u}_f$; interpolated from u_c for the mixed grid with linear velocity field.

6.4 Convergence for advection methods

In this section the convergence of the five advection methods (first-order upwind FOU, second-order upwind SOU, Central, α-weighted and integrated over the face using Simpson's rule FIS) and the diffusion is studied over grids of different sizes and related to the convergence of the velocity and the velocity gradient.

6.4.1 Test description

For this test, we have three grids of triangular cells, one with 15 cells, one with 60 cells (the 15 cells split in 4), and one with 240 cells that is again a refinement of the 60-cell grid (see figure 5). A triangle is split in four triangles by connecting the midpoints of the faces. The grids are not regular since the center of mass of each triangle does not coincide with the circumcenter, hence Perot's velocity reconstruction is not second order accurate. As the circumcenter is uniquely defined for all triangles, but not for all quadrilaterals, only triangles are studied.

Both the dual correction method and the least squares are computed with BiCGstab with a tolerance of $1{\cdot}10^{-12}$.

Note that the velocity field is divergence free, which may also have a positive effect on the results. This is, however, necessary for mass conservation.

We expect linear convergence for the first order advection methods: first order upwind, central and alpha and quadratic convergence for the second order advection methods: second order upwind and Simpson's rule over the faces, but in the methods we assume u_f to be constant, while it varies over the face. When making the integral over $\partial\Omega$, we get rid of the first order error but not of the second order error in u_f. This error gets multiplied with the velocity and gradient, hence the convergence of these methods should, for second order upwind and Simpson's rule, be better than linear but probably not quadratic.

To determine the convergence, the errors

$$L^\infty = \max_{1\le i\le \#\text{cells}} (y_i - f(x_i)) \tag{106}$$

$$L^1_w = \frac{1}{\sum_{i=1}^{\#\text{cells}} V_{c(i)}} \sum_{i=1}^{\#\text{cells}} |y_i - f(x_i)| V_{c(i)} \tag{107}$$

$$L^2_w = \sqrt{\frac{1}{\sum_{i=1}^{\#\text{cells}} V_{c(i)}} \sum_{i=1}^{\#\text{cells}} (y_i - f(x_i))^2 V_{c(i)}} \tag{108}$$

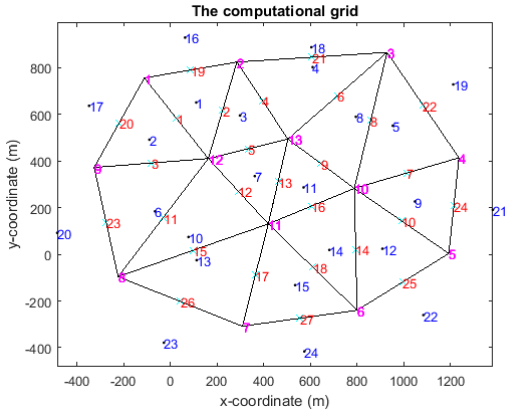
where y_i are the measured values in the circumcenter of cell i and $f(x_i)$ are the analytical velocities in the point x_c are computed and recorded in tables.

We can find convergence rates (using some error $L \in \left\{L^\infty, L^1_w, L^2_w\right\}$) by solving the equation

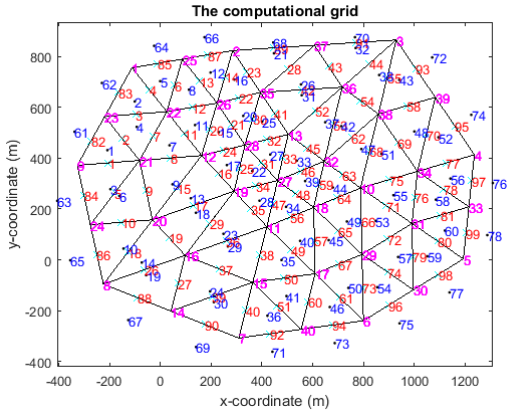
$$L(\text{coarse grid}) = 2^r L(\text{fine grid}) \tag{109}$$

for r, where the fine grid is one step refined with respect to the coarse grid; i.e. the convergence rate between the 15-cell grid and the 60-cell grid, and the rate between the 60-cell grid and the 240-cell grid are computed. This gives $r = \log(L(\text{coarse grid})/L(\text{fine grid}))/\log(2)$, where the logarithm is the natural logarithm.

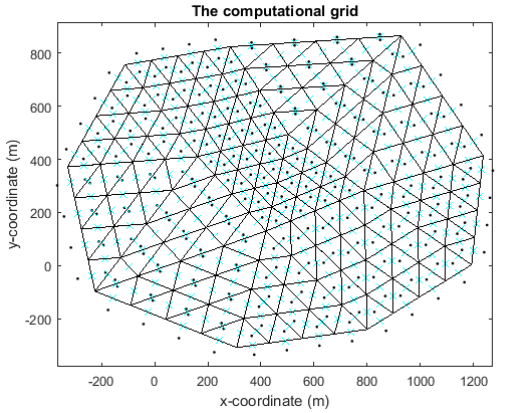
Figure 5: Grids for the advection convergence tests



(a) 15 cells



(b) 60 cells



(c) 240 cells

Negative values imply no convergence, if the convergence rate r is 1, we have found first order convergence, if $r = 2$ we have second order convergence, and so forth. This method assumes that the error does not change sign within the cell.

The tolerance of BiCGStab is 10^{-12}, hence all errors of that size or smaller can be considered machine precision. For machine precision errors, convergence is not possible, which will be indicated in the convergence tables using the marker "s.p." (for solver precision). When the rate is negative, there is no convergence, which will be indicated with the marker " no conv.".

6.4.2 Constant velocity field

A constant velocity field gives solver-precision errors for all velocity reconstruction methods and all computed quantities on a triangular grid (see table 1), hence a convergence study would not be interesting and we proceed directly to the linear velocity field.

6.4.3 Linear velocity field: results

For the linear velocity field $(u, v) = -2(y - y_{\min})/(y_{\max} - y_{\min}), 3(x - x_{\min})/(x_{\max} - x_{\min})$. with Perot's reconstruction, assuming that the grid is not regular, the first order error term E exists, but the second order error term does not. Since we compute the gradient for Perot's velocity reconstruction using $\tilde{u}^P_{cx} = \frac{1}{V}\sum_f (u^P_{cw} + u^P_{ccw})(y_{ccw} - y_{cw})$, the gradient may have a zeroth order error. We know u_f has a first order error, which drops out when we take the sum, but the overall error in the first order upwind will still be first order. In second order upwind, we may get zeroth order, the α and central will have first order, and the face-integrated method may have zeroth order errors.

For second order reconstruction, $\vec{u}_c = \vec{u}^P_c + E$ is exact, but the gradients have a first order error, allowing a better performance for the second-order upwind and the face-integrated method.

Table 2 shows the L^∞ error, table 3 gives the L^1 error, table 4 gives the L^2 error and table 5 gives the convergence rates in L^1 and L^∞. The tables show solver precision errors for the second order velocities and their gradients, and first order convergence for Perot's velocity reconstruction (but sublinear in L^1 for the gradients). The first order upwind advection method does not seem to converge, the second order upwind converges for the dual and least squares velocity reconstruction in both L^1 and L^∞. The central and α-weighted advection methods converge sublinearly in L^1 for Perot and the dual method but not for the least squares velocity method and not in L^∞. The face-integrated method using Simpson's rule, which gives the same advection field as the second order upwind method for Perot's and the dual reconstruction method, only converges for the dual method. The diffusion calculation has solver precision in the second order methods for the L^1 error, it converges sublinearly in L^∞ for Perot's velocity reconstruction and linearly in L^∞ for the second order methods.

In conclusion, table 5 shows that using two second order methods on top of each other (i.e. dual velocity reconstruction together with second order upwind advection) gives only first order convergence.

6.4.4 Quadratic velocity field: results

For a quadratic field, we cannot reasonably expect solver precision errors, but we can expect convergence. Table 6 shows the L^∞ error, table 7 gives the L^1 error, table 8 gives the L^2 error and

table 9 gives the convergence rates in L^1 and L^∞. The velocity field is $u = -3((y-y_{\min})^2)/((y_{\max}-y_{\min})^2)$, $v = 2(x-x_{\min})/(x_{\max}-x_{\min})$.

The tables show almost linear (sometimes sublinear, but occassionally better than linear) convergence for Perot's velocities and almost quadratic convergence for second order velocities. The gradients for Perot converge sublinearly in L^1 but not in L^∞. The gradients for the second order methods show almost linear behaviour.

First order upwind advection does not show convergence, while second order upwind advection shows almost linear convergence in L^1 for the second order methods, and in L^∞ for the dual method. The central and α advection methods show sublinear convergence for L^1 but not for L^∞. The face-integrated method using Simpson's rule shows sublinear convergence for L^∞ and the diffusion has sublinear convergence for L^∞, sublinear convergence for the L^1 error in the second order velocity methods but no L^1-convergence for Perot's velocity reconstruction method.

None of the advection methods is particularly good here but the α-method seems to work best for Perot's velocity reconstruction, and the second order upwind advection should probably be used for the second order velocity methods.

Table 2: Advection convergence for a linear field on three grids: L^∞ error:

	Perot			Dual			least squares		
	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells
u_c	$2.8{\cdot}10^{-2}$	$1.4{\cdot}10^{-2}$	$7.2{\cdot}10^{-3}$	$2.1{\cdot}10^{-12}$	$9.5{\cdot}10^{-12}$	$1.6{\cdot}10^{-11}$	$6.6{\cdot}10^{-16}$	$1.8{\cdot}10^{-11}$	$2.0{\cdot}10^{-11}$
v_c	$1.8{\cdot}10^{-2}$	$1.2{\cdot}10^{-2}$	$6.2{\cdot}10^{-3}$	$1.0{\cdot}10^{-12}$	$7.1{\cdot}10^{-12}$	$1.7{\cdot}10^{-11}$	$1.1{\cdot}10^{-15}$	$6.1{\cdot}10^{-11}$	$2.2{\cdot}10^{-11}$
$\frac{\partial u}{\partial x}$	$5.4{\cdot}10^{-5}$	$6.3{\cdot}10^{-5}$	$6.4{\cdot}10^{-5}$	$3.6{\cdot}10^{-15}$	$3.6{\cdot}10^{-14}$	$1.6{\cdot}10^{-13}$	$5.6{\cdot}10^{-18}$	$4.7{\cdot}10^{-14}$	$7.3{\cdot}10^{-14}$
$\frac{\partial u}{\partial y}$	$6.5{\cdot}10^{-5}$	$1.1{\cdot}10^{-4}$	$1.1{\cdot}10^{-4}$	$5.7{\cdot}10^{-15}$	$7.5{\cdot}10^{-14}$	$2.3{\cdot}10^{-13}$	$2.2{\cdot}10^{-18}$	$6.3{\cdot}10^{-14}$	$2.4{\cdot}10^{-13}$
$\frac{\partial v}{\partial x}$	$6.0{\cdot}10^{-5}$	$6.4{\cdot}10^{-5}$	$6.4{\cdot}10^{-5}$	$2.4{\cdot}10^{-15}$	$3.7{\cdot}10^{-14}$	$2.3{\cdot}10^{-13}$	$1.5{\cdot}10^{-18}$	$4.2{\cdot}10^{-13}$	$1.2{\cdot}10^{-13}$
$\frac{\partial v}{\partial y}$	$4.0{\cdot}10^{-5}$	$5.1{\cdot}10^{-5}$	$5.9{\cdot}10^{-5}$	$1.7{\cdot}10^{-15}$	$5.8{\cdot}10^{-14}$	$2.0{\cdot}10^{-13}$	$1.2{\cdot}10^{-18}$	$8.5{\cdot}10^{-14}$	$2.1{\cdot}10^{-13}$
FOU-adv(x)	$2.9{\cdot}10^{-3}$	$3.2{\cdot}10^{-3}$	$3.4{\cdot}10^{-3}$	$3.0{\cdot}10^{-3}$	$3.3{\cdot}10^{-3}$	$3.4{\cdot}10^{-3}$	$3.5{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$2.9{\cdot}10^{-2}$
FOU-adv(y)	$2.2{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$	$2.7{\cdot}10^{-2}$	$2.4{\cdot}10^{-3}$	$2.8{\cdot}10^{-3}$	$2.9{\cdot}10^{-3}$	$2.5{\cdot}10^{-2}$	$6.1{\cdot}10^{-2}$	$1.3{\cdot}10^{-1}$
SOU-adv(x)	$5.4{\cdot}10^{-4}$	$7.8{\cdot}10^{-4}$	$8.6{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$1.0{\cdot}10^{-4}$	$5.0{\cdot}10^{-5}$	$2.0{\cdot}10^{-4}$	$1.0{\cdot}10^{-4}$	$5.0{\cdot}10^{-5}$
SOU-adv(y)	$7.4{\cdot}10^{-4}$	$8.2{\cdot}10^{-4}$	$9.6{\cdot}10^{-4}$	$2.9{\cdot}10^{-4}$	$1.5{\cdot}10^{-4}$	$7.4{\cdot}10^{-5}$	$2.9{\cdot}10^{-4}$	$1.5{\cdot}10^{-4}$	$7.4{\cdot}10^{-5}$
Central adv(x)	$1.6{\cdot}10^{-3}$	$1.8{\cdot}10^{-3}$	$1.9{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$1.8{\cdot}10^{-3}$	$1.9{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$2.6{\cdot}10^{-2}$	$5.6{\cdot}10^{-2}$
Central adv(y)	$1.2{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$3.1{\cdot}10^{-2}$	$6.7{\cdot}10^{-2}$
α-adv(x)	$3.2{\cdot}10^{-3}$	$3.6{\cdot}10^{-3}$	$3.8{\cdot}10^{-3}$	$3.2{\cdot}10^{-3}$	$3.5{\cdot}10^{-3}$	$3.8{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$2.7{\cdot}10^{-2}$	$5.6{\cdot}10^{-2}$
α-adv(y)	$2.2{\cdot}10^{-3}$	$2.1{\cdot}10^{-3}$	$2.2{\cdot}10^{-3}$	$2.1{\cdot}10^{-3}$	$2.1{\cdot}10^{-3}$	$2.2{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$3.1{\cdot}10^{-2}$	$6.7{\cdot}10^{-2}$
FIS-adv(x)	$5.4{\cdot}10^{-4}$	$7.8{\cdot}10^{-4}$	$8.6{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$1.0{\cdot}10^{-4}$	$5.0{\cdot}10^{-5}$	$4.6{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$3.0{\cdot}10^{-2}$
FIS-adv(y)	$7.4{\cdot}10^{-4}$	$8.1{\cdot}10^{-4}$	$9.5{\cdot}10^{-4}$	$3.0{\cdot}10^{-4}$	$1.4{\cdot}10^{-4}$	$7.4{\cdot}10^{-5}$	$2.8{\cdot}10^{-2}$	$6.4{\cdot}10^{-2}$	$1.3{\cdot}10^{-1}$
diff(x)	1.09	0.57	0.29	1.10	0.55	0.27	1.10	0.55	0.27
diff(y)	0.91	0.45	0.23	0.90	0.45	0.22	0.90	0.45	0.22

Table 3: Advection convergence for a linear field on three grids: weighted L^1 error:

	Perot			Dual			least squares		
	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells
u_c	$1.1{\cdot}10^{-2}$	$5.9{\cdot}10^{-3}$	$2.9{\cdot}10^{-3}$	$7.8{\cdot}10^{-12}$	$9.8{\cdot}10^{-11}$	$5.1{\cdot}10^{-10}$	$1.6{\cdot}10^{-14}$	$9.2{\cdot}10^{-11}$	$2.9{\cdot}10^{-10}$
v_c	$1.1{\cdot}10^{-2}$	$5.3{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$	$5.9{\cdot}10^{-12}$	$1.1{\cdot}10^{-10}$	$5.6{\cdot}10^{-10}$	$8.4{\cdot}10^{-15}$	$1.9{\cdot}10^{-10}$	$5.1{\cdot}10^{-10}$
$\frac{\partial u}{\partial x}$	$2.3{\cdot}10^{-5}$	$1.6{\cdot}10^{-5}$	$9.9{\cdot}10^{-6}$	$2.3{\cdot}10^{-14}$	$5.8{\cdot}10^{-13}$	$6.2{\cdot}10^{-12}$	$1.2{\cdot}10^{-17}$	$2.1{\cdot}10^{-13}$	$1.1{\cdot}10^{-12}$
$\frac{\partial u}{\partial y}$	$2.3{\cdot}10^{-5}$	$1.8{\cdot}10^{-5}$	$1.2{\cdot}10^{-5}$	$2.5{\cdot}10^{-14}$	$6.3{\cdot}10^{-13}$	$6.9{\cdot}10^{-12}$	$4.6{\cdot}10^{-17}$	$1.6{\cdot}10^{-13}$	$1.8{\cdot}10^{-12}$
$\frac{\partial v}{\partial x}$	$2.7{\cdot}10^{-5}$	$1.5{\cdot}10^{-5}$	$8.5{\cdot}10^{-6}$	$1.7{\cdot}10^{-14}$	$7.5{\cdot}10^{-13}$	$6.7{\cdot}10^{-12}$	$1.7{\cdot}10^{-17}$	$5.9{\cdot}10^{-13}$	$1.6{\cdot}10^{-12}$
$\frac{\partial v}{\partial y}$	$1.6{\cdot}10^{-5}$	$1.3{\cdot}10^{-5}$	$8.2{\cdot}10^{-6}$	$1.6{\cdot}10^{-14}$	$7.0{\cdot}10^{-13}$	$7.8{\cdot}10^{-12}$	$1.5{\cdot}10^{-17}$	$3.2{\cdot}10^{-13}$	$1.4{\cdot}10^{-12}$
FOU-adv(x)	$1.1{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$1.0{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$
FOU-adv(y)	$8.1{\cdot}10^{-4}$	$8.7{\cdot}10^{-4}$	$9.1{\cdot}10^{-4}$	$8.6{\cdot}10^{-4}$	$9.1{\cdot}10^{-4}$	$9.4{\cdot}10^{-4}$	$4.7{\cdot}10^{-3}$	$4/9{\cdot}10^{-3}$	$5{\cdot}10^{-3}$
SOU-adv(x)	$2.1{\cdot}10^{-4}$	$1.9{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$8.8{\cdot}10^{-5}$	$4.4{\cdot}10^{-5}$	$2.2{\cdot}10^{-5}$	$8.8{\cdot}10^{-5}$	$4.4{\cdot}10^{-5}$	$2.2{\cdot}10^{-5}$
SOU-adv(y)	$2.0{\cdot}10^{-4}$	$2.2{\cdot}10^{-4}$	$2.2{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$	$6.1{\cdot}10^{-5}$	$3.1{\cdot}10^{-5}$	$1.2{\cdot}10^{-4}$	$6.1{\cdot}10^{-5}$	$3.1{\cdot}10^{-5}$
Central adv(x)	$4.9{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$	$1.5{\cdot}10^{-4}$	$4.5{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$	$2.2{\cdot}10^{-3}$	$2.1{\cdot}10^{-3}$	$2.0{\cdot}10^{-3}$
Central adv(y)	$3.8{\cdot}10^{-4}$	$2.3{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$	$3.6{\cdot}10^{-4}$	$2.2{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$	$3.2{\cdot}10^{-3}$	$3.3{\cdot}10^{-3}$	$3.3{\cdot}10^{-3}$
α-adv(x)	$9.1{\cdot}10^{-4}$	$5.0{\cdot}10^{-4}$	$2.8{\cdot}10^{-4}$	$8.6{\cdot}10^{-4}$	$4.7{\cdot}10^{-4}$	$2.6{\cdot}10^{-4}$	$2.6{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$2.1{\cdot}10^{-3}$
α-adv(y)	$7.3{\cdot}10^{-4}$	$4.3{\cdot}10^{-4}$	$2.4{\cdot}10^{-4}$	$7.0{\cdot}10^{-4}$	$4.2{\cdot}10^{-4}$	$2.4{\cdot}10^{-4}$	$3.3{\cdot}10^{-3}$	$3.5{\cdot}10^{-3}$	$3.4{\cdot}10^{-3}$
FIS-adv(x)	$2.0{\cdot}10^{-4}$	$1.9{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$8.8{\cdot}10^{-5}$	$4.4{\cdot}10^{-5}$	$2.2{\cdot}10^{-5}$	$5.4{\cdot}10^{-4}$	$5.6{\cdot}10^{-4}$	$6.0{\cdot}10^{-4}$
FIS-adv(y)	$2.1{\cdot}10^{-4}$	$2.3{\cdot}10^{-4}$	$2.2{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$	$6.1{\cdot}10^{-5}$	$3.1{\cdot}10^{-5}$	$4.3{\cdot}10^{-3}$	$4.3{\cdot}10^{-3}$	$4.3{\cdot}10^{-3}$
diff(x)	$1.2{\cdot}10^{-7}$	$2.2{\cdot}10^{-7}$	$3.1{\cdot}10^{-7}$	$1.1{\cdot}10^{-17}$	$1.5{\cdot}10^{-16}$	$7.8{\cdot}10^{-16}$	$1.9{\cdot}10^{-20}$	$4.2{\cdot}10^{-17}$	$1.4{\cdot}10^{-16}$
diff(y)	$1.3{\cdot}10^{-7}$	$1.7{\cdot}10^{-7}$	$2.3{\cdot}10^{-7}$	$7.5{\cdot}10^{-18}$	$1.7{\cdot}10^{-16}$	$8.5{\cdot}10^{-16}$	$4.6{\cdot}10^{-21}$	$1.0{\cdot}10^{-16}$	$1.3{\cdot}10^{-16}$

Table 4: Advection convergence for a linear field on three grids: weighted L^2 error:

	Perot			Dual			least squares		
	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells
u_c	$2.1{\cdot}10^{-4}$	$5.3{\cdot}10^{-5}$	$1.3{\cdot}10^{-5}$	$1.1{\cdot}10^{-23}$	$3.9{\cdot}10^{-22}$	$3.1{\cdot}10^{-21}$	$1.3{\cdot}10^{-28}$	$1.1{\cdot}10^{-21}$	$1.8{\cdot}10^{-21}$
v_c	$1.5{\cdot}10^{-4}$	$3.7{\cdot}10^{-5}$	$9.3{\cdot}10^{-6}$	$3.5{\cdot}10^{-24}$	$3.9{\cdot}10^{-22}$	$3.1{\cdot}10^{-21}$	$7.6{\cdot}10^{-30}$	$6.7{\cdot}10^{-21}$	$7.2{\cdot}10^{-21}$
$\frac{\partial u}{\partial x}$	$7.8{\cdot}10^{-10}$	$5.6{\cdot}10^{-10}$	$3.5{\cdot}10^{-10}$	$7.2{\cdot}10^{-29}$	$1.0{\cdot}10^{-26}$	$4.2{\cdot}10^{-25}$	$3.6{\cdot}10^{-35}$	$6.9{\cdot}10^{-27}$	$2.7{\cdot}10^{-26}$
$\frac{\partial u}{\partial y}$	$1.0{\cdot}10^{-9}$	$8.9{\cdot}10^{-10}$	$5.3{\cdot}10^{-10}$	$9.5{\cdot}10^{-29}$	$1.9{\cdot}10^{-26}$	$5.8{\cdot}10^{-25}$	$1.4{\cdot}10^{-33}$	$5.2{\cdot}10^{-27}$	$1.3{\cdot}10^{-25}$
$\frac{\partial v}{\partial x}$	$1.1{\cdot}10^{-9}$	$4.9{\cdot}10^{-10}$	$2.3{\cdot}10^{-10}$	$3.2{\cdot}10^{-29}$	$1.3{\cdot}10^{-26}$	$4.6{\cdot}10^{-25}$	$4.3{\cdot}10^{-35}$	$1.8{\cdot}10^{-25}$	$8.9{\cdot}10^{-26}$
$\frac{\partial v}{\partial y}$	$4.3{\cdot}10^{-10}$	$3.5{\cdot}10^{-10}$	$2.1{\cdot}10^{-10}$	$2.3{\cdot}10^{-29}$	$1.7{\cdot}10^{-26}$	$5.4{\cdot}10^{-25}$	$6.6{\cdot}10^{-35}$	$2.2{\cdot}10^{-26}$	$7.7{\cdot}10^{-26}$
FOU-adv(x)	$1.6{\cdot}10^{-6}$	$1.9{\cdot}10^{-6}$	$2.2{\cdot}10^{-6}$	$1.6{\cdot}10^{-6}$	$1.9{\cdot}10^{-6}$	$2.1{\cdot}10^{-6}$	$2.4{\cdot}10^{-6}$	$5.6{\cdot}10^{-6}$	$1.2{\cdot}10^{-5}$
FOU-adv(y)	$1.0{\cdot}10^{-6}$	$1.1{\cdot}10^{-6}$	$1.2{\cdot}10^{-6}$	$1.1{\cdot}10^{-6}$	$1.2{\cdot}10^{-6}$	$1.3{\cdot}10^{-6}$	$8.3{\cdot}10^{-5}$	$1.7{\cdot}10^{-4}$	$3.4{\cdot}10^{-4}$
SOU-adv(x)	$6.7{\cdot}10^{-8}$	$6.6{\cdot}10^{-8}$	$6.9{\cdot}10^{-8}$	$1.2{\cdot}10^{-8}$	$2.9{\cdot}10^{-9}$	$7.4{\cdot}10^{-10}$	$1.2{\cdot}10^{-8}$	$2.9{\cdot}10^{-9}$	$7.5{\cdot}10^{-10}$
SOU-adv(y)	$7.6{\cdot}10^{-8}$	$7.6{\cdot}10^{-8}$	$7.6{\cdot}10^{-8}$	$2.5{\cdot}10^{-8}$	$6.3{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$	$2.5{\cdot}10^{-8}$	$6.3{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$
Central adv(x)	$3.5{\cdot}10^{-7}$	$1.8{\cdot}10^{-7}$	$8.8{\cdot}10^{-8}$	$3.5{\cdot}10^{-7}$	$1.6{\cdot}10^{-7}$	$7.6{\cdot}10^{-8}$	$1.5{\cdot}10^{-5}$	$2.9{\cdot}10^{-5}$	$5.7{\cdot}10^{-5}$
Central adv(y)	$2.2{\cdot}10^{-7}$	$1.1{\cdot}10^{-7}$	$4.8{\cdot}10^{-8}$	$2.2{\cdot}10^{-7}$	$9.4{\cdot}10^{-8}$	$4.3{\cdot}10^{-8}$	$2.9{\cdot}10^{-5}$	$6.3{\cdot}10^{-5}$	$1.3{\cdot}10^{-4}$
α-adv(x)	$1.4{\cdot}10^{-6}$	$6.7{\cdot}10^{-7}$	$3.2{\cdot}10^{-7}$	$1.4{\cdot}10^{-6}$	$6.3{\cdot}10^{-7}$	$3.0{\cdot}10^{-7}$	$1.7{\cdot}10^{-5}$	$3.0{\cdot}10^{-5}$	$5.8{\cdot}10^{-5}$
α-adv(y)	$8.5{\cdot}10^{-7}$	$3.8{\cdot}10^{-7}$	$1.7{\cdot}10^{-7}$	$7.8{\cdot}10^{-7}$	$3.6{\cdot}10^{-7}$	$1.7{\cdot}10^{-7}$	$2.8{\cdot}10^{-5}$	$6.2{\cdot}10^{-5}$	$1.3{\cdot}10^{-4}$
FIS-adv(x)	$6.3{\cdot}10^{-8}$	$6.6{\cdot}10^{-8}$	$6.9{\cdot}10^{-8}$	$1.2{\cdot}10^{-8}$	$2.9{\cdot}10^{-9}$	$7.4{\cdot}10^{-10}$	$1.5{\cdot}10^{-6}$	$4.2{\cdot}10^{-6}$	$1.0{\cdot}10^{-5}$
FIS-adv(y)	$8.3{\cdot}10^{-8}$	$7.9{\cdot}10^{-8}$	$7.8{\cdot}10^{-8}$	$2.5{\cdot}10^{-8}$	$6.3{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$	$9.1{\cdot}10^{-5}$	$1.7{\cdot}10^{-4}$	$3.5{\cdot}10^{-4}$
diff(x)	$2.9{\cdot}10^{-14}$	$1.0{\cdot}10^{-13}$	$2.2{\cdot}10^{-13}$	$2.6{\cdot}10^{-34}$	$4.9{\cdot}10^{-32}$	$1.7{\cdot}10^{-30}$	$1.6{\cdot}10^{-39}$	$6.8{\cdot}10^{-33}$	$7.9{\cdot}10^{-32}$
diff(y)	$2.8{\cdot}10^{-14}$	$5.6{\cdot}10^{-14}$	$1.0{\cdot}10^{-13}$	$8.3{\cdot}10^{-35}$	$4.9{\cdot}10^{-32}$	$1.6{\cdot}10^{-30}$	$4.8{\cdot}10^{-41}$	$9.7{\cdot}10^{-32}$	$7.1{\cdot}10^{-32}$

Table 5: Convergence rate r for linear field on three grids

Convergence rates based on L_w^1:						
	Perot		Dual		Least squares	
u_c	1	1	s.p.	s.p.	s.p.	s.p.
v_c	1	1	s.p.	s.p.	s.p.	s.p.
$\frac{\partial u}{\partial x}$	0.525	0.712	s.p.	s.p.	s.p.	s.p.
$\frac{\partial u}{\partial y}$	0.325	0.619	s.p.	s.p.	s.p.	s.p.
$\frac{\partial v}{\partial x}$	0.834	0.830	s.p.	s.p.	s.p.	s.p.
$\frac{\partial v}{\partial y}$	0.319	0.686	s.p.	s.p.	s.p.	s.p.
FOU-adv(x)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
FOU-adv(y)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
SOU-adv(x)	0.135	no conv.	1	1	1	1
SOU-adv(y)	no conv.	0.0088	1	1	1	1
Central adv(x)	0.843	0.849	0.874	0.877	0.083	0.047
Central adv(y)	0.712	0.823	0.702	0.812	no conv.	0.014
α-adv(x)	0.852	0.866	0.867	0.87	0.155	0.092
α-adv(y)	0.748	0.858	0.731	0.847	no conv.	0.028
FIS-adv(x)	0.049	no conv.	1	1	no conv.	no conv.
FIS-adv(y)	no conv.	0.032	1	1	0.017	0.015
diff(x)	no conv.	no conv.	s.p.	s.p.	s.p.	s.p.
diff(y)	no conv.	no conv.	s.p.	s.p.	s.p.	s.p.

Convergence rates based on L^∞:						
	Perot		Dual		LSQ	
u_c	1	1	s.p.	s.p.	s.p.	s.p.
v_c	1	1	s.p.	s.p.	s.p.	s.p.
$\frac{\partial u}{\partial x}$	no conv.	no conv.	s.p.	s.p.	s.p.	s.p.
$\frac{\partial u}{\partial y}$	no conv.	no conv.	s.p.	s.p.	s.p.	s.p.
$\frac{\partial v}{\partial x}$	no conv.	no conv.	s.p.	s.p.	s.p.	s.p.
$\frac{\partial v}{\partial y}$	no conv.	no conv.	s.p.	s.p.	s.p.	s.p.
FOU-adv(x)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
FOU-adv(y)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
SOU-adv(x)	no conv.	no conv.	1	1	1	1
SOU-adv(y)	no conv.	no conv.	1	1	1	1
Central adv(x)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
Central adv(y)	no conv.	no conv.	0.099	no conv.	no conv.	no conv.
α-adv(x)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
α-adv(y)	0.093	no conv.	0.013	no conv.	no conv.	no conv.
FIS-adv(x)	no conv.	no conv.	1	1	no conv.	no conv.
FIS-adv(y)	no conv.	no conv.	1	1	no conv.	no conv.
diff(x)	0.935	0.985	1	1	1	1
diff(y)	0.998	1	1	1	1	1

Table 6: Advection convergence for a quadratic field on three grids: L^∞ error:

	Perot			Dual			least squares		
	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells
u_c	$4.2{\cdot}10^{-1}$	$2.3{\cdot}10^{-1}$	$1.2{\cdot}10^{-1}$	$9.0{\cdot}10^{-2}$	$7.1{\cdot}10^{-4}$	$4.7{\cdot}10^{-4}$	$1.3{\cdot}10^{-1}$	$4.2{\cdot}10^{-2}$	$1.0{\cdot}10^{-2}$
v_c	$2.9{\cdot}10^{-1}$	$1.8{\cdot}10^{-1}$	$1.0{\cdot}10^{-1}$	$2.0{\cdot}10^{-2}$	$1.2{\cdot}10^{-2}$	$3.3{\cdot}10^{-3}$	$5.7{\cdot}10^{-2}$	$1.2{\cdot}10^{-2}$	$3.0{\cdot}10^{-3}$
$\frac{\partial u}{\partial x}$	$5.3{\cdot}10^{-4}$	$9.2{\cdot}10^{-4}$	$1.1{\cdot}10^{-3}$	$4.3{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$	$4.0{\cdot}10^{-4}$	$2.3{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$
$\frac{\partial u}{\partial y}$	$1.2{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$5.8{\cdot}10^{-4}$	$4.0{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$1.3{\cdot}10^{-3}$	$7.9{\cdot}10^{-4}$	$3.9{\cdot}10^{-4}$
$\frac{\partial v}{\partial x}$	$8.0{\cdot}10^{-4}$	$7.5{\cdot}10^{-4}$	$8.2{\cdot}10^{-4}$	$7.6{\cdot}10^{-5}$	$6.6{\cdot}10^{-5}$	$2.8{\cdot}10^{-5}$	$2.1{\cdot}10^{-4}$	$9.8{\cdot}10^{-5}$	$4.9{\cdot}10^{-5}$
$\frac{\partial v}{\partial y}$	$4.5{\cdot}10^{-4}$	$5.9{\cdot}10^{-4}$	$7.2{\cdot}10^{-4}$	$4.9{\cdot}10^{-5}$	$4.6{\cdot}10^{-5}$	$2.2{\cdot}10^{-5}$	$4.6{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$
FOU-adv(x)	$4.7{\cdot}10^{-3}$	$8.0{\cdot}10^{-3}$	$9.3{\cdot}10^{-3}$	$5.7{\cdot}10^{-3}$	$9.3{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$5.7{\cdot}10^{-3}$	$2.8{\cdot}10^{-2}$	$7.0{\cdot}10^{-2}$
FOU-adv(y)	$6.1{\cdot}10^{-3}$	$9.1{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$1.3{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$2.6{\cdot}10^{-2}$	$5.6{\cdot}10^{-2}$
SOU-adv(x)	$3.4{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$1.3{\cdot}10^{-2}$	$1.8{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$7.2{\cdot}10^{-4}$	$1.7{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$7.7{\cdot}10^{-4}$
SOU-adv(y)	$5.8{\cdot}10^{-3}$	$9.4{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$3.4{\cdot}10^{-4}$	$2.3{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$	$5.3{\cdot}10^{-4}$	$6.6{\cdot}10^{-4}$	$4.8{\cdot}10^{-4}$
Central adv(x)	$3.0{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$3.1{\cdot}10^{-3}$	$2.8{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$1.2{\cdot}10^{-2}$	$2.9{\cdot}10^{-2}$	$6.1{\cdot}10^{-2}$
Central adv(y)	$1.3{\cdot}10^{-3}$	$2.4{\cdot}10^{-3}$	$2.9{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.2{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$5.9{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$2.9{\cdot}10^{-2}$
α-adv(x)	$4.3{\cdot}10^{-3}$	$3.8{\cdot}10^{-3}$	$3.7{\cdot}10^{-3}$	$3.8{\cdot}10^{-3}$	$4.0{\cdot}10^{-3}$	$4.0{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$3.0{\cdot}10^{-2}$	$6.2{\cdot}10^{-2}$
α-adv(y)	$1.5{\cdot}10^{-3}$	$2.0{\cdot}10^{-3}$	$2.2{\cdot}10^{-3}$	$1.9{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$2.5{\cdot}10^{-3}$	$5.8{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$2.9{\cdot}10^{-2}$
FIS-adv(x)	$3.4{\cdot}10^{-3}$	$1.1{\cdot}10^{-2}$	$1.3{\cdot}10^{-2}$	$1.8{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$7.4{\cdot}10^{-4}$	$4.6{\cdot}10^{-3}$	$2.4{\cdot}10^{-2}$	$6.6{\cdot}10^{-2}$
FIS-adv(y)	$5.8{\cdot}10^{-3}$	$9.4{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$4.2{\cdot}10^{-4}$	$2.4{\cdot}10^{-4}$	$1.4{\cdot}10^{-4}$	$1.3{\cdot}10^{-2}$	$2.7{\cdot}10^{-2}$	$5.7{\cdot}10^{-2}$
diff(x)	2.92	1.50	0.78	2.84	1.53	0.795	2.24	1.35	0.751
diff(y)	0.559	0.293	0.157	0.606	0.303	0.151	0.695	0.314	0.153

Table 7: Advection convergence for a quadratic field on three grids: weighted L^1 error:

	Perot			Dual			least squares		
	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells
u_c	$1.0{\cdot}10^{-1}$	$4.8{\cdot}10^{-2}$	$2.3{\cdot}10^{-2}$	$8.2{\cdot}10^{-1}$	$7.4{\cdot}10^{-1}$	$7.2{\cdot}10^{-1}$	1.02	1.15	1.16
v_c	$6.5{\cdot}10^{-2}$	$3.4{\cdot}10^{-2}$	$1.7{\cdot}10^{-2}$	$1.3{\cdot}10^{-1}$	$1.8{\cdot}10^{-1}$	$1.9{\cdot}10^{-1}$	$2.4{\cdot}10^{-1}$	$1.9{\cdot}10^{-1}$	$1.6{\cdot}10^{-1}$
$\frac{\partial u}{\partial x}$	$3.1{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$	$3.3{\cdot}10^{-3}$	$6.3{\cdot}10^{-3}$	$1.3{\cdot}10^{-2}$	$2.1{\cdot}10^{-3}$	$5.4{\cdot}10^{-3}$	$1.2{\cdot}10^{-2}$
$\frac{\partial u}{\partial y}$	$4.0{\cdot}10^{-4}$	$2.9{\cdot}10^{-4}$	$1.7{\cdot}10^{-4}$	$4.4{\cdot}10^{-3}$	$9.9{\cdot}10^{-3}$	$2.0{\cdot}10^{-2}$	$9.1{\cdot}10^{-3}$	$1.4{\cdot}10^{-2}$	$2.5{\cdot}10^{-2}$
$\frac{\partial v}{\partial x}$	$2.2{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$	$7.1{\cdot}10^{-5}$	$4.5{\cdot}10^{-4}$	$1.1{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$1.8{\cdot}10^{-3}$	$2.5{\cdot}10^{-3}$
$\frac{\partial v}{\partial y}$	$1.7{\cdot}10^{-4}$	$1.4{\cdot}10^{-4}$	$9.1{\cdot}10^{-5}$	$4.1{\cdot}10^{-4}$	$9.1{\cdot}10^{-4}$	$1.3{\cdot}10^{-3}$	$2.2{\cdot}10^{-3}$	$5.5{\cdot}10^{-3}$	$1.2{\cdot}10^{-2}$
FOU-adv(x)	$1.2{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$1.4{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$2.2{\cdot}10^{-3}$	$2.4{\cdot}10^{-3}$
FOU-adv(y)	$1.4{\cdot}10^{-3}$	$1.4{\cdot}10^{-3}$	$1.4{\cdot}10^{-3}$	$3.7{\cdot}10^{-4}$	$3.8{\cdot}10^{-4}$	$4.1{\cdot}10^{-4}$	$2.5{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$
SOU-adv(x)	$9.4{\cdot}10^{-4}$	$1.2{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$6.0{\cdot}10^{-4}$	$3.2{\cdot}10^{-4}$	$1.6{\cdot}10^{-4}$	$8.3{\cdot}10^{-4}$	$4.1{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$
SOU-adv(y)	$1.1{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$1.4{\cdot}10^{-4}$	$7.1{\cdot}10^{-5}$	$3.3{\cdot}10^{-5}$	$1.9{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$	$6.2{\cdot}10^{-5}$
Central adv(x)	$8.5{\cdot}10^{-4}$	$6.1{\cdot}10^{-4}$	$3.8{\cdot}10^{-4}$	$8.5{\cdot}10^{-4}$	$5.2{\cdot}10^{-4}$	$2.9{\cdot}10^{-4}$	$2.4{\cdot}10^{-3}$	$2.5{\cdot}10^{-3}$	$2.4{\cdot}10^{-3}$
Central adv(y)	$4.8{\cdot}10^{-4}$	$3.5{\cdot}10^{-4}$	$2.1{\cdot}10^{-4}$	$2.3{\cdot}10^{-4}$	$1.4{\cdot}10^{-4}$	$8.4{\cdot}10^{-5}$	$1.6{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$
α-adv(x)	$1.2{\cdot}10^{-3}$	$7.6{\cdot}10^{-4}$	$4.5{\cdot}10^{-4}$	$1.3{\cdot}10^{-3}$	$8.1{\cdot}10^{-4}$	$4.6{\cdot}10^{-4}$	$2.7{\cdot}10^{-3}$	$2.7{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$
α-adv(y)	$4.6{\cdot}10^{-4}$	$3.7{\cdot}10^{-4}$	$2.4{\cdot}10^{-4}$	$4.3{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$	$1.5{\cdot}10^{-4}$	$1.7{\cdot}10^{-3}$	$1.8{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$
FIS-adv(x)	$9.6{\cdot}10^{-4}$	$1.2{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$6.6{\cdot}10^{-4}$	$3.3{\cdot}10^{-4}$	$1.7{\cdot}10^{-4}$	$1.2{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$9.7{\cdot}10^{-4}$
FIS-adv(y)	$1.2{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$1.4{\cdot}10^{-3}$	$1.4{\cdot}10^{-4}$	$7.2{\cdot}10^{-5}$	$3.3{\cdot}10^{-5}$	$2.4{\cdot}10^{-3}$	$2.4{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$
diff(x)	$1.7{\cdot}10^{-6}$	$2.4{\cdot}10^{-6}$	$2.8{\cdot}10^{-6}$	$1.5{\cdot}10^{-6}$	$1.4{\cdot}10^{-11}$	$5.3{\cdot}10^{-7}$	$3.4{\cdot}10^{-6}$	$1.8{\cdot}10^{-6}$	$1.0{\cdot}10^{-6}$
diff(y)	$1.1{\cdot}10^{-6}$	$1.5{\cdot}10^{-6}$	$1.8{\cdot}10^{-6}$	$1.8{\cdot}10^{-7}$	$7.4{\cdot}10^{-12}$	$1.4{\cdot}10^{-7}$	$4.3{\cdot}10^{-7}$	$3.4{\cdot}10^{-7}$	$2.5{\cdot}10^{-7}$

Table 8: Advection convergence for a quadratic field on three grids: weighted L^2 error:

	Perot			Dual			least squares		
	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells	15 cells	60 cells	240 cells
u_c	$2.2{\cdot}10^{-2}$	$5.4{\cdot}10^{-3}$	$1.4{\cdot}10^{-3}$	$5.2{\cdot}10^{-2}$	$1.1{\cdot}10^{-2}$	$2.5{\cdot}10^{-3}$	$8.5{\cdot}10^{-2}$	$2.5{\cdot}10^{-2}$	$6.2{\cdot}10^{-3}$
v_c	$9.8{\cdot}10^{-3}$	$2.6{\cdot}10^{-3}$	$6.6{\cdot}10^{-4}$	$1.7{\cdot}10^{-3}$	$1.0{\cdot}10^{-3}$	$3.0{\cdot}10^{-4}$	$6.6{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$2.2{\cdot}10^{-4}$
$\frac{\partial u}{\partial x}$	$1.1{\cdot}10^{-7}$	$7.0{\cdot}10^{-8}$	$4.4{\cdot}10^{-8}$	$9.7{\cdot}10^{-7}$	$8.8{\cdot}10^{-7}$	$8.8{\cdot}10^{-7}$	$5.4{\cdot}10^{-7}$	$6.7{\cdot}10^{-7}$	$7.8{\cdot}10^{-7}$
$\frac{\partial u}{\partial y}$	$2.6{\cdot}10^{-7}$	$1.7{\cdot}10^{-7}$	$9.4{\cdot}10^{-8}$	$1.7{\cdot}10^{-6}$	$2.1{\cdot}10^{-6}$	$2.2{\cdot}10^{-6}$	$7.4{\cdot}10^{-6}$	$4.9{\cdot}10^{-6}$	$3.9{\cdot}10^{-6}$
$\frac{\partial v}{\partial x}$	$1.2{\cdot}10^{-7}$	$6.0{\cdot}10^{-8}$	$2.9{\cdot}10^{-8}$	$1.9{\cdot}10^{-8}$	$3.3{\cdot}10^{-8}$	$2.1{\cdot}10^{-8}$	$2.2{\cdot}10^{-7}$	$1.4{\cdot}10^{-7}$	$9.3{\cdot}10^{-8}$
$\frac{\partial v}{\partial y}$	$4.4{\cdot}10^{-8}$	$4.3{\cdot}10^{-8}$	$2.6{\cdot}10^{-8}$	$1.7{\cdot}10^{-8}$	$2.5{\cdot}10^{-8}$	$1.4{\cdot}10^{-8}$	$5.5{\cdot}10^{-7}$	$6.8{\cdot}10^{-7}$	$7.9{\cdot}10^{-7}$
FOU-adv(x)	$2.7{\cdot}10^{-6}$	$4.8{\cdot}10^{-6}$	$6.2{\cdot}10^{-6}$	$4.5{\cdot}10^{-6}$	$6.9{\cdot}10^{-6}$	$8.1{\cdot}10^{-6}$	$5.5{\cdot}10^{-6}$	$2.3{\cdot}10^{-5}$	$4.4{\cdot}10^{-5}$
FOU-adv(y)	$4.5{\cdot}10^{-6}$	$5.1{\cdot}10^{-6}$	$5.8{\cdot}10^{-6}$	$2.6{\cdot}10^{-7}$	$2.6{\cdot}10^{-7}$	$2.7{\cdot}10^{-7}$	$1.8{\cdot}10^{-5}$	$3.9{\cdot}10^{-5}$	$7.9{\cdot}10^{-5}$
SOU-adv(x)	$1.8{\cdot}10^{-6}$	$5.6{\cdot}10^{-6}$	$8.5{\cdot}10^{-6}$	$6.4{\cdot}10^{-7}$	$1.8{\cdot}10^{-7}$	$4.9{\cdot}10^{-8}$	$9.0{\cdot}10^{-7}$	$2.4{\cdot}10^{-7}$	$6.4{\cdot}10^{-8}$
SOU-adv(y)	$4.3{\cdot}10^{-6}$	$5.0{\cdot}10^{-6}$	$5.6{\cdot}10^{-6}$	$3.1{\cdot}10^{-8}$	$8.1{\cdot}10^{-9}$	$1.7{\cdot}10^{-9}$	$6.6{\cdot}10^{-8}$	$2.4{\cdot}10^{-8}$	$7.2{\cdot}10^{-9}$
Central adv(x)	$1.5{\cdot}10^{-6}$	$7.6{\cdot}10^{-7}$	$4.1{\cdot}10^{-7}$	$1.7{\cdot}10^{-6}$	$6.3{\cdot}10^{-7}$	$2.4{\cdot}10^{-7}$	$1.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-5}$	$7.9{\cdot}10^{-5}$
Central adv(y)	$4.4{\cdot}10^{-7}$	$3.8{\cdot}10^{-7}$	$2.4{\cdot}10^{-7}$	$1.2{\cdot}10^{-7}$	$5.2{\cdot}10^{-8}$	$2.2{\cdot}10^{-8}$	$6.1{\cdot}10^{-6}$	$1.4{\cdot}10^{-5}$	$2.9{\cdot}10^{-5}$
α-adv(x)	$3.1{\cdot}10^{-6}$	$1.3{\cdot}10^{-6}$	$6.0{\cdot}10^{-7}$	$3.7{\cdot}10^{-6}$	$1.6{\cdot}10^{-6}$	$7.1{\cdot}10^{-7}$	$2.1{\cdot}10^{-5}$	$4.2{\cdot}10^{-5}$	$8.0{\cdot}10^{-5}$
α-adv(y)	$4.1{\cdot}10^{-7}$	$3.5{\cdot}10^{-7}$	$2.0{\cdot}10^{-7}$	$4.1{\cdot}10^{-7}$	$1.9{\cdot}10^{-7}$	$8.4{\cdot}10^{-8}$	$6.0{\cdot}10^{-6}$	$1.4{\cdot}10^{-5}$	$2.9{\cdot}10^{-5}$
FIS-adv(x)	$1.8{\cdot}10^{-6}$	$5.6{\cdot}10^{-6}$	$8.5{\cdot}10^{-6}$	$6.8{\cdot}10^{-7}$	$1.9{\cdot}10^{-7}$	$5.1{\cdot}10^{-8}$	$2.9{\cdot}10^{-6}$	$1.4{\cdot}10^{-5}$	$3.2{\cdot}10^{-5}$
FIS-adv(y)	$4.4{\cdot}10^{-6}$	$5.2{\cdot}10^{-6}$	$5.8{\cdot}10^{-6}$	$3.5{\cdot}10^{-8}$	$8.4{\cdot}10^{-9}$	$1.8{\cdot}10^{-9}$	$2.0{\cdot}10^{-5}$	$4.1{\cdot}10^{-5}$	$8.1{\cdot}10^{-5}$
diff(x)	$4.2{\cdot}10^{-12}$	$1.4{\cdot}10^{-11}$	$3.2{\cdot}10^{-11}$	$3.1{\cdot}10^{-12}$	$1.3{\cdot}10^{-12}$	$6.7{\cdot}10^{-13}$	$1.3{\cdot}10^{-11}$	$5.9{\cdot}10^{-12}$	$2.9{\cdot}10^{-12}$
diff(y)	$3.2{\cdot}10^{-12}$	$7.4{\cdot}10^{-12}$	$1.4{\cdot}10^{-11}$	$5.2{\cdot}10^{-14}$	$7.7{\cdot}10^{-14}$	$3.8{\cdot}10^{-14}$	$3.7{\cdot}10^{-13}$	$2.2{\cdot}10^{-13}$	$1.5{\cdot}10^{-13}$

Table 9: Convergence rate r for quadratic field

Convergence rates based on L_w^1:						
	Perot		Dual		Least squares	
u_c	1.067	1.025	2.153	2.056	1.803	1.978
v_c	0.947	0.992	1.493	1.899	2.265	2.269
$\frac{\partial u}{\partial x}$	0.635	0.734	1.006	0.963	0.474	0.820
$\frac{\partial u}{\partial y}$	0.481	0.727	0.854	0.955	1.395	1.150
$\frac{\partial v}{\partial x}$	0.808	0.856	0.703	1.534	1.726	1.514
$\frac{\partial v}{\partial y}$	0.267	0.618	0.792	1.486	0.594	0.833
FOU-adv(x)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
FOU-adv(y)	0.0447	0.0003	no conv.	no conv.	no conv.	no conv.
SOU-adv(x)	no conv.	no conv.	0.919	0.965	1.015	1.016
SOU-adv(y)	no conv.	no conv.	0.985	1.094	0.742	0.889
Central adv(x)	0.485	0.693	0.703	0.827	no conv.	0.0721
Central adv(y)	0.445	0.713	0.663	0.790	no conv.	0.018
α-adv(x)	0.627	0.736	0.670	0.820	no conv.	0.106
α-adv(y)	0.281	0.660	0.696	0.837	no conv.	0.0347
FIS-adv(x)	no conv.	no conv.	0.989	0.9995	0.144	0.165
FIS-adv(y)	no conv.	no conv.	0.948	1.103	0.0237	0.0401
diff(x)	no conv.	no conv.	0.802	0.734	0.939	0.807
diff(y)	no conv.	no conv.	no conv.	0.542	0.331	0.461

Convergence rates based on L^∞:						
	Perot		Dual		LSQ	
u_c	0.839	0.926	2.005	1.971	1.660	2
v_c	0.640	0.849	0.722	1.879	2.262	1.989
$\frac{\partial u}{\partial x}$	no conv.	no conv.	0.916	0.935	0.864	1
$\frac{\partial u}{\partial y}$	no conv.	no conv.	0.544	0.990	0.796	1
$\frac{\partial v}{\partial x}$	0.0825	no conv.	0.205	0.877	1.111	1
$\frac{\partial v}{\partial y}$	no conv.	no conv.	0.486	1.007	0.864	1
FOU-adv(x)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
FOU-adv(y)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
SOU-adv(x)	no conv.	no conv.	0.518	0.782	0.538	0.558
SOU-adv(y)	no conv.	no conv.	0.598	0.839	no conv.	0.460
Central adv(x)	0.404	no conv.	0.121	0.138	no conv.	no conv.
Central adv(y)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
α-adv(x)	0.178	0.0542	no conv.	0.00603	no conv.	no conv.
α-adv(y)	no conv.	no conv.	no conv.	no conv.	no conv.	no conv.
FIS-adv(x)	no conv.	no conv.	0.491	0.831	no conv.	no conv.
FIS-adv(y)	no conv.	no conv.	0.817	0.765	no conv.	no conv.
diff(x)	0.955	0.943	0.891	0.947	0.729	0.846
diff(y)	0.933	0.899	0.997	1.011	1.147	1.034

7 Numerical tests with the shallow water model

After the simplified model already discussed, we proceed with testing the velocity reconstruction methods in the shallow water solver. This adds the influence of the bed discretization, the waterlevel discretization and the time integration, which was first left out. All this is done in Fortran, as this is faster than Matlab. The code to compute the second order reconstruction method over the dual volume is added to the D-Flow FM program, but the method that integrates over the cell itself is left out, as it is almost the same.

In D-Flow FM, we will try and simulate again a constant velocity field by taking a simple channel flow with a constant bed slope and bottom friction. We will simulate a linear field using a moving plates test and a quadratic field using a channel with a no-slip condition on the channel walls. These can all be done on the same grids. Taking a channel with length $L = 400$ meters in east-west direction, and a width of $W = 100$m (north-south), we let the water flow in at the west boundary and out at the east boundary. The north and south boundary can be used to apply slip conditions or velocity boundaries. We take a water depth of 10m (a bed level at $z = -10m$, to which we apply a slope for the constant and quadratic velocity field) and record for each test the discharge through the east (outflow) boundary, the water surface level and the maximum velocity and its profile.

As all of the testcases have the analytical advection equal to zero, the advection calculation as seen in section 4.2 will be turned off for half of the simulations to test its accuracy.

We can use square grids with cells of size $10 \times 10m$, $5 \times 5m$, $2.5 \times 2.5m$), mixed grids (figure 7a) where the left part of the grid has squares of $10m$ and $5m$, then a transition area using triangles and to the right a rectangular grid with cells of $10 \times 5m$ and $5 \times 2.5m$ respectively, and fully triangular grids (figure 7b).

After careful consideration, the least squares method will not be analyzed here, as its matrices are degenerate on aligned grids, and the simulations become unstable for almost every test in this section.

For consistency, we will use the following colors throughout all the following plots:

blue: Dual velocity reconstruction with advection

red: Dual velocity reconstruction without advection

green: Perot's velocity reconstruction with advection

magenta: Perot's velocity reconstruction without advection

We set $\theta = 1$. The wave computations are stable since $\theta \geq \frac{1}{2}$, (see section 4.6) but the waves will be dissipated for large θ. However, for the stationary simulations, we choose faster convergence and ignore the dissipation.

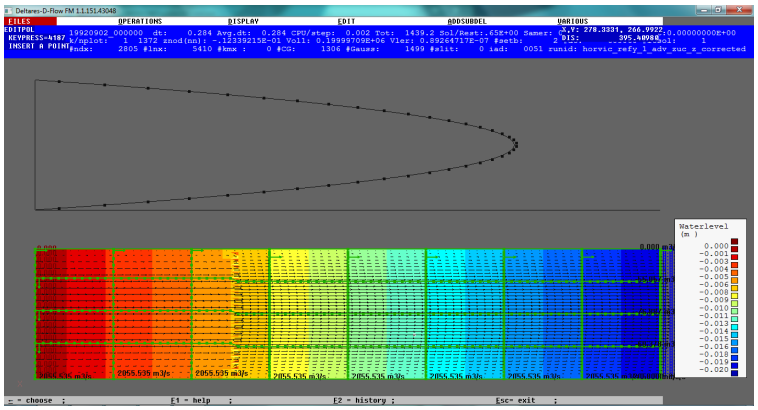
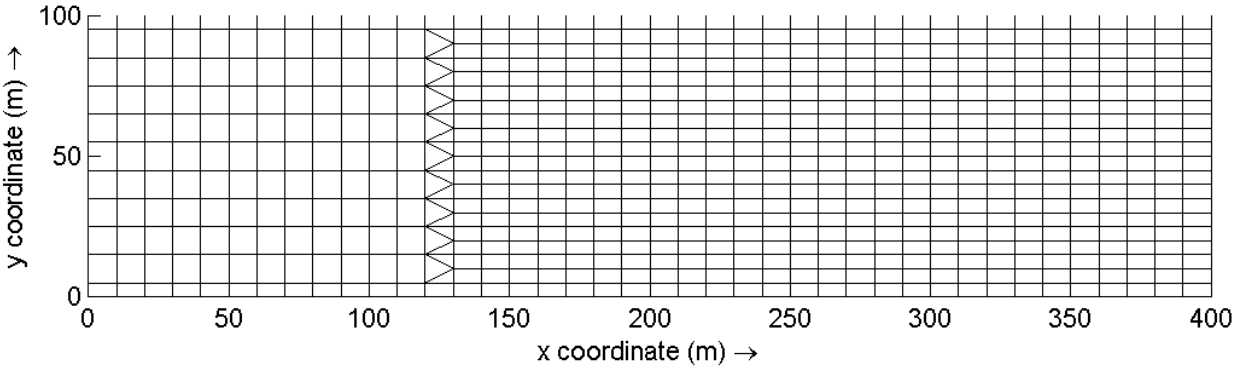
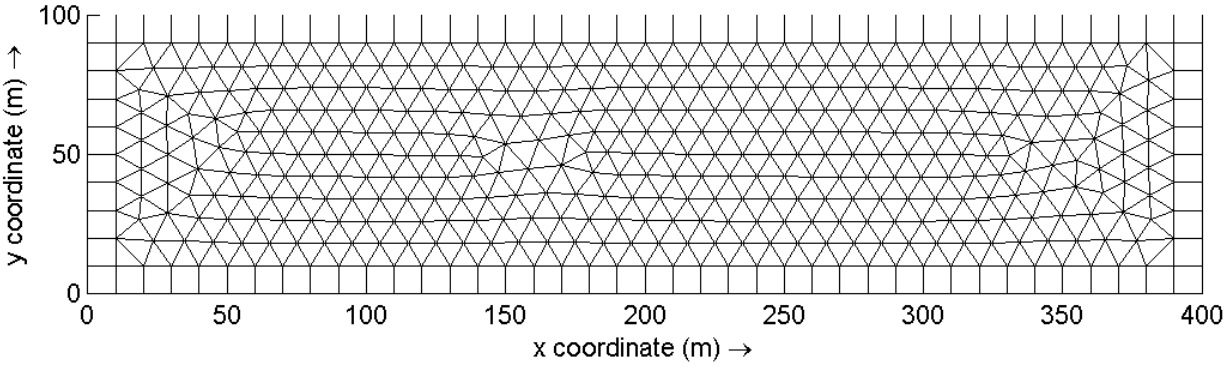


Figure 6: Screenshot of the program



(a) Mixed grid



(b) Triangular grid

Figure 7: Types of grids used for D-Flow FM simulations, the fully square grid is not pictured

7.1 Constant velocity profile: uniform channel flow without wall friction

7.1.1 Test description

A constant velocity profile is obtained by taking a bottom slope of $i = 5 \cdot 10^{-5} m/m$, and a water depth of $10m$ at the in- and outflow boundaries (the bottom is at $z = -10m$ at the inflow boundary and at $z = -10.02m$ at the outflow boundary, we apply boundary conditions $\zeta = 0m$ at the inflow boundary and $\zeta = -0.02m$ at the outflow boundary). The bottom friction is $C = 65\sqrt{m}/s$ (C represents the Chézy coefficient).

7.1.2 Analytical solution

We can use Chézy's formula $|u| = C\sqrt{Ri}$ where R is the hydraulic radius, which is the area of a cross-section orthogonal to the flow divided by the wetted perimeter(the length of the cross section boundaries that touch water). Due to the implementation in D-Flow FM, we can substitute the total water depth H here. Finally, i is the bottom slope to determine the velocity:

$$u = C\sqrt{Ri} = 65\sqrt{5 \cdot 10^{-4}} = 1.453444185374863\, m/s. \tag{110}$$

Since the velocity is constant in x-direction and zero in the y-direction, there is no advection. The discharge area is $A = 1000\, m^2$ and hence the discharge is $Q = Au = 1453.444185374863\, m^3/s$.

7.1.3 Results

It turns out to be necessary to slightly change the discretization of $h\vec{u}\big|_f$. We are working with a discretized bed that exactly follows the prescribed slope (most discretizations work with small steps in the bottom). If cell 1 has outflow through face f and cell 2 has inflow through that same face, we define

$$(h\vec{u})_f = (1 - \alpha_f)(h\vec{u})_{c1} + \alpha_f (h\vec{u})_{c2}. \tag{111}$$

This scheme turns out to be stable for the intended simulation but may become unstable for other situations, as it contains a weighted interpolation for an explicit quantity.

Table 10 shows the velocity through the outflow boundary. Since the dual method with advection explodes on the coarsest triangle grid, the refined triangle grid is considered instead.

We can show that all grids show the same convergence behaviour starting from the initial state (figure 8) and have correct water levels up to an error of a quarter millimeter (table 12).

These results are insufficient to show whether Perot's velocity reconstruction or the dual method is better, but this was expected. Since Perot is a first order method and the dual method is second order, both should be able to reconstruct the velocity up to solver precision, but it turns out that the dual method converges to a slightly different value. Its causes will be discussed in chapter 8.

We can also see the relative errors in the velocity (figure 10) where we see that the simulations with advection are better at the boundary, but distort the uniform profile by an amount of $2 \cdot 10^{-9}$. Table 11 shows that Perot's velocity reconstruction performs better than the dual, and table 12 shows that the square grid has the most accurate waterlevels. The differences between Perot's reconstruction and the dual reconstruction are discussed in section 8.

Table 10: Outflow velocities u in m/s (should be $1.453444185374863 m/s$) for uniform channel flow

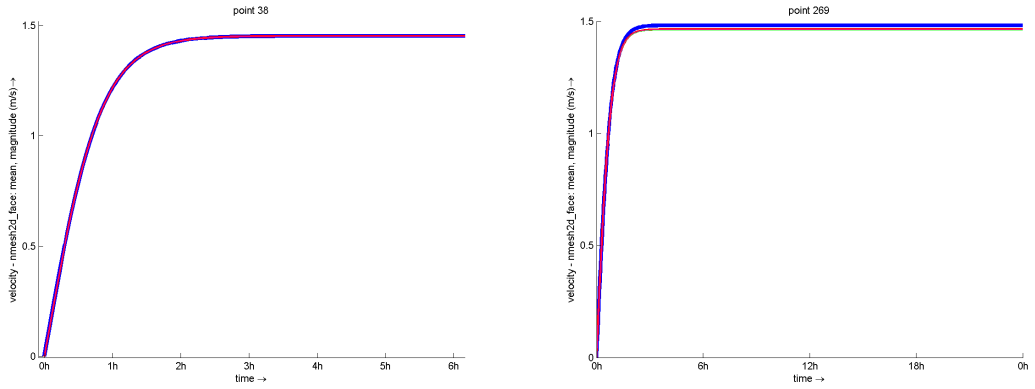
Grid		u with Advection	u without Adv
Square	Perot	1.453444185375	1.453444185375
	Dual	1.454822543164	1.455783309386
Mixed	Perot	1.453444225404	1.453444225317
	Dual	1.454824179518	1.455783319692
Triangles	Perot	1.453445952744	1.453445949843
	Dual	(explodes)	1.455783835685
Tr. Refined	Perot	1.450192082956	1.461790473772
	Dual	1.448057509625	1.461711063228

Table 11: Velocity errors over entire grid in the uniform velocity profile

		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
Square	Perot	$8.5{\cdot}10^{-8}$	$8.5{\cdot}10^{-8}$	$1.1{\cdot}10^{-7}$	$1.1{\cdot}10^{-7}$	$9.7{\cdot}10^{-8}$	$9.7{\cdot}10^{-8}$
	Dual	$1.4{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$1.8{\cdot}10^{-3}$	$3.0{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$2.7{\cdot}10^{-3}$
Mixed	Perot	$4.7{\cdot}10^{-8}$	$4.7{\cdot}10^{-8}$	$6.2{\cdot}10^{-8}$	$6.2{\cdot}10^{-8}$	$5.3{\cdot}10^{-8}$	$5.3{\cdot}10^{-8}$
	Dual	$1.4{\cdot}10^{-3}$	$2.4{\cdot}10^{-3}$	$1.9{\cdot}10^{-3}$	$3.3{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$2.7{\cdot}10^{-3}$
Triangle	Perot	$1.9{\cdot}10^{-6}$	$1.9{\cdot}10^{-6}$	$2.2{\cdot}10^{-6}$	$2.2{\cdot}10^{-6}$	$2.0{\cdot}10^{-6}$	$2.0{\cdot}10^{-6}$
refined	Dual	$1.4{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$3.0{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$2.7{\cdot}10^{-3}$

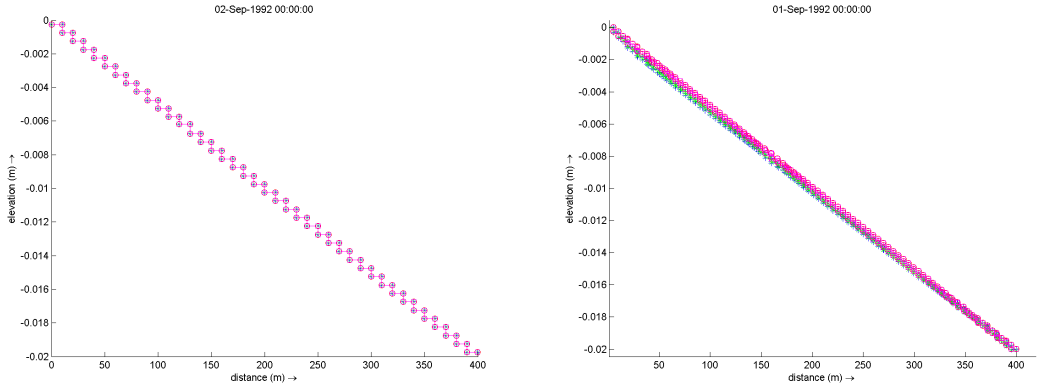
Table 12: Waterlevel errors over entire grid in the uniform velocity profile

		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
Square	Perot	$1.9{\cdot}10^{-15}$	$1.9{\cdot}10^{-15}$	$2.2{\cdot}10^{-16}$	$2.2{\cdot}10^{-16}$	$4.1{\cdot}10^{-16}$	$4.1{\cdot}10^{-16}$
	Dual	$2.5{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$	$3.0{\cdot}10^{-4}$	$2.8{\cdot}10^{-4}$	$2.6{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$
Mixed	Perot	$8.2{\cdot}10^{-10}$	$8.2{\cdot}10^{-10}$	$4.6{\cdot}10^{-10}$	$4.6{\cdot}10^{-10}$	$4.7{\cdot}10^{-10}$	$4.7{\cdot}10^{-10}$
	Dual	$2.5{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$	$3.1{\cdot}10^{-4}$	$3.0{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$	$2.6{\cdot}10^{-4}$
Triangles	Perot	$3.2{\cdot}10^{-9}$	$3.2{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$
refined	Dual	$2.5{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$	$3.0{\cdot}10^{-4}$	$2.8{\cdot}10^{-4}$	$2.6{\cdot}10^{-4}$	$2.5{\cdot}10^{-4}$



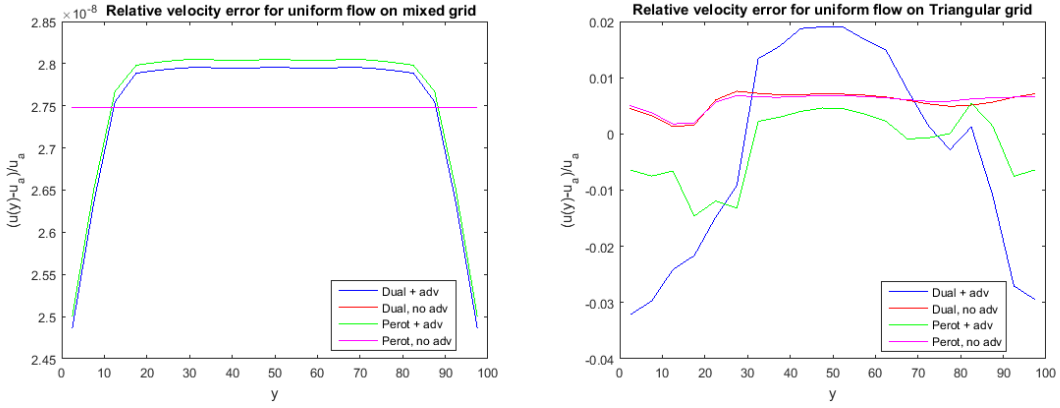
(a) Mixed grid (b) Triangular grid

Figure 8: Convergence behaviour of the velocity for uniform channel flow



(a) Mixed grid (b) Triangular refined grid

Figure 9: The water levels (vertical axis) along the channel centerline for uniform channel flow



(a) Mixed grid (b) Triangular grid

Figure 10: Relative errors in velocity as a function of y for uniform channel flow at the outflow boundary

7.2 Linear velocity profile: Couette moving plates flow

7.2.1 Test description

Consider two infinitely long plates parallel to one another, with water flowing in between. The distance between the plates (width of the channel) is W. If we move the plates in the east-west direction, the water will adhere to the plates and the water velocity will show a linear profile. We can simulate this by letting the water flow from west to east, set the tangential velocity on the northern boundary to 10m/s and the tangential velocity on the southern boundary to zero. The water surface level $\zeta = 0$ at the in- and outflow boundary, and the bottom friction is zero.

7.2.2 Analytical solution

Starting from the Navier-Stokes equations, we know that $\frac{\partial u}{\partial x}$ and v should be zero. We cross out all terms that would be zero in a steady state: $\frac{\partial u}{\partial t}$, $u\frac{\partial u}{\partial x}$, $v\frac{\partial u}{\partial y}$, $u\frac{\partial v}{\partial x}$, $v\frac{\partial v}{\partial y}$, $\frac{\partial^2 u}{\partial x^2}$, $\frac{\partial^2 v}{\partial y^2}$, and we get: $\frac{1}{\rho}\frac{\partial p}{\partial x} = \nu\frac{\partial^2 u}{\partial y^2}$. We know that $\frac{\partial p}{\partial x}$ is constant since u does not depend on x. Let $u(x,0) = 0$, $u(x,W) = U_W$, and we integrate:

$$\begin{aligned}
\frac{\partial u}{\partial y} &= \int \frac{\partial^2 u}{\partial y^2}\, dy \\
&= \int \frac{1}{\rho\nu}\frac{\partial p}{\partial x}\, dy \\
&= \frac{1}{\rho\nu}\frac{\partial p}{\partial x} y + C_1 \qquad (112) \\
u(y) &= \int \frac{\partial u}{\partial y}\, dy \\
&= \int_0^{U_W} \frac{1}{\rho\nu}\frac{\partial p}{\partial x} y + C_1\, dy \\
&= \frac{1}{2\rho\nu}\frac{\partial p}{\partial x} y^2 + C_1 y + C_2 \qquad (113) \\
u(0) = 0 &\Rightarrow C_2 = 0 \qquad (114) \\
u(W) = U_W &\Rightarrow C_1 = \frac{U_W}{W} - \frac{W}{2\rho\nu}\frac{dp}{dx} \qquad (115) \\
u(y) &= \frac{1}{2\rho\nu}\frac{dp}{dx} y(y - W) + \frac{U_W}{W} y \qquad (116)
\end{aligned}$$

If we assume incompressibility, i.e. $\frac{\partial p}{\partial x} = 0$, we get $u = \frac{U_W}{W} y$. We can integrate to get the discharge, with a depth of $10m$ and width $W = 100m$, and $U_W = 10m/s$, this is

$$Q = \int_{z=-10}^{0} \int_{y=0}^{100} \frac{y}{10}\, dy\, dz = 5000\, m^3/s.$$

The surface of the water is exactly zero over the entire domain.

Implementation boundary conditions

The ghost points where the velocity boundaries are defined lie at a distance of $\frac{1}{2}\sqrt{V_c}$ outside of the channel, where V_c is the area of the corresponding interior cell. For the coarsest triangular grid,

Table 13: Discharge Q in m^3/s for flow between two plates and the percentual increase from the correct value of $5000m^3/s$

Grid		Q with Advection	Q without Adv
Square	Perot	4999.999996443687	5000.000000002055
	increase (%)	-7.1126·10^{-8}	4.1091·10^{-11}
	Dual	4999.999967612755	4999.999999997543
	increase (%)	-6.4774·10^{-7}	-4.9149·10^{-11}
Mixed	Perot	4999.940088164551	5000.110331962351
	increase (%)	-0.0012	0.0022
	Dual	4998.669147418284	5000.107002698290
	increase (%)	-0.0266	0.0021
Triangles	Perot	4937.418569010896	4999.431100102393
	increase (%)	-1.2516	-0.0114
	Dual	5050.031122902201	5000.345021685037
	increase (%)	1.0006	0.0069

Table 14: Velocity errors over the entire grid for flow between two plates

		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
Square	Perot	1.8·10^{-8}	1.8·10^{-8}	3.5·10^{-9}	3.5·10^{-9}	4.4·10^{-9}	4.4·10^{-9}
	Dual	1.8·10^{-8}	1.8·10^{-8}	3.5·10^{-9}	1.4·10^{-9}	4.4·10^{-9}	3.5·10^{-9}
Mixed	Perot	0.12	0.12	0.004	0.004	0.011	0.011
	Dual	0.080	0.077	0.0031	0.0034	0.0074	0.0086
Triangle	Perot	1.0012	1.0012	0.381	0.381	0.430	0.430
	Dual	0.274	0.0361	0.0976	0.0052	0.115	0.0071

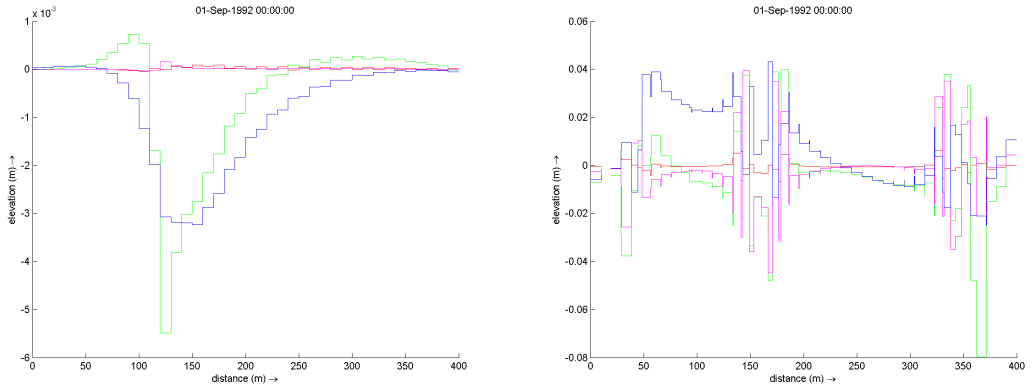
$A = 100m^2$, hence the ghost points lie at $y = -5m$ and $y = 105m$ and we can define the velocity in the ghost points as $u = (0, -0.5)$ and $(0, 10.5)$. The mixed grid has boundary cells of size $50m^2$, hence the ghost points are at a distance of $\sqrt{50}/2$ from the boundary, making the velocity in the ghost points equal to $u = (0, -\sqrt{50}/20)$ and $(0, 10 + \sqrt{50}/20)$.

7.2.3 Results

The discharges are listed in table 13, with a highest error of 1.25%. The velocity errors L^∞, L^1, L^2 are in table 14, the waterlevel errors (remember $\zeta = 0$ everywhere) are in table 15 and a cross section along the middle of the channel is shown in figure 11. Figure 12 shows relative velocity errors at the outflow boundary as a function of y. The maximum waterlevel error is $22cm$, while the maximum velocity error is $1m/s$. Figure 11 shows that the maximal error in ζ for the mixed grid coincides with the row of triangles. The smallest errors are obtained for the square grid, where the dual method performs the same as Perot's method, but for the mixed and triangular grid we can see that the errors in the dual method are slightly smaller than the errors for Perot's method. This is expected as Perot's method is not exact for a linear profile on an irregular grid.

Table 15: Waterlevel errors in m (correct: $\zeta = 0$) over the entire grid for flow between two plates

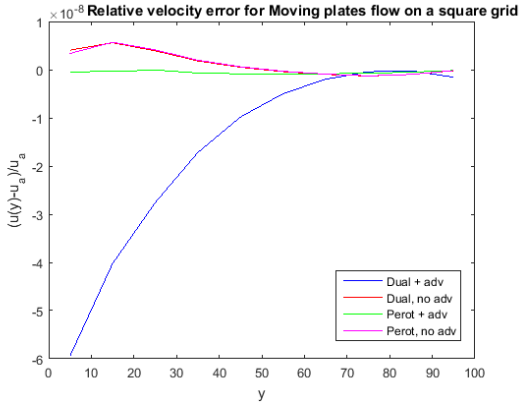
		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
Square	Perot	$8.7{\cdot}10^{-9}$	$8.7{\cdot}10^{-9}$	$1.3{\cdot}10^{-9}$	$1.3{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$	$1.6{\cdot}10^{-9}$
	Dual	$8.7{\cdot}10^{-9}$	$8.9{\cdot}10^{-9}$	$1.3{\cdot}10^{-9}$	$3.3{\cdot}10^{-10}$	$1.6{\cdot}10^{-9}$	$1.2{\cdot}10^{-9}$
Mixed	Perot	0.1042	0.1042	0.0017	0.0017	0.0092	0.0092
	Dual	0.0399	0.0379	0.0013	0.00062	0.00399	0.00353
Triangles	Perot	0.221	0.221	0.0172	0.0172	0.0296	0.0296
	Dual	0.092	0.039	0.014	0.0014	0.019	0.0036



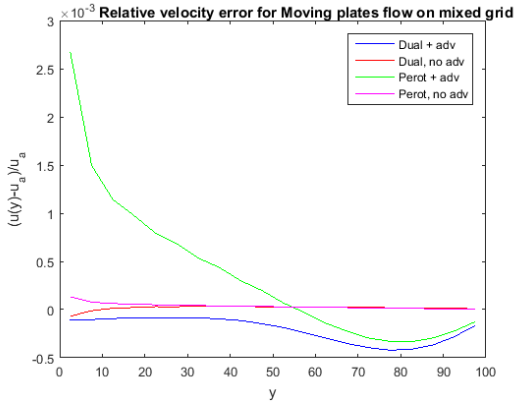
(a) Mixed grid

(b) Triangular grid

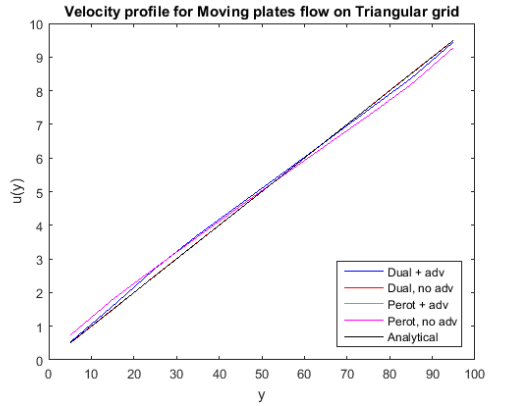
Figure 11: The water levels (vertical axis) along the line from $(x, y) = (0, 50)$ to $(400, 50)$ for flow between two plates



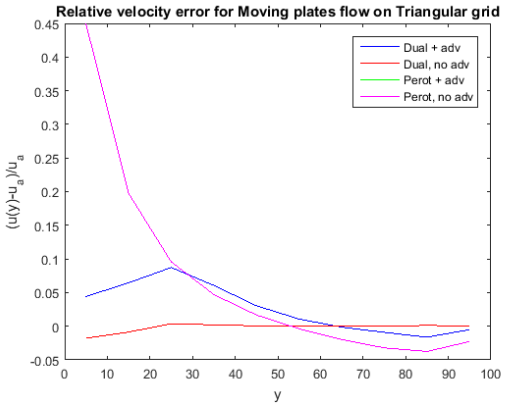
(a) Square grid



(b) Mixed grid



(c) Velocity profile triangular grid



(d) Relative errors triangular grid

Figure 12: Relative velocity errors for flow between two plates as a function of y at the outflow boundary

7.3 Quadratic velocity profile: Poiseuille Flow

7.3.1 Test description

The Poiseuille Open Channel Flow [2] has a no-slip condition on the north and south boundaries, which causes the water to be faster in the middle of the channel. There is no bottom friction. The correct solution has a quadratic velocity profile, so we do not expect either Perot's or the dual-integrated velocity reconstruction method to reach machine precision. Since the boundary conditions are implemented in a linear fashion, this might be impossible. For this reason, a convergence study is performed just as in section 6.4.

7.3.2 Analytical solution

For a rectangular channel with a constant bed slope and corresponding waterlevels (the water column has constant height H), assume that the density ρ and kinematic viscosity ν are constant, there is no bottom friction or influence from the Coriolis force, or from wind or waves. This means that $\zeta(x=0)=\zeta_{in}$, $\zeta(x=L)=\zeta_{in}-\Delta\zeta$, where the parameters ζ_{in} and $\Delta\zeta$ are given. The boundaries in y-direction are no-slip boundaries: $u=v=0$ at $y=0$ and $y=W$. The Shallow Water Equations reduce since the terms $\frac{\partial\zeta}{\partial t}$, $\frac{\partial u}{\partial y}$, $\frac{\partial u}{\partial t}$, $\frac{\partial v}{\partial t}$, $g\frac{\partial\zeta}{\partial y}$, $u\frac{\partial u}{\partial x}$, $v\frac{\partial u}{\partial y}$, $\frac{\partial^2 u}{\partial x^2}$, $\frac{\partial^2 v}{\partial y\partial x}$, $u\frac{\partial v}{\partial x}$, $v\frac{\partial v}{\partial y}$, $\frac{\partial^2 v}{\partial x^2}$, $\frac{\partial^2 u}{\partial x\partial y}$, $\frac{\partial^2 v}{\partial y^2}$ are all zero once the flow has reached a steady state and we obtain the system

$$H\big(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\big)=0, \qquad g\frac{\partial\zeta}{\partial x}=\nu\frac{\partial^2 u}{\partial y^2} \tag{117}$$

We had assumed the water column height H to be constant, and since we know the bed slope and the boundaries for ζ, we immediately know that $\zeta(x)=\zeta_{in}-\Delta\zeta\frac{x}{L}$. Then, knowing that $\frac{\partial u}{\partial y}$ needs to have a zero at W/2 (the middle of the channel is where the flow is fastest, hence u has a local maximum), we have

$$\frac{\partial^2 u}{\partial y^2}=\frac{g}{\nu}\frac{\partial\zeta}{\partial x}=-\frac{g\Delta\zeta}{\nu L} \quad\Rightarrow\quad \frac{\partial u}{\partial y}=\frac{-g\Delta\zeta}{\nu L}(y-W/2) \tag{118}$$

Then, using $u=v=0$ at the no-slip boundaries, we find

$$u(y)=\frac{-g\Delta\zeta}{\nu L}\int\big(y-\frac{W}{2}\big)\,dy=\frac{-g\Delta\zeta}{2\nu L}(y^2-Wy) \tag{119}$$

We find $u_{\max}=u(x,W/2)=\frac{g\Delta\zeta}{2\nu L}\cdot 2500$, the discharge Q (independent of x) through a cross-section of the channel orthogonal to the flow is

$$Q=\int_{z=-H}^{0}\int_{y=0}^{W}u(y)\,dy\,dz=\frac{H}{6}\cdot 10^6\cdot\frac{g\Delta\zeta}{\nu L}\,m^3/s. \tag{120}$$

Using $H=10$, $\Delta\zeta=2cm$, $\nu=0.1$, we get $Q=4087.5$ and $u_{\max}=6.131$.

7.3.3 Convergence on a square grid

Since this profile is quadratic, it would be unreasonable to expect either Perot or the dual velocity reconstruction to reach machine precision, even on a square grid. This is why we first make a

Table 16: Errors in the Poiseuille flow waterlevels over entire square grid

		L^∞		L^1		L^2	
Grid size		Adv	no Adv	Adv	no Adv	Adv	no Adv
10m	Perot	$1.8{\cdot}10^{-3}$	$1.8{\cdot}10^{-3}$	$1.0{\cdot}10^{-3}$	$1.0{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$
	Dual	$1.8{\cdot}10^{-3}$	$1.4{\cdot}10^{-4}$	$1.0{\cdot}10^{-3}$	$8.3{\cdot}10^{-5}$	$1.1{\cdot}10^{-3}$	$8.4{\cdot}10^{-5}$
5m	Perot	$4.2{\cdot}10^{-4}$	$4.2{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$	$2.7{\cdot}10^{-4}$
	Dual	$4.2{\cdot}10^{-4}$	$5.6{\cdot}10^{-5}$	$2.7{\cdot}10^{-4}$	$3.1{\cdot}10^{-5}$	$2.7{\cdot}10^{-4}$	$3.1{\cdot}10^{-5}$
2.5m	Perot	$1.2{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$	$7.9{\cdot}10^{-5}$	$7.9{\cdot}10^{-5}$	$7.8{\cdot}10^{-5}$	$7.8{\cdot}10^{-5}$
	Dual	$1.2{\cdot}10^{-4}$	$2.2{\cdot}10^{-5}$	$7.9{\cdot}10^{-5}$	$1.0{\cdot}10^{-5}$	$7.8{\cdot}10^{-5}$	$9.8{\cdot}10^{-6}$
Convergence rates							
		L^∞		L^1		L^2	
Perot + advection		2.08	1.84	1.87	1.80	1.95	1.81
Perot without adv		2.08	1.84	1.87	1.80	1.95	1.81
Dual + advection		2.08	1.84	1.87	1.80	1.95	1.81
Dual without adv		1.31	1.32	1.41	1.62	1.45	1.64

convergence study on a series of square grids, with cell sizes of $10 \times 10m$, $5 \times 5m$ and $2.5 \times 2.5m$. The channel is as before: 400m long and 100m wide, so we work with 400 cells, 1600 cells and 6400 cells. We will consider convergence rates as in section 6.4, for each method (Perot with and without advection, dual velocity reconstruction with and without advection). Figure 13 shows that the velocities at the boundaries differ most from the prescribed profile, but this may be caused by the linearization in the boundary conditions (see paragraph 5.6).

Again, the convergence rates are solutions to the equation $L^p_{\text{coarse}} = 2^r L^p_{\text{fine}}$, with $p \in \{1, 2, \infty\}$. When $r = 1$, this means linear convergence, when $r = 2$ the convergence is quadratic, etc. The L^1 norm mostly shows the convergence of primary variables such as velocity and water levels, while L^2 can consider the convergence of quantities such as the kinetic energy $E_k = \frac{1}{2}m|u|^2$, and their limit L^∞ gives the maximum deviation from the correct value.

Convergence rates are in table 18, based on the errors in the same table. The errors are calculated over the entire grid. Since all cells are square, Perot's velocity reconstruction method has (almost) quadratic behaviour and the matrix A formed by the second order correction is just an identity matrix. According to table 19, the methods show the same order in all the errors (meaning that the error caused by a non-zero calculated advection is small compared to the overall error). Only the dual method without advection differs from the other three situations; we see that this converges faster than it should, since $r > 2$, the convergence seems better than quadratic. As the velocity data for the runs using the dual method without advection contain a period in the last digits, which does not disappear when the simulation is run for a longer time or when the maximal Courant number is lowered, full time-convergence is not obtained and the convergence rate r may be unreliable. This explains the cases where $r > 2$. As advection has a dissipating effect, this period in the data disappears for the runs with advection enabled. However, for Perot's method and the dual with advection, we see superlinear convergence, as expected.

Table 17: Discharge Q in m^3/s for Poiseuille flow over square grid, correct value is $4087.5 m^3/s$ and the percentual difference (increase compared to correct value):

	Perot			Dual	
Square	with advection	without advection		with advection	without advection
10 m	4119.308461207200	4095.487871438126	10 m	4119.308461207199	4142.104339836873
increase (%)	0.7782	0.1954		0.7782	1.3359
5 m	4095.487871438126	4098.729045990548	5 m	4095.487871438127	4098.729045992969
increase (%)	0.1954	0.2747		0.1954	0.2747
2.5m	4089.529567153803	4089.773057728640	2.5m	4089.529567153802	4089.773057728640
increase (%)	0.0497	0.0556		0.0497	0.0556

Table 18: Errors in the Poiseuille flow velocity over entire square grid

Grid size		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
10m	Perot	0.036	0.036	0.022	0.022	0.024	0.024
	Dual	0.036	0.053	0.022	0.050	0.024	0.046
5m	Perot	0.012	0.012	0.0075	0.0075	0.0084	0.0084
	Dual	0.012	0.014	0.0075	0.012	0.0084	0.012
2.5m	Perot	0.0034	0.0034	0.0022	0.0022	0.0025	0.0025
	Dual	0.0034	0.0036	0.0022	0.0028	0.0025	0.0029
Convergence rates							
		L^∞		L^1		L^2	
Perot + advection		1.60	1.78	1.53	1.74	1.53	1.76
Perot without adv		1.60	1.78	1.53	1.74	1.53	1.76
Dual + advection		1.60	1.78	1.53	1.74	1.53	1.76
Dual without adv		1.93	1.95	2.11	2.07	2.02	1.98

Table 19: Errors in the Poiseuille flow velocity over entire (non-uniform) grid

		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
Mixed	Perot 10m	0.50	0.50	0.15	0.15	0.16	0.16
	Perot 5m	0.22	0.22	0.027	0.027	0.039	0.039
	Dual 10 m	0.42	0.036	0.17	0.026	0.19	0.024
	Dual 5 m	0.18	0.012	0.031	0.0066	0.041	0.0060
Triangles	Perot 10m	1.23	1.23	0.48	0.48	0.49	0.49
	Perot 5m	0.79	0.79	0.22	0.22	0.22	0.22
	Dual 10m	0.69	0.058	0.34	0.034	0.34	0.040
	Dual 5m	0.51	0.10	0.12	0.012	0.12	0.014
		Convergence rates					
		L^∞		L^1		L^2	
Mixed	Perot + advection		1.16		2.47		2.05
	Perot without adv		1.16		2.47		2.05
	Dual + advection		1.19		2.47		2.25
	Dual without adv		1.59		1.98		2.01
Triangles	Perot + advection		0.65		1.15		1.13
	Perot without adv		0.65		1.15		1.13
	Dual + advection		0.41		1.46		1.44
	Dual without adv		no conv.		1.47		1.55

7.3.4 Convergence results on non-uniform grids

Convergence rates for the dual velocity reconstruction method according to table 19 are only slightly better than those for Perot's method; the same is seen in the waterlevels (table 21). This is expected since a second order method cannot reconstruct a quadratic field on an unstructured grid to machine precision. The mixed grid shows better convergence than the triangular grid. The discharges (table 20) have the best result on the mixed grid, where it does not matter if Perot's method or the dual method is employed. For the triangular grid, the discharges are more accurate with the dual velocity method.

Table 20: Discharge Q in m^3/s for Poiseuille flow over non-uniform grid, and the percentual increase compared to the correct value of $4087.5 m^3/s$:

		Perot		Dual	
		with advection	without advection	with advection	without advection
Mixed	10m	4121.472492720819	4112.379823648109	4131.907814431218	4117.174245421137
increase (%)		0.8311	0.6087	0.864	0.7260
	5m	4089.141989251060	4092.713149988351	4090.540939775473	4094.561307010622
increase (%)		0.0402	0.1275	0.0744	0.1728
Triangles	10m	4296.404604173152	3716.198689253789	4291.865011730972	4129.165508965953
increase (%)		5.1108	-9.0838	4.9998	1.0193
	5m	4167.015275640806	3809.114770554570	4133.956318626508	4093.571361283370
increase (%)		1.9453	-6.8106	1.1365	0.1485

Table 21: Waterlevel errors in the Poiseuille flow velocity over entire (non-uniform) grid

		L^∞		L^1		L^2	
		Adv	no Adv	Adv	no Adv	Adv	no Adv
Mixed	Perot 10m	0.0083	0.0083	0.0058	0.0058	0.0057	0.0057
	Perot 5m	0.0060	0.0060	0.0021	0.0021	0.0024	0.0024
	Dual 10 m	0.021	0.00024	0.013	$2.5{\cdot}10^{-5}$	0.013	$3.4{\cdot}10^{-5}$
	Dual 5 m	0.010	0.00033	0.0034	$7.9{\cdot}10^{-6}$	0.0037	$1.6{\cdot}10^{-5}$
Triangles	Perot 10m	0.082	0.082	0.044	0.044	0.041	0.041
	Perot 5 m	0.050	0.050	0.025	0.025	0.022	0.022
	Dual 10m	0.14	0.00017	0.026	$4.9{\cdot}10^{-5}$	0.025	$4.9{\cdot}10^{-5}$
	Dual 5 m	0.066	0.0012	0.011	$2.2{\cdot}10^{-5}$	0.010	$6.4{\cdot}10^{-5}$
Convergence rates							
		L^∞		L^1		L^2	
Mixed	Perot + advection	0.466		1.47		1.26	
	Perot without adv	0.466		1.47		1.26	
	Dual + advection	1.03		1.97		1.87	
	Dual without adv	no conv.		1.68		1.14	
Triangles	Perot + advection	0.71		0.82		0.90	
	Perot without adv	0.71		0.82		0.90	
	Dual + advection	1.07		1.30		1.31	
	Dual without adv	no conv.		1.137084		no conv.	

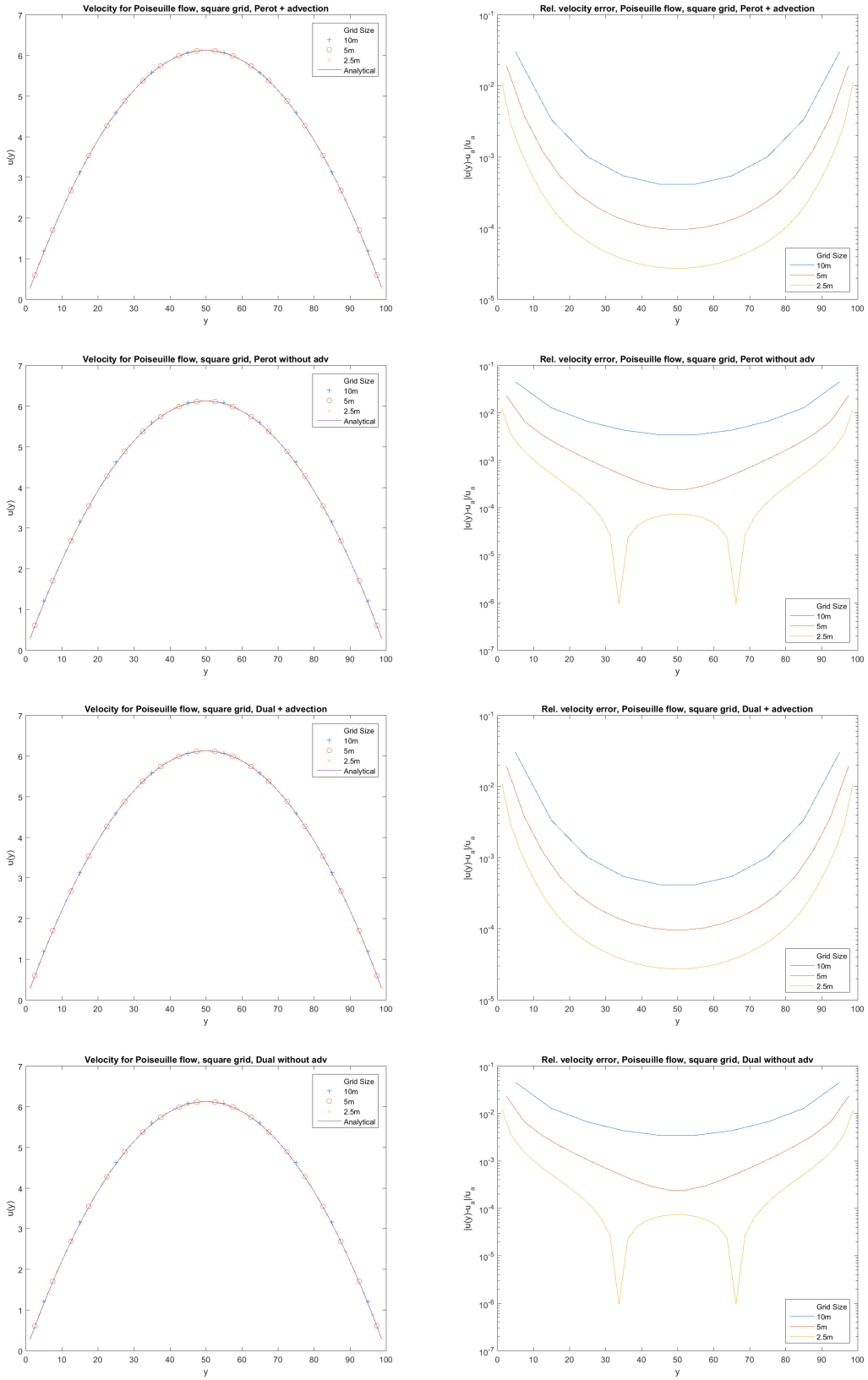


Figure 13: Velocity profiles for Poiseuille flow on square grid

8 Discussion

In the chapters 6 and 7, we see the second order velocity reconstruction methods perform better than Perot's first order velocity reconstruction method. Still, a few remarks need to be made.

The experiments with velocity reconstruction in the simplified model (chapter 6) show that the second order methods perform better than Perot's method when the grid consists of only triangles, but not when quadrilaterals are involved. This is probably due to the fact that the circumcenter in a quadrilateral does not always exist, and the approximation formula does not converge to a unique location. On the mixed grid, only the least squares method gives significant improvement over Perot.

We also saw that the advection calculation suffers from the fact that u_f is taken as a point value, not as a quantity varying over the face. This is especially true for second order advection with Perot's velocity reconstruction.

For advection and diffusion, both the dual and the least squares method perform better than Perot's method, though they need to be paired with a second order advection method (preferably second order upwind) to guarantee convergence (linear convergence for linear fields, sublinear convergence on quadratic fields).

Lastly, we need to remember that the quantities other than velocity, advection and diffusion (water depth, viscosity, ...) are assumed to be unity so the results in D-Flow FM will never be as good as this.

The experiments in D-Flow FM (chapter 7) involve a full matrix which contains mostly zeros being used in the initialization step. As this matrix (A from the dual method) is only based on geometry, it only needs to be calculated once. Changing this to sparse indexing would involve rewriting a lot of the algorithm but, as some attempted simulations needed more memory than available, it would be necessary for an implementation that would be used by external parties.

The experiments with uniform channel flow show that the dual velocity reconstruction method converges to a different value than Perot's method. The uniform flow has this problem both when starting from the analytical solution and when starting with a velocity of $0m/s$. The uniform flow is dominated by the bottom friction term, while the moving plates and Poiseuille are dominated by viscosity. Therefore it needs investigating if the bottom friction somehow reduces the improvements made by the dual method.

The moving plates flow does not show a big difference between Perot's and dual velocity reconstruction. It would be expected that the dual method is capable of reconstructing the velocities up to machine precision, while Perot's method cannot, but the dual method only shows little improvements. The probable cause is the non-linear aspects in the simulations which we have not considered.

For the Poiseuille flow, we see mostly results confirming the earlier data but the non-converging simulations are troubling. As stated before, the dissipating effect of the advection calculation (which is absent here) may be the cause. Another possible explanation for the difference is that the diffusion calculation $\nabla^2 \vec{u}$ (which is solved explicitly) would work better with the second order reconstruction method than with Perot, though we have insufficient data to conclude this.

When simulations diverge, this may be due to the explicit nature of the advection calculation. Setting a lower Courant number or letting the simulations run longer does not help with this problem, but a fully implicit program is not (yet) available, and working on this is outside the scope of this thesis. Another possible problem is that the advection computation uses first-order ingredients (also for the second order velocity reconstruction) and point values u_f.

A last consideration is the computational time of the model; as Perot's velocity reconstruction often produces good results, it is not always necessary to use the second order velocity reconstruction. The matrix A for the dual velocity reconstruction is constructed only once (since it depends only on the grid geometry), but used to solve an equation in every timestep. It also does not replace Perot's velocity reconstruction, but uses and then overwrites it. The matrix to solve for ζ is constructed and solved every timestep, but its size is $n \times n$ (where n is the number of cells in the grid), while the matrix A has size $2n \times 2n$ and has four times as many non-zero entries. The matrix solver in D-Flow FM uses Gaussian Elimination where possible and Conjugate Gradients in other cases [13]. Both of these methods have time complexity $\mathcal{O}(n^3)$ [9] for a $n \times n$ matrix, so solving the water level matrix has costs $k \cdot n^3$ for some constant k while A has costs $k(2n)^3 = 8kn^3$, which means it takes eight times as long to solve as the waterlevel matrix. On the other hand, the co-opting of the waterlevel solver for the second order velocities may be a problem in itself. As the solver is optimized to solve (common instances of the) equation system for ζ, better results may be possible with another solver, especially one that does not have the requirement of a symmetric system.

9 Conclusion

The dual velocity reconstruction method with second order upwind advection is the best choice (when it comes to accuracy) in the simplified model on a triangular grid. For a mixed grid, the least squares velocity reconstruction method is better in some cases, though the convergence of any of the advection methods cannot be guaranteed for the least squares velocities. The second order reconstruction integrated over the cell itself is an acceptable alternative for integration over the dual volume. The hybrid method (section 5.4) does not improve on Perot.

While Perot's method shows first order grid convergence in the velocities, the second order methods (correction using integration over the dual volume or the cell itself, and least squares method) have second order grid convergence in the velocities and first order grid convergence in the gradients, and for some choices of the advection and diffusion discretization methods.

In D-Flow FM, for both the linear and quadratic velocity profile, the dual method is equal to Perot's velocity reconstruction method on square grids, as expected. For non-uniform grids, it is only a little better in each of the measured quantities: discharge, velocity and water level. We assume the dual method without advection to be unreliable for the reasons stated in chapter 8.

We can conclude that the second order velocity methods are indeed more accurate than Perot's method. The dual method has satisfactory convergence behaviour in both the simplified Matlab model and in D-Flow FM, though a second order advection method would need to be added. Furthermore, it is a general method, appropriate for all grids and for both explicit and implicit discretizations. The second order method that integrates over the cell and the least squares method show satisfying behaviour in the simplified model.

As the dual method does not show large improvements over Perot's method, and deteriorates when the grid becomes more irregular, might not be worth the additional running time. Whether this can be solved by using another (differently optimized) matrix solver or by optimizing the existing code was not part of this research. A possibly easier solution is, for a domain requiring a very irregular grid where the dual method does show a clear advantage over Perot, a faster solution may be to perform a domain decomposition which enables a more regular grid to be used. This may add time in the initialization step of the model but give faster running times. In this case, the least squares velocity reconstruction method combined with a fully implicit discretization may be considered as well.

10 Recommendations for further research

Every scientific investigation yields new questions. As I have often encountered unstable simulations in this thesis, which could often be solved by choosing different discretizations for either the bed or the $(hu)_f$, it is natural to wonder if the explicit part of the integration is responsible for this. Programming everything implicitly was not possible in the time available, nor inside the scope of this thesis, but it may improve the stability of the D-Flow FM program.

The dual method with second order advection was not studied in D-Flow FM, though the Matlab experiments suggest it would be the best combination of methods. The diffusion as done in the Matlab experiments (chapter 6) is not used in the D-Flow FM experiments, but it shows good convergence behaviour. The time-integration of the advection and diffusion methods are not studied here. The time-integration of least squares and the cell-integrated second order velocity method are not studied. Even though we have reason to assume that the cell-integrated method would behave the same as the dual velocity reconstruction method, this needs to be verified. The time-integration for the least squares method also needs to be studied, as it might become better when done implicitly.

As the mixed grids show their biggest errors in the triangles area, and the all-triangles grid shows the highest errors of all in the D-Flow FM experiments, even though it has uniquely defined circumcenters for all of its cells, the question remains whether triangles cause large errors (especially in the waterlevels ζ) or the transition between two types of cells (rectangles to triangles and vice versa) is responsible for this type of error.

The uniform flow simulations leave us wondering if the bottom friction has a negative influence on the second order velocity reconstruction.

As pointed out, the large running times of the dual velocity reconstruction method are a big disadvantage, and alternatives should be considered. A fully implicit model of the SWE with a least squares velocity reconstruction, or generating a new, more regular grid (possibly using a domain decomposition) which employs the current discretization may be appropriate solutions.

Acknowledgements

I would like to thank Frank Platzek and Mart Borsboom at Deltares for supervising my research and helping me when I got stuck. I would like to thank my colleagues at Deltares, especially Mohamed Nabi, Arthur van Dam, Herman Kernkamp, Sander van der Pijl, Michal Kleczek, Bert Jagers for discussing mathematics, helping with the implementation of the algorithm, answering questions about the existing D-Flow FM code, or helping with Visual Studio or Delft3D-QuickPlot issues.

Furthermore, I would like to thank Jason Frank, for being my supervisor at Utrecht University.

References

[1] M. Borsboom. Construction and analysis of D-Flow FM-type discretizations. Deltares memo, August 2013.

[2] M. Borsboom and S. van der Pijl. Assessment of accuracy of discretization of convection and viscosity in D-Flow FM. Deltares memo, 2012.

[3] W. Boscheri, M. Dumbser, and M. Righetti. A semi-implicit scheme for 3D free surface flows with high-order velocity reconstruction on unstructured voronoi meshes. *International journal for numerical methods in fluids*, 72:607–631, 2013.

[4] V. Casulli and E. Cattani. Stability, accuracy and efficiency of a semi-implicit method for threedimensional shallow water flow. *Computers and Mathematics with Applications*, 27(4):99–112, 1994.

[5] V. Casulli and R.T. Cheng. Evaluation of the UnTRIM model for 3-D tidal circulation. *Estuarine and Coastal Modeling: Proceedings of the Seventh International Conference in St. Petersburg, FL, November 2001*, pages 628–642, 2001.

[6] V. Casulli and R.A. Walters. An unstructured grid, threedimensional model based on the shallow water equations. *International journal for numerical methods in fluids*, 32:331–348, 2000.

[7] V. Casulli and P. Zanolli. Semi-implicit numerical modeling of nonhydrostatic free-surface flows for environmental problems. *Mathematical and Computer Modelling*, 36:1131–1149, 2002.

[8] O.B. Fringer, M. Gerritsen, and R.L. Street. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling*, 14:139–173, 2006.

[9] G.H. Golub and C.F. van Loan. *Matrix Computations*. John Hopkins University Press, 4th edition, 2013.

[10] D.A. Ham, J. Pietrzak, and G.S. Stelling. A scalable unstructured grid 3-dimensional finite volume model for the shallow water equations. *Ocean Modelling*, 10:153–169, 2005.

[11] P.C. Hammer and A.H. Stroud. Numerical integration over simplexes. *Mathematical tables and other aids to computation*, 10:137–139, 1956.

[12] F.H. Harlow and J.E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8(12):2182–2189, December 1965.

[13] H.W.J. Kernkamp, A. van Dam, G.S.Stelling, and E.D. de Goede. Efficient scheme for the shallow water equations on unstructured grids with application to the continental shelf. *Ocean Dynamics*, 61:1175–1188, 2011.

[14] S.C. Kramer and G.S. Stelling. A conservative unstructured scheme for rapidly varied flows. *International journal for numerical methods in fluids*, 58:183–212, 2008.

[15] P.S. Peixoto and S.R.M. Barros. On vector field reconstructions for semi-lagrangian transport methods on geodesic staggered grids. *Journal of Computational Physics*, 273:185–211, 2014.

[16] J.B. Perot. Conservation properties of unstructured staggered mesh schemes. *Journal of Computational Physics*, 159:58–89, 2000.

[17] F.W. Platzek. *Accuracy and efficiency aspects in numerical river modelling (work in progress)*. PhD thesis, T.U. Delft, 2016.

[18] Stichting Deltares. *D-Flow FM Technical Reference*, 2015.

[19] Maarten van Reeuwijk. A mimetic mass, momentum and energy conserving discretization for the shallow water equations. *Computers and Fluids*, pages 411–416, 2011.

[20] D. Vidovic. Polynomial reconstruction of staggered unstructured vector fields. *Theoret. Appl. Mech.*, 36(2):85–99, 2009.

[21] I. Wenneker, A. Segal, and P. Wesseling. A mach-uniform unstructured staggered grid method. *International Journal for Numerical Methods in Fluids*, 40:1209–1235, 2002.

[22] Pieter Wesseling. *Principles of Computational Fluid Dynamics*. Springer, 2000.

[23] X. Zhang, D. Schmidt, and J. B. Perot. Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics. *Journal of Computational Physics*, 175:764–791, 2002. `http://www.ecs.umass.edu/mie/tcfd/Papers/USM_3D.pdf`.