MASTER'S THESIS

Implementing a low-dissipative symmetry preserving discretization scheme in OpenFOAM

Author:
JANNES HOPMAN, s2018152

Supervisors University of Groningen:	*Supervisors UPC Terrassa:*	*Supervisors NRG Petten:*
PROF. DR. IR. R.W.C.P. VERSTAPPEN	ASSOC. PROF. DR. F.X. TRIAS	IR. E.M.J. KOMEN
PROF. DR. IR. P.R. ONCK	A. PONT VILCHEZ	DR. IR. E.M.A. FREDERIX

Abstract

In this work, the fully-conservative symmetry preserving discretization method of Verstappen and Veldman (2003) [1], which was later generalized to unstructured collocated grids by Trias et al. (2014) [2], is implemented in open source CFD software OpenFOAM. By upholding underlying symmetries in operators of the Navier-Stokes equations, a discretization scheme is derived that preserves energy at all length scales to reduce numerical dissipation, and more accurately depicts turbulence. Using a Taylor-Green vortex case, significant improvements in terms of numerical dissipation were shown in comparison with OpenFOAM's standard icoFoam solver. This effect was mainly caused by the Van Kan pressure prediction method [3], in comparison to the Chorin method [4], which does not include a pressure prediction. The spatial discretization of the symmetry preserving method, which uses midpoint interpolation and projected distances in its gradients, simultaneously improved stability, even on distorted grids, while slightly under-estimating physical diffusion. By performing a temporal consistency study on the lid-driven cavity flow, it was shown that the pressure prediction method was able to increase the order of accuracy of the pressure error from $\mathcal{O}(\Delta t \Delta h^2)$ to $\mathcal{O}(\Delta t^2 \Delta h^2)$. Nevertheless, in a more realistic turbulent channel fow case, the low-dissipative character of the symmetry preserving method started to show some instabilities in the form of checkerboarding. Methods to damp these modes while preserving underlying symmetries could lead to stabilizing these results while conserving energy at all length scales, and potentially open up the way to using even higher order pressure predictions.

April 26, 2020

Table of contents

1 Introduction

In the early nineteenth century, the work of Claude-Louis Navier and George Stokes in fluid mechanics lead to the now famous Navier-Stokes equations, as seen in equations 1 and 2, which govern the flow of incompressible fluids.

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = \nu \nabla^2 \vec{u} - \nabla p \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

Analytical solutions to these equations were only obtained for some two-dimensionsonal and some simple three-dimensional cases, as many three-dimensional vortices and turbulence are too complex to predict. The equations are, however, very useful in approximating these complex cases. A common approach in computational fluid dynamics (CFD), is to approximate these cases by subdividing the domain of a case into many small control volumes, or cells, to which physical variables are attributed, called the finite volume method (FVM). By applying conservation of mass, energy and momentum to each cell, a large system of algebraic equations is created. Up until the late 1950's this approach remained mostly theoretic, because of the sheer number of equations required for even very basic cases. With the rise of computing technology, the interest in, and practical use of CFD has surged. Nowadays, many software packages are available that perform these calculations, accessible to everyday engineers of almost any skill level, without much required knowledge of the underlying principles.

One of these software packages is OpenFOAM, an open source software written in C++, that stores the flow variables at the cell centers, the so-called collocated mesh approach, figure 1 (left). Its counterpart is the staggered mesh approach, which stores velocity components at the cell faces and the pressure variable at the cell center, figure 1 (right). The advantage of collocated meshes is that conversion of grids to complex solution domains is more straightforward than in staggered grids [5]. Moreover, the data structure required to store the variables is simpler and more compact, leading to lower computational costs and easier implementation of code. Therefore, many industrial and academic software packages, such as ANSYS-FLUENT, STAR-CCM+, Code-Saturn and OpenFOAM, use this approach.

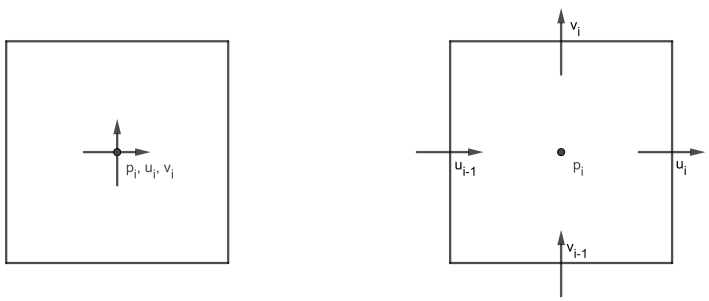


Figure 1: Collocated variable arrangement (left). Staggered variable arrangement (right)

However, this approach has two major disadvantages. First, using a collocated mesh in combination with a central discritisation stencil leads to the checkerboard problem, see [6,

7] for instance. This arises because cell-centered gradients are calculated using variables of directly neighbouring cells, which causes an odd-even decoupling, creating a checkerboard-like pattern. This problem is often circumvented by calculating gradients at the face directly from the neighbouring cells as opposed to interpolating the decoupled gradients to the face in between. This method was first posed by Rhie and Chow (1983) [8] and is also used in OpenFOAM because the laplacian operator in equation 1 can be calculated on a compact stencil, requiring a less complex data structure. This method does introduce some dissipation of kinetic energy however [6, 7, 8]. Second, the collocated mesh gives rise to a problem concerning the compressibility constraint, equation 2. Collocated meshes make use of an auxilary velocity field to comply to the compressibility constraint, this field stores its variables at the face centers, resulting in two separate velocity fields. The compressibility constraint of equation 2 is posed solely onto the face-centered velocities, making the cell-centered velocity field only approximately divergence free [9, 10].

These disadvantages lead to numerical dissipation and inaccuracies in solutions. Moreover, industrial codes often disregard fundamental properties of the differential operators of equations 1 and 2 on unstructured grids, in favor of stability. The dissipation of energy, caused by these inaccuracies, affects motions at the smallest length scales, i.e. turbulence [2]. By preserving the symmetries of the underlying differential operators, Verstappen and Veldman (2003) developed a non-dissipative discretization method, which preserves energy at all length scales to reduce numerical dissipation while also guaranteeing solution stability [1]. Trias et al. (2014) used this philosophy and generalized the discretization to unstructured collocated meshes [2]. In this thesis, this symmetry preserving discretization technique is successfully implemented into the source code of OpenFOAM, to increase the stability and accuracy of simulations of turbulence in incompressible flows of Newtonian fluids. Using standard cases, the Taylor-Green vortex, the lid-driven cavity flow and turbulent channel flow, solver stability and accuracy, order of temporal accuracy and sources of kinetic energy dissipation were examined.

This thesis is structured as follows. In section 2 the approach to discretize and solve the Navier-Stokes equation is discussed, in terms of the domain, symmetry requirements, pressure-velocity coupling, and temporal and spatial discretization. In section 3 the methods to implement the solver scheme into OpenFOAM are presented, discussing the scheme step-by-step, written in the required fundamental mathematical operations. Subsequently, in section 4, the results are presented for tests of accuracy, order of temporal accuracy and sources of kinetic energy dissipation using several test cases, including a Taylor-Green vortex, a lid-driven cavity and a channel flow. Then, in section 5, the findings are summarised and the conclusions are presented and discussed. Finally, recommendations for future research are presented in section 6.

2 Discretization methods

To approximate and solve the continuous Navier-Stokes equations, equations 1 and 2, they are discretized for unstructured collocated grids using FVM, while retaining the underlying symmetries of the differential operators, to reduce numerical dissipation and increase stability. To discretize these equations, the method of Trias et al. (2014) is closely followed [2].

2.1 Discretization of the domain

To discretize the Navier-Stokes equations using FVM with unstructured collocated grids, definitions of the geometric variables and relations have to be established first. The cell volumes are of a general polyhedral shape, with flat polygonal faces between two adjacent cells, as for example figure 2.

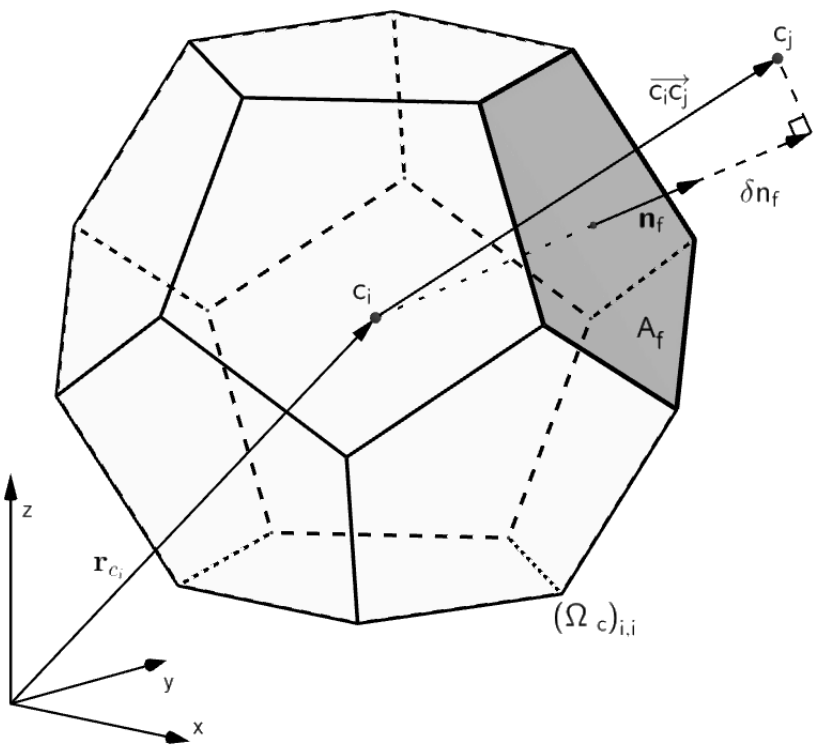


Figure 2: Example of polyhedral cell i, depicting relation to neighbouring cell j

For cell i, the collocated physical variables are stored at its centroid, C_i, with relation to the origin $\mathbf{r}_{c_i}$. Its collocated volume is given by $(\Omega_c)_{i,i}$. The face between cells i and j is labeled f. For a domain with n cells and m faces, the cells are labeled C_1 through C_n and the area of the faces between them are labeled A_1 through A_m. The normal of face f has its direction from C_i to C_j, for $i < j$, and is labeled $\mathbf{n}_f$. In this relation, cell i is referred to as the owner cell and cell j is referred to as the neighbour cell, labeled C_o and C_n respectively. The relation between cell centers C_i and C_j is given by $\overrightarrow{C_i C_j} = \mathbf{r}_{c_i} - \mathbf{r}_{c_j}$ and the length of its projection onto $\mathbf{n}_f$ is given by $\delta n_f = |\mathbf{n}_f \cdot \overrightarrow{C_i C_j}|$. Each face also has a corresponding staggered volume, labeled $(\Omega_s)_{f,f} = A_f \delta n_f$. This definition is discussed in section 2.6 and more extensively in appendix, section C.

2.2 Discretization of the Navier-Stokes equations

Using the geometric variables and relations defined in the previous section, equations 1 and 2 for FVM on unstructured collocated grids become:

$$\Omega_c \frac{d\mathbf{u}_c}{dt} + C(\mathbf{u}_s)\mathbf{u}_c + D\mathbf{u}_c + \Omega_c G_c \mathbf{p}_c = \mathbf{0}_c \tag{3}$$

$$M\mathbf{u}_s = \mathbf{0}_c \tag{4}$$

Here, the previously discussed cell-centered and face-centered control volumes appear in matrices Ω_c and Ω_s respectively. In three dimensions, $\Omega_c \in \mathbb{R}^{3n\times 3n}$ is a block diagonal matrix, $\Omega_c = I_3 \otimes \Omega_{c\#}$, where $I_3 \in \mathbb{R}^{3\times 3}$ is the identity matrix and $\Omega_{c\#} \in \mathbb{R}^{n\times n}$ is a square diagonal matrix containing the collocated control volumes, $(\Omega_c)_{i,i}$, as seen in figure 2. Similarly, $\Omega_s \in \mathbb{R}^{m\times m}$ is a square diagonal matrix containing the staggered control volumes, $(\Omega_s)_{f,f}$.

$\mathbf{u}_c \in \mathbb{R}^{3n}$ gives the cell-centered velocities in three dimensions and $\mathbf{p}_c \in \mathbb{R}^n$ gives the cell-centered pressures. Vector $\mathbf{u}_c$ is constructed as $\mathbf{u}_c = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)^T$, where the velocity components in each x_i-direction are given by $\mathbf{u}_i = \left((u_i)_1, (u_i)_2, \ldots, (u_i)_n\right)^T$. Equation 4 uses the auxiliary staggered velocity field, $\mathbf{u}_s \in \mathbb{R}^m$, containing the face normal components of the velocities. $\mathbf{u}_s$ is constructed as $\mathbf{u}_s = \left((u_i)_1, (u_i)_2, \ldots, (u_i)_m\right)^T$. The collocated and staggered velocity fields are related by the linear shift transformation, $\Gamma_{c\to s} \in \mathbb{R}^{m\times 3n}$, through:

$$\mathbf{u}_s := \Gamma_{c\to s}\mathbf{u}_c \tag{5}$$

Next, the differential operators are defined as follows. The convective operator, $C(\mathbf{u}_s) \in \mathbb{R}^{3n\times 3n}$, and the diffusive operator, $D \in \mathbb{R}^{3n\times 3n}$, are both block diagonal matrices constructed similar to Ω_c:

$$C(\mathbf{u}_s) = I_3 \otimes C_\#(\mathbf{u}_s) \tag{6}$$

$$D = I_3 \otimes D_\# \tag{7}$$

where $C_\#(\mathbf{u}_s)$ gives the scalar convection between neighbouring cells i and j, dependent on the velocity at the connecting face f, $\mathbf{u}_s$. Similarly, $D_\#$ gives the scalar diffusion between neighbouring cells, though independent of the face velocity. Finally, $G_c \in \mathbb{R}^{3n\times n}$ and $M \in \mathbb{R}^{n\times m}$ give the discretized operators for the collocated gradient and divergence, respectively. It is important to note that vectors $\mathbf{0}_c$ in equations 3 and 4 are of different length, namely $3n$ and n respectively.

2.3 Operator symmetry requirements

The continuous operators of equations 1 and 2 possess energy conserving properties due to their underlying symmetries. To spatially discretize these operators and conserve kinetic energy, it is important that these symmetries are translated to their discrete analogs, discussed in the previous section [1]. To do so, the global kinetic energy is considered, which can be expressed as $\|\mathbf{u}_c\|^2 = \mathbf{u}_c^T \Omega_c \mathbf{u}_c$. The evolution of this term can be obtained by left-multiplying equation 3 by $\mathbf{u}_c^T$ and summing the result with its transpose:

$$\frac{d}{dt}\left\|\mathbf{u}^2\right\| = -\mathbf{u}_c^T\big(C(\mathbf{u}_s) + C^T(\mathbf{u}_s)\big)\mathbf{u}_c - \mathbf{u}_c^T\big(D + D^T\big)\mathbf{u}_c - \mathbf{u}_c^T \Omega_c G_c \mathbf{p}_c - \mathbf{p}_c^T G_c^T \Omega_c^T \mathbf{u}_c \tag{8}$$

When there is no diffusion, $D = 0$, there should be no evolution in global kinetic energy. Therefore the convective and pressure term should obey:

$$\mathbf{u}_c^T\big(C(\mathbf{u}_s) + C^T(\mathbf{u}_s)\big)\mathbf{u}_c = 0 \tag{9}$$

$$\mathbf{u}_c^T \Omega_c G_c \mathbf{p}_c + \mathbf{p}_c^T G_c^T \Omega_c^T \mathbf{u}_c = 0 \tag{10}$$

Equation 9 leads to:

$$C(\mathbf{u}_s) = -C^T(\mathbf{u}_s) \tag{11}$$

which means operator C should be skew-symmetric to conserve global kinetic energy. To reduce the pressure terms to 0 as in equation 10, the incompressibility constraint is considered after interpolating $\mathbf{u}_c$, equations 4 and 5. This leads to:

$$-(\Omega_c G_c)^T = M\Gamma_{c\to s} \tag{12}$$

which can be verified to reduce both parts of equation 10 to 0:

$$\mathbf{u}_c^T \Omega_c G_c = -\mathbf{u}_c^T \Gamma_{c\to s}^T M^T = -(M\Gamma_{c\to s}\mathbf{u}_c)^T = 0 \tag{13}$$

$$G_c^T \Omega_c^T \mathbf{u}_c = -M\Gamma_{c\to s}\mathbf{u}_c = 0 \tag{14}$$

Therefore, the equality in equation 12 leads to the pressure term vanishing in equation 10. With both these terms in equation 8, this equation reduces to:

$$\frac{d}{dt}\|\mathbf{u}_c\|^2 = -\mathbf{u}_c^T(D + D^T)\mathbf{u}_c \leq 0 \tag{15}$$

The remaining term is always less than or equal to 0 because the viscosity term should be strictly dissipative. Therefore $(D + D^T)$ must be positive-definite. Moreover, although not strictly necessary, operator D is assumed to be symmetric, like its continuous analog $-\Delta$ [9]. The equalities and properties of the operators presented in this section so far, ensure a non-dissipative scheme.

2.4 Pressure-velocity coupling

In this section, the steps involved in solving the discretized Navier-Stokes equations, as seen in equations 3 and 4, are presented, using discrete operators that preserve the symmetries of their continuous analogs. Rewriting equation 3, using an explicit temporal scheme with a pressure term at time step $n+1$, the scheme with which the momentum equation is solved is obtained:

$$\frac{\mathbf{u}_c^{n+1} - \mathbf{u}_c^n}{\Delta t^n} = R(\mathbf{u}_c^n) - G_c \mathbf{p}_c^{n+1} \tag{16}$$

with $R(\mathbf{u}_c^n) := -\Omega_c^{-1}(C(\mathbf{u}_s^n)\mathbf{u}_c^n + D\mathbf{u}_c^n)$ and Δt^n as the time between step n and step $n+1$. The incompressibility constraint is solved implicitly:

$$M\mathbf{u}_s^{n+1} = \mathbf{0}_c \tag{17}$$

To solve the pressure-velocity coupling, the fractional step method is used [11, 12]. Using Helmholtz-Hodge vector decomposition, the staggered predictor velocity can be decomposed into a divergence free vector, $\mathbf{u}_s^{n+1}$ and a curl-free vector corresponding to the gradient of the cell-centered correction pressure, $G\tilde{\mathbf{p}}_c'$ [4]:

$$\mathbf{u}_s^p = \mathbf{u}_s^{n+1} + G\tilde{\mathbf{p}}_c' \tag{18}$$

where the cell-centered correction pressure is given by:

$$\tilde{\mathbf{p}}_c^{n+1} = \tilde{\mathbf{p}}_c^p + \tilde{\mathbf{p}}_c' \tag{19}$$

where $\tilde{\mathbf{p}}_c^p$ is the predictor pressure and $\tilde{\mathbf{p}}$ includes Δt, such that: $\Delta t^n \mathbf{p}_c^{n+1} = \Delta t^n \mathbf{p}_c^p + \Delta t^n \mathbf{p}_c'$. Operator G calculates the gradient of the cell-centered pressure at the faces and is defined using the divergence operator by:

$$G := -\Omega_s^{-1} M^T \tag{20}$$

Because $\mathbf{u}_s^{n+1}$ is divergence free, taking the divergence of equation 18, leads to the discrete Poisson equation:

$$M\mathbf{u}_s^p = MG\tilde{\mathbf{p}}_c' = L\tilde{\mathbf{p}}_c' \tag{21}$$

with:

$$L := MG = -M\Omega_s^{-1} M^T \tag{22}$$

giving the Laplacian operator, which is symmetric and negative-definite and therefore also holds the notion that operator $D = -\Delta$ is symmetric and positive definite.

To find an expression for G_c from G, an opposite operator corresponding to $\Gamma_{c\to s}$ is needed. This interpolation in the case of pressure gradients leads to a definition for the collocated gradient, from equation 20 :

$$G_c = -\Gamma_{s\to c}\Omega_s^{-1} M^T \tag{23}$$

The combination of equations 12 and 23 lead to an expression for for $\Gamma_{s\to c}$:

$$(\Omega_c \Gamma_{s\to c} \Omega_s^{-1} M^T)^T = M\Gamma_{c\to s} \tag{24}$$

and therefore:

$$\Gamma_{s\to c} := \Omega_c^{-1} \Gamma_{c\to s}^T \Omega_s \tag{25}$$

so that equation 23 becomes:

$$G_c := \Omega_c^{-1} \Gamma_{c\to s}^T M^T \tag{26}$$

An important observation to make now is that $\Gamma_{s\to c}\Gamma_{c\to s}$ does not exactly equal I. It introduces a small error and therefore: $\Gamma_{s\to c}\Gamma_{c\to s} \approx I$, which means that interpolating a variable from centroid to face and back leads to an error factor, which is further discussed in the appedix, section A. This is necessary to uphold equation 12, so that the pressure term in the evolution of global kinetic energy vanishes.

Now that the relevant discrete operators have been defined, an expression for the cell-centered predictor velocity, $\mathbf{u}_c^p$, is constructed, which in turn will be used to find the staggered predictor velocity, $\mathbf{u}_s^p$, as used in equation 21. To find this predictor velocity, an estimate of equation 16 is made, using the gradient of the predictor pressure, $G_c\tilde{\mathbf{p}}_c^p$:

$$\mathbf{u}_c^p = \mathbf{u}_c^n + \Delta t R(\mathbf{u}_c^n) - G_c \tilde{\mathbf{p}}_c^p \tag{27}$$

Next, the Poisson equation is posed to find the correction pressure, $\tilde{\mathbf{p}}_c'$, using equations 5 and 21:

$$L\tilde{\mathbf{p}}_c' = M\mathbf{u}_s^p = M\Gamma_{c\to s}\mathbf{u}_c^p \tag{28}$$

The predictor pressure in equation 27 will be given by $\tilde{\mathbf{p}}_c^n$, to give a good estimate which can be used in explicit calculations. This method corresponds to the van Kan projection method [3], in contrast to the Chorin projection method which takes $\mathbf{0}$ as its prediction [4]. When formulating this equation, the choice for the prediction pressure becomes important, as the Laplacian on the left-hand side (LHS) of equation 28 is evaluated on a compact stencil, whereas the pressure term on the right-hand side (RHS) is interpolated back and forth, thereby creating a broad stencil. By choosing $\mathbf{p}_c^n$ as a predictor, a larger part of $\mathbf{p}_c^{n+1}$ is calculated explicitly. The LHS implicitly introduces a small error term to the momentum equation, similar to the approach by Rhie and Chow (1983) and is sufficient to eliminate the checkerboard problem introduced by the broad stencil on the RHS [8]. After this, equation 18 can be used in combination with the interpolation of equation 25 to find an expression for $\mathbf{u}_c^{n+1}$. When calculating $\mathbf{u}_s^p$, the velocities are interpolated from cell to face, whereas here they are interpolated back from face to cell. Instead of writing $\Gamma_{s\to c}\Gamma_{c\to s}\mathbf{u}_c^p$, $\mathbf{u}_c^p$ is directly used, to avoid the small error that is introduced when interpolating from cell to face and back. The final expression then becomes:

$$\mathbf{u}_c^{n+1} = \mathbf{u}_c^p - \Gamma_{s\to c} G \tilde{\mathbf{p}}_c' \tag{29}$$

With the full expression written out this becomes:

$$\mathbf{u}_c^{n+1} = \mathbf{u}_c^p + \Omega_c^{-1}\Gamma_{c\to s}^T M^T L^{-1} M \Gamma_{c\to s}\mathbf{u}_c^p = (I - \Omega_c^{-1} P)\mathbf{u}_c^p \tag{30}$$

with $P = -\Gamma_{c\to s}^T \Omega_c^{-1} M^T L^{-1} M \Gamma_{c\to s}$, which is symmetric and positive-definite. This matrix is however not an exact projector, unless $\Gamma_{s\to c}\Gamma_{c\to s} = I$. This could give rise to spurious modes and non-physical components in the solution. This problem is related to the checkerboard problem and requires the elimination of these spurious modes. One way to do so is by linearising the convective term, to prevent the emergence of these modes [2]. A symmetry-preserving regularisation, which was first posed by Verstappen, is discussed in the appendix, section B [13, 14, 15]. As a result of the steps taken in this section we are able to calculate $\mathbf{u}_c^{n+1}$ and $\mathbf{p}_c^{n+1}$ from their corresponding previous time step values. In short, the following steps are taken in order to find the pressure and velocity terms at the next time step:

1. Find the predictor velocity:
   $\mathbf{u}_c^p = \mathbf{u}_c^n - \Delta t^n \Omega_c^{-1}(C(\mathbf{u}_s^n)\mathbf{u}_c^n + D\mathbf{u}_c^n) - G_c\tilde{\mathbf{p}}_c^n$

2. Find the correction pressure by solving the Poisson equation:
   $L\tilde{\mathbf{p}}_c' = M\Gamma_{c\to s}\mathbf{u}_c^p$

3. Update the velocity:
   $\mathbf{u}_c^{n+1} = \mathbf{u}_c^p - \Gamma_{s\to c} G\tilde{\mathbf{p}}_c'$

4. Update the pressure:
   $\tilde{\mathbf{p}}_c^{n+1} = \tilde{\mathbf{p}}_c^n + \tilde{\mathbf{p}}_c'$

2.5 Implicit time discretization

Up to this point, only explicit time integration was considered in the solver. Implicit time integration methods are generally more stable and more practical because larger time steps can be used [16]. Therefore the solver scheme in the previous section will be slightly edited to fit implicit calculation. In this formulation, the RHS terms of equation 16 will be taken at time step $n+1$. To solve this implicit formulation, the "pressure-implicit with splitting of operators" (PISO) approach by Issa (1986) is followed [17]. The momentum and compressibility equations, equations 16 and 17, are again the starting point, but the momentum equation is also formulated implicitly:

$$\frac{\mathbf{u}_c^{n+1} - \mathbf{u}_c^n}{\Delta t^n} = R(\mathbf{u}_c^{n+1}) - G_c \mathbf{p}_c^{n+1} \tag{31}$$

PISO uses an iterative process to approximate the terms at time step $n+1$. When iterated sufficiently, the solution becomes equal to equation 31. The iterative levels are denoted by k:

$$\frac{\mathbf{u}_c^{k} - \mathbf{u}_c^n}{\Delta t^n} = H(\mathbf{u}_s^k)\mathbf{u}_c^k - G_c \mathbf{p}_c^k \tag{32}$$

where H is an operator as a function of $\mathbf{u}_s^k$, that includes the convective and diffusive operators, given by:

$$H(\mathbf{u}_s^k) = -\Omega_c^{-1}(C(\mathbf{u}_s^k) + D) \tag{33}$$

To avoid non-linearity in the convective term we will approach operator $H(\mathbf{u}_s^k)$ by making it independent of $\mathbf{u}_s^k$ and instead dependent on $\mathbf{u}_s^n$ which stays constant during the iterative process:

$$H = -\Omega_c^{-1}(C(\mathbf{u}_s^n) + D) \tag{34}$$

Equation 32 now implicitly calculates the prediction of $\mathbf{u}_c^k$. To calculate $\mathbf{u}_c^1$, equation 32 is further simplified by taking the RHS terms at time step n:

$$\frac{\mathbf{u}_c^{1} - \mathbf{u}_c^n}{\Delta t^n} = H(\mathbf{u}_s^n)\mathbf{u}_c^n - G_c \mathbf{p}_c^n \tag{35}$$

PISO now moves on to the corrector step, which is the part that is iterated until the value of $\mathbf{u}_c^{k+1}$ has converged, it is given by:

$$\frac{\mathbf{u}_c^{k+1} - \mathbf{u}_c^n}{\Delta t^n} = H\mathbf{u}_c^k - G_c \mathbf{p}_c^k \tag{36}$$

In order to solve this, operator H is split into a diagonal part, A^d, which operates on $\mathbf{u}_c^{k+1}$ and an off-diagonal part, H^{od}, which operates on $\mathbf{u}_c^k$:

$$\mathbf{u}_c^{k+1} = \Delta t^n A^d \mathbf{u}_c^{k+1} + \Delta t^n H^{od}\mathbf{u}_c^k + \mathbf{u}_c^n - G_c \tilde{\mathbf{p}}_c^k \tag{37}$$

where $\Delta t^n \mathbf{p} = \tilde{\mathbf{p}}$. This can be rewritten as:

$$B\mathbf{u}_c^{k+1} = \Delta t^n H^{od}\mathbf{u}^k + \mathbf{u}_c^n - G_c \tilde{\mathbf{p}}_c^k \tag{38}$$

in which $B = I - \Delta t^n A^d$. This leads to:

$$\mathbf{u}_c^{k+1} = B^{-1}(\Delta t^n H^{od}\mathbf{u}_c^k + \mathbf{u}_c^n) - B^{-1}G_c\tilde{\mathbf{p}}_c^k \tag{39}$$

The incompressibility constraint is imposed onto $\mathbf{u}_c^{k+1}$:

$$M\Gamma_{c\to s}\mathbf{u}_c^{k+1} = \mathbf{0}_c \tag{40}$$

to equate the right-hand side terms of equation 39:

$$M\Gamma_{c\to s}B^{-1}(\Delta t^n H^{od}\mathbf{u}_c^k + \mathbf{u}_c^n) = M\Gamma_{c\to s}B^{-1}G_c\tilde{\mathbf{p}}_c^k \tag{41}$$

Then $\tilde{\mathbf{p}}_c^k$ is split using: $\tilde{\mathbf{p}}_c^k = \tilde{\mathbf{p}}_c^n + \tilde{\mathbf{p}}_c'$ and the explicit part is taken to the LHS:

$$M\Gamma_{c\to s}B^{-1}(\Delta t^n H^{od}\mathbf{u}_c^k + \mathbf{u}_c^n - G_c\tilde{\mathbf{p}}_c^n) = M\Gamma_{c\to s}B^{-1}G_c\tilde{\mathbf{p}}_c' \tag{42}$$

In the explicit solver, $\Gamma_{c\to s}\Gamma_{s\to c} \approx I$ is used to rewrite $M\Gamma_{c\to s}G_c\tilde{\mathbf{p}}_c' = M\Gamma_{c\to s}\Gamma_{s\to c}G\tilde{\mathbf{p}}_c' \approx MG\tilde{\mathbf{p}}_c' = L\tilde{\mathbf{p}}_c'$, to avoid checkerboarding and move the Poisson equation to a compact stencil. In the implicit case however, multiplication of $G\tilde{\mathbf{p}}_c'$ by B^{-1} happens after interpolation to the cell centers. To avoid this, the cell-centered scalar values on the diagonal of B^{-1} are interpolated to the faces, so that multiplication with $G\tilde{\mathbf{p}}_c'$ takes place without back and forth interpolation. This leads to:

$$M\Gamma_{c\to s}B^{-1}(\Delta t^n H^{od}\mathbf{u}_c^k + \mathbf{u}_c^n - G_c\tilde{\mathbf{p}}_c^n) = MB_s^{-1}G\tilde{\mathbf{p}}_c' \tag{43}$$

Where $B_s \in \mathbb{R}^{m\times m}$ is also a square diagonal matrix. To construct its diagonal, $\mathsf{diag}(B_s) \in \mathbb{R}^m$, one of the repeating diagonals in B, $\mathsf{diag}(B_\#) \in \mathbb{R}^n$ is taken. Where $B_\#$ and B are related by $B = I_3 \otimes B_\#$, for the 3 dimensional case. The values of $\mathsf{diag}(B_\#)$ are interpolated to the faces using weights of $\frac{1}{2}$, resulting in $\mathsf{diag}(B_s)$. The operation for this step, $\Pi_{c\to s}$, is described in the next section.

This Poisson equation, equation 43, can be solved in a similar fashion to the method used in the explicit case, to find $\tilde{\mathbf{p}}_c'$. We use $\tilde{\mathbf{p}}_c^k$ with: $\tilde{\mathbf{p}}_c^k = \tilde{\mathbf{p}}_c^n + \tilde{\mathbf{p}}_c'$ and equation 37 to find our value for $\mathbf{u}_c^{k+1}$. This concludes the correction step. This step can be iterated until $\mathbf{u}_c^{k+1}$ has converged to approximate $\mathbf{u}_c^{n+1}$. Using this scheme the implicitly discretized Navier-Stokes equations can be solved.

2.6 Discretization of the operators

In this section, the spatial discretization of the variables and operators is discussed, using the relations and constraints discussed in the previous sections. A simple case is used to explicitly show examples of the entries of each vector or matrix. A two-dimensional structured Cartesian grid with $n = 6$ and $m = 17$, as seen in figure 3, will be used. In this section only general operations and variables inside the domain are of interest, therefore values at domain boundary faces are simply denoted by "BC_f". Starting from the physical variables in equation 3 and 4, constraints for symmetry will be used to derive the remaining operators.

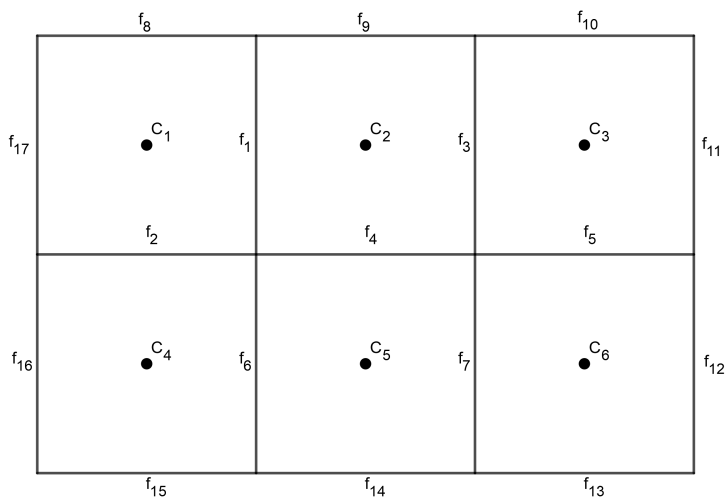


Figure 3: Cartesian structured grid with $n = 6$ and $m = 17$

Matrix $\Omega_c \in \mathbb{R}^{2n\times 2n}$ is block diagonal and contains the cell volumes $\Omega_\#$. In the example with $n = 6$ it is given by:

$$\Omega_c = I_2 \otimes \Omega_\# = \begin{pmatrix} \Omega_\# & 0 \\ 0 & \Omega_\# \end{pmatrix}, \qquad \Omega_\# = \begin{pmatrix} (\Omega_c)_{1,1} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & (\Omega_c)_{6,6} \end{pmatrix} \tag{44}$$

where $I_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix.
Vector $\mathbf{p}_c \in \mathbb{R}^n$ contains the scalar pressures at the cell centers, vector $\mathbf{u}_c \in \mathbb{R}^{2n}$ contains the scalar x- and y-components of the velocities at the cell centers and vector $\mathbf{u}_s \in \mathbb{R}^m$ contains the scalar projections of the interpolated velocities at the faces to the face normals:

$$p_c = \begin{pmatrix} p_1 \\ \vdots \\ p_6 \end{pmatrix}, \qquad \mathbf{u}_c = \begin{pmatrix} \mathbf{u}_{x,1} \\ \vdots \\ \mathbf{u}_{x,6} \\ \mathbf{u}_{y,1} \\ \vdots \\ \mathbf{u}_{y,6} \end{pmatrix}, \qquad \mathbf{u}_s = \begin{pmatrix} \mathbf{u}_{\perp 1} \\ \vdots \\ \mathbf{u}_{\perp 7} \\ \mathbf{u}_{BC8} \\ \vdots \\ \mathbf{u}_{BC17} \end{pmatrix} \tag{45}$$

The values of $\mathbf{u}_s$, the auxiliary velocity field, are constructed by interpolating the velocities at the face centers using equation 18, where matrix $\Gamma_{c\to s} \in \mathbb{R}^{m\times 2n}$, is a linear shift transformation. It is given by:

$$\Gamma_{c\to s} = N_s \Pi \tag{46}$$

In 2D, matrix $N_s \in \mathbb{R}^{m\times 2m}$ consists of two diagonal matrices, $N_{s,x} \in \mathbb{R}^{m\times m}$ and $N_{s,y} \in \mathbb{R}^{m\times m}$, which contain the x- and y- components of the face normals respectively. In the example N_s is given by:

$$N_s = \begin{pmatrix} N_{s,x} & N_{s,y} \end{pmatrix}, \qquad N_{s,x} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{BC8} & \ddots & \vdots \\ \vdots & & & & & & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & n_{BC17} \end{pmatrix} \tag{47}$$

All components are either 1 or 0 in this example with a structured cartesian grid. $N_{s,y}$ is constructed similarly, but with the y-components of the face normals.
Matrix $\Pi \in \mathbb{R}^{2m\times 2n}$ is a block diagonal matrix constructed with two times matrix $\Pi_{c\to s} \in \mathbb{R}^{m\times n}$, where $\Pi_{c\to s}$ interpolates scalars stored at the cell centers to the faces. To construct $\Pi_{c\to s}$, the interpolation weights from the cell-centers to the faces need to be known. These weights are chosen such that matrix C becomes skew-symmetric which is necessary to conserve global kinetic energy, as seen in equation 11. For the symmetry constraint of the convective term to hold, the weights need to be $\frac{1}{2}$ each. This is discussed below, in the discretization of operator C. In conclusion, this gives:

$$\Pi = I_2 \otimes \Pi_{c\to s} = \begin{pmatrix} \Pi_{c\to s} & 0 \\ 0 & \Pi_{c\to s} \end{pmatrix} \tag{48}$$

with:

$$\Pi_{c\to s} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} \\ \vdots & & & & & \vdots \\ \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} & \Pi_{BC8,1} \end{pmatrix} \tag{49}$$

In summary, Π and N_s combined result in an interpolation operator, $\Gamma_{c\to s}$, that is used to interpolate cell-centered vectors to their adjacent faces, retaining the normal components to the faces such that:

$$\left[\Gamma_{c\to s}\mathbf{u}_c\right]_f = \left[N_s(\Pi\mathbf{u}_c)\right]_f = \frac{1}{2}\left([\mathbf{u}_c]_i + [\mathbf{u}_c]_j\right)\cdot\mathbf{n}_f \tag{50}$$

Since the opposing transformation operator, $\Gamma_{s\to c}$, is defined by equation 25, a definition for the staggered control volumes, Ω_s is also needed. Matrix $\Omega_s \in \mathbb{R}^{m\times m}$ is a diagonal matrix that contains these staggered control volumes. It is given by:

$$\Omega_s = \begin{pmatrix} (\Omega_s)_{1,1} & 0 & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & (\Omega_s)_{7,7} & \ddots & & \vdots \\ \vdots & & \ddots & \Omega_{BC8} & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & \Omega_{BC17} \end{pmatrix} \tag{51}$$

where the individual control volumes of the faces f are given by:

$$(\Omega_s)_{f,f} = A_f \delta n_f \tag{52}$$

This definition of the control volumes is necessary to assure a conservation of total domain control volume, when interpolating between collocated and staggered formulation. In other words, to make sure the trace of Ω_c is equal to the trace of Ω_s, even for unstructured grids. This is shown in the appendix, section C.

To calculate the convection per cell using $C(\mathbf{u}_s)\mathbf{u}_c$ the flux at each face is multiplied by the interpolated velocity at each face and the result is summed. Matrix entry c_{ij} then contains the flux from cell i to cell j, at face f, given by ϕ_{ij}, multiplied by interpolation weight w_{jf} with which $\mathbf{u}_j$ is interpolated to face f. Entry c_{ji} then contains the flux from cell j to i, at face f, given by $\phi_{ji} = -\phi_{ij}$ multiplied by interpolation weight w_{if} with which $\mathbf{u}_i$ is interpolated to face f. Since C has to be skew symmetric, $w_{jf} = w_{if}$. Also, because we are interpolating two values: $w_{if} = 1 - w_{jf}$. Therefore skew-symmetry is only preserved when interpolation from cell to face occurs with weights $\frac{1}{2}$, regardless of spatial coordination and $\Gamma_{c\to s}$ is constructed with weights $\frac{1}{2}$, as seen before. Matrix $C(\mathbf{u}_s)$ can now be constructed. Matrix $C(\mathbf{u}_s) \in \mathbb{R}^{2n\times 2n}$, the convective matrix, is a block diagonal matrix and contains two times square matrix $C_\# \in \mathbb{R}^{n\times n}$:

$$C = I_2 \otimes C_\# = \begin{pmatrix} C_\# & 0 \\ 0 & C_\# \end{pmatrix}, \qquad C_\# = \begin{pmatrix} c_{1,1} & c_{1,2} & 0 & c_{1,4} & 0 & 0 \\ c_{2,1} & c_{2,2} & c_{2,3} & 0 & c_{2,5} & 0 \\ 0 & c_{3,2} & c_{3,3} & 0 & 0 & c_{3,6} \\ c_{4,1} & 0 & 0 & c_{4,4} & c_{4,5} & 0 \\ 0 & c_{5,2} & 0 & c_{5,4} & c_{5,5} & c_{5,6} \\ 0 & 0 & c_{6,3} & 0 & c_{6,5} & c_{6,6} \end{pmatrix} \tag{53}$$

where the off-diagonal terms are non-zero for entries $c_{i,j}$ when i and j are neighbouring cells. The off-diagonal terms are given by half the fluxes through the faces f between neighbouring cells i and j:

$$c_{i,j} = \frac{\phi_{ij}}{2} \tag{54}$$

where the fluxes are given by:

$$\phi_{ij} = A_f[\mathbf{u}_s]_f(\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f) \tag{55}$$

Where $\mathbf{n}_{\overrightarrow{ij}}$ is the face normal vector pointing outward of cell i to face j. The term $(\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f)$ ensures that fluxes are positive when $\mathbf{u}_s$ is pointing outwards of the cell and negative when pointing inwards. Components $c_{i,j}$ and $c_{j,i}$ are constructed similarly, except that $\mathbf{n}_{\overrightarrow{ij}} = -\mathbf{n}_{\overrightarrow{ji}}$, therefore they give oposite flux values. Finally, the diagonal terms, $c_{i,i}$ are given by:

$$c_{i,i} = \frac{1}{2} \sum_{f \in F_f(i)} \phi_f \tag{56}$$

Where $F_f(i)$ is the set of faces bordering cell i. The sum of fluxes through all faces is equal to the divergence of a cell, according to the divergence theorem. The divergence of a cell is given by equation 4 and is 0 because of incompressibility. Therefore the diagonal becomes 0 and C is skew-symmetric, hence does not contribute to global evolution of kinetic energy.

From the incompressibility constraint which is used to define the convective operator, the divergence operator can also be defined. Since the sum of all outward fluxes should be zero and the fluxes are given by equation 55, the entries of the divergence operator are given by:

$$m_{i,f} = (\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f) A_f \tag{57}$$

when face f is adjacent to cell i and zero otherwise. $\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f = 1$ when the normal vector and the outward face normal vector of cell i align and $\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f = -1$ when they are in opposite directions i.e. 1 and -1 for owner and neighbour cells respectively. In the example this leads to:

$$M = \begin{pmatrix} A_1 & A_2 & 0 & 0 & 0 & 0 & 0 & m_{BC1,8} & \dots & m_{BC1,17} \\ -A_1 & 0 & A_3 & A_4 & 0 & 0 & 0 & m_{BC2,8} & & m_{BC2,17} \\ 0 & 0 & -A_3 & 0 & A_5 & 0 & 0 & m_{BC3,8} & & m_{BC3,17} \\ 0 & -A_2 & 0 & 0 & 0 & A_6 & 0 & m_{BC4,8} & & m_{BC4,17} \\ 0 & 0 & 0 & -A_4 & 0 & -A_6 & A_7 & m_{BC5,8} & & m_{BC5,17} \\ 0 & 0 & 0 & 0 & -A_5 & 0 & -A_7 & m_{BC6,8} & \dots & m_{BC6,17} \end{pmatrix} \tag{58}$$

Subsequently, the center-to-face staggered gradient operator, $G \in \mathbb{R}^{m \times n}$, can be discretized using equation 20. Matrices Ω_s and M are given in turn by equations 51 and 57. The resulting combination leads to the entries of G:

$$g_{f,i} = -\frac{(\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f) A_f}{\delta n_f A_f} = -\frac{\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f}{\delta n_f} \tag{59}$$

when cells i and j are adjacent and zero otherwise. Then, matrix G is constructed as follows:

$$
\begin{pmatrix}
-\frac{1}{\delta n_f} & \frac{1}{\delta n_f} & 0 & 0 & 0 & 0 \\
-\frac{1}{\delta n_f} & 0 & 0 & \frac{1}{\delta n_f} & 0 & 0 \\
0 & -\frac{1}{\delta n_f} & \frac{1}{\delta n_f} & 0 & 0 & 0 \\
0 & -\frac{1}{\delta n_f} & 0 & 0 & \frac{1}{\delta n_f} & 0 \\
0 & 0 & -\frac{1}{\delta n_f} & 0 & 0 & \frac{1}{\delta n_f} \\
0 & 0 & 0 & -\frac{1}{\delta n_f} & \frac{1}{\delta n_f} & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{\delta n_f} & \frac{1}{\delta n_f} \\
g_{BC8,1} & g_{BC8,2} & g_{BC8,3} & g_{BC8,4} & g_{BC8,5} & g_{BC8,6} \\
\vdots & & & & & \vdots \\
g_{BC17,1} & g_{BC17,2} & g_{BC17,3} & g_{BC17,4} & g_{BC17,5} & g_{BC17,6}
\end{pmatrix}
\tag{60}
$$

From this, the center to center gradient $G_c \in \mathbb{R}^{2n\times n}$, as used in equation 3 can be easily found by:

$$
G_c = \Gamma_{s\to c} G \tag{61}
$$

Operators M and G, in turn, constitute the Laplacian operator, which is defined by equation 22. Finally, the diffusive operator, D, is a block diagonal matrix consisting of two times square matrix $D_\# \in \mathbb{R}^{n\times n}$. Each block given by:

$$
D_\# = -\nu L \tag{62}
$$

where ν is the kinematic viscosity. This leads to D being constructed as:

$$
D = I_2 \otimes D_\# = \begin{pmatrix} D_\# & 0 \\ 0 & D_\# \end{pmatrix}, \qquad D_\# = \begin{pmatrix}
d_{1,1} & d_{1,2} & 0 & d_{1,4} & 0 & 0 \\
d_{2,1} & d_{2,2} & d_{2,3} & 0 & d_{2,5} & 0 \\
0 & d_{3,2} & d_{3,3} & 0 & 0 & d_{3,6} \\
d_{4,1} & 0 & 0 & d_{4,4} & d_{4,5} & 0 \\
0 & d_{5,2} & 0 & d_{5,4} & d_{5,5} & d_{5,6} \\
0 & 0 & d_{6,3} & 0 & d_{6,5} & d_{6,6}
\end{pmatrix}
\tag{63}
$$

where the off-diagonal terms are non-zero for entries $d_{i,j}$ when i and j are neighbouring cells. The off-diagonal terms are given by:

$$
d_{i,j} = -\frac{\nu A_f}{\delta n_f} \tag{64}
$$

The diagonal terms are given by the negative of the sum of the off-diagonal terms:

$$
d_{ii} = -\sum_{j=1}^{i-1} d_{i,j} - \sum_{j=i+1}^{n} d_{i,j} \tag{65}
$$

Notice that $d_{i,j} = d_{j,i}$ and $d_{i,j} < 0$, therefore $d_{ii} > 0$ and D is symmetric and positive definite. This makes sense as the Laplacian operator, $L = \Delta = \nabla\cdot\nabla$ is negative definite and symmetric.

3 Implementation into OpenFOAM

Using the methods of pressure-velocity coupling of sections 2.4 and 2.5 in combination with the discretization presented in section 2.6, a solver scheme is composed for the simple Forward Euler time integration. For this, the steps at the end of section 2.4 are followed, with additional steps in between for clarity.

Step 1a: $\mathbf{u}_s^n = \Gamma_{c\to s}\mathbf{u}_c^n$

To find the staggered velocities at time step n at face f, $[\mathbf{u}_s]_f^n$, interpolate the cell-centered velocities of owner and neighbour cell at time step n, $[\mathbf{u}_c]_o^n$ and $[\mathbf{u}_c]_n^n$ respectively, to the faces, with weights $\frac{1}{2}$, then take the component normal to the face:

$$[\mathbf{u}_s]_f^n = \frac{1}{2}([\mathbf{u}_c]_o^n + [\mathbf{u}_c]_n^n)\cdot\mathbf{n}_f \tag{66}$$

Step 1b: Calculate $C(\mathbf{u}_s^n)\mathbf{u}_c^n$

The convective term, $C(\mathbf{u}_s^n)\mathbf{u}_c^n$, is calculated for all dimensional components of $\mathbf{u}_c^n$ separately. For each dimensional component, calculate the convection over every face and sum them to find the total convection of cell i in the respective dimensional component. To calculate the convection over face f in dimension x_i at time step n, $[C(\mathbf{u}_s)\mathbf{u}_c]_{f,x_i}^n$, take half of the sum of the respective cell-centered velocity components of the owner and neigbhour cells, $\frac{1}{2}([\mathbf{u}_c^n]_{o,x_i} + [\mathbf{u}_c^n]_{n,x_i})$ and the flux at face f and time step n, ϕ_f^n:

$$[C(\mathbf{u}_s)\mathbf{u}_c]_{f,x_i}^n = \frac{[\mathbf{u}_c]_{o,x_i}^n + [\mathbf{u}_c]_{n,x_i}^n}{2}\phi_f^n \tag{67}$$

Where ϕ_f^n is given by the face area, A_f, the staggered velocity at face f at time step n as calculated in step 1a, $[\mathbf{u}_s]_f^n$, and the dot product of the face normal of face f pointing outward of cell i and the normal of face f, $\mathbf{n}_{\overrightarrow{ij}}\cdot\mathbf{n}_f$:

$$\phi_f^n = A_f(\mathbf{n}_{\overrightarrow{ij}}\cdot\mathbf{n}_f)[\mathbf{u}_s]_f^n \tag{68}$$

Then, sum over all faces bordering cell i to get the convection in dimensional component x_i at time step n for cell i:

$$[C(\mathbf{u}_s)\mathbf{u}_c]_{i,x_i}^n = \sum_{f\in F_f(i)}[C(\mathbf{u}_s)\mathbf{u}_c]_{f,x_i}^n \tag{69}$$

Where $F_f(i)$ is the set of faces bordering cell i.

Step 1c: Calculate $D\mathbf{u}_c^n$

The diffusive term, $D\mathbf{u}_c^n$, is also calculated separately for each dimensional component of $\mathbf{u}_c$. For each dimensional component, calculate the diffusion over every face and sum them to find the total diffusion of cell i in the respective dimensional component. To calculate the diffusion over face f in dimension x_i at time step n, $[D\mathbf{u}_c]_{f,x_i}^n$, take the difference between the respective cell-centered velocity components of the owner and neighbour cell at time step n, $[\mathbf{u}_c]_{o,x_i} - [\mathbf{u}_c]_{n,x_i}^n$, the viscosity, ν, the face area, A_f, and divide by the length of the

normal component of the vector connecting the centroids of the owner and neighbour cells, δn_f, then correct the sign for owner and neighbour cells with $(\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f)$:

$$[D\mathbf{u}_c]^n_{f,x_i} = ([\mathbf{u}_c]^n_{n,x_i} - [\mathbf{u}_c]^n_{o,x_i}) \frac{\nu A_f}{\delta n_f} (\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f) \tag{70}$$

with:

$$\delta n_f = |\mathbf{n}_f \cdot \overrightarrow{c_o c_n}| \tag{71}$$

Then, sum over all faces bordering cell i to get the diffusion in dimensional component x_i at time step n for cell i:

$$[D\mathbf{u}_c]^n_{i,x_i} = \sum_{f \in F_f(i)} [D\mathbf{u}_c]^n_{f,x_i} \tag{72}$$

Step 1d: $\mathbf{u}_c^{p1} = \mathbf{u}_c^n - \Delta t \Omega_c^{-1} (C(\mathbf{u}_s^n)\mathbf{u}_c^n + D\mathbf{u}_c^n)$

A simple operation to calculate an intermediate cell-centered predictor velocity, $\mathbf{u}_c^{p1}$, which uses the collocated velocities at time step n, $\mathbf{u}_c^n$, the size of the time step, Δt, the cell volumes, Ω_c, the convection term as calculated in step 1a, $C(\mathbf{u}_s^n)\mathbf{u}_c^n$, and the diffusion term as calculated in step 1c, $D\mathbf{u}_c^n$, for each dimensional component at cell i:

$$[\mathbf{u}_c^{p1}]_{i,x_i} = [\mathbf{u}_c]^n_{i,x_i} - \Delta t^n [\Omega_c]_i^{-1} ([C(\mathbf{u}_s)\mathbf{u}_c]^n_{i,x_i} + [D\mathbf{u}_c]^n_{i,x_i}) \tag{73}$$

Step 1e: $\mathbf{u}_c^p = \mathbf{u}_c^{p1} - G_c \tilde{\mathbf{p}}_c^n$

To find the cell-centered predictor velocity, $\mathbf{u}_c^p$, correct the first predictor velocity, $\mathbf{u}_c^{p1}$, with the cell centered pressure gradient, $G_c \tilde{\mathbf{p}}_c^n$. Operation G_c calculates the cell-centered gradient between cell-centered scalars. This operator is given by a sequence of operations, $-\Omega_c^{-1} \Gamma^T_{c \to s} M^T$, as seen in equation 23. This sequence will be followed in order to find $G_c \tilde{\mathbf{p}}_c^n$. The first operation, $M^T \tilde{\mathbf{p}}_c^n$, calculates a value at every face f and time step n, using the face area, A_f, and the difference between the cell-centered pressures of the owner and neighbour cells, $\tilde{\mathbf{p}}_{c,o}^n - \tilde{\mathbf{p}}_{c,n}^n$:

$$[M^T \tilde{\mathbf{p}}_c]_f^n = A_f (\tilde{\mathbf{p}}_{c,o}^n - \tilde{\mathbf{p}}_{c,n}^n) \tag{74}$$

Subsequently, these face values are interpolated back to the cells for each dimensional component, as a result of the first and second operations, $\Gamma^T_{c \to s} M^T \tilde{\mathbf{p}}_c^n$. To do so we calculate the values for each face separately and then sum over all of them. The value of face f in dimensional component x_i at time step n, $[\Gamma^T_{c \to s} M^T \tilde{\mathbf{p}}_c]^n_{f,x_i}$, takes the x_i-component of the face normal, n_{f,x_i}, and half of the scalar value stored at the face, $\frac{1}{2}[M^T \tilde{\mathbf{p}}_c]_f^n$:

$$[\Gamma^T_{c \to s} M^T \tilde{\mathbf{p}}_c]^n_{f,x_i} = \frac{n_{f,x_i} [M^T \tilde{\mathbf{p}}_c]_f^n}{2} \tag{75}$$

To find the total value in dimensional component x_i for each cell i at time step n, we sum over all faces:

$$[\Gamma^T_{c\to s} M^T \tilde{\mathbf{p}}_c]^n_{i,x_i} = \sum_{f\in F_f(i)} [\Gamma^T_{c\to s} M^T \tilde{\mathbf{p}}_c]^n_{f,x_i} \tag{76}$$

Finally, these cell-centered values in each dimensional component are divided by the cell volumes, Ω_c, and negated as a result of all three operations, $-\Omega_c^{-1}\Gamma^T_{c\to s} M^T \tilde{\mathbf{p}}^n_c$. This results in the cell-centered pressure gradient, $G_c\tilde{\mathbf{p}}^n_c$, of cell i for each dimensional component x_i at time step n:

$$[G_c\tilde{\mathbf{p}}_c]^n_{i,x_i} = -[\Omega_c^{-1}]_i [\Gamma^T_{c\to s} M^T \tilde{\mathbf{p}}_c]^n_{i,x_i} \tag{77}$$

Which can then be used to correct the first predictor velocity, $\mathbf{u}^{p1}_s$, to find the cell-centered predictor velocity for cell i in dimensional component x_i:

$$[\mathbf{u}^p_c]_{i,x_i} = [\mathbf{u}^{p1}_c]_{i,xi} - [G_c\tilde{\mathbf{p}}_c]^n_{i,x_i} \tag{78}$$

Step 2a: $\mathbf{u}^p_s = \Gamma_{c\to s}\mathbf{u}^p_c$

To find the staggered predictor velocities, $\mathbf{u}^p_s$, interpolate the cell-centered predictor velocities, $\mathbf{u}^p_c$ to the faces, similar to step 1, for face f:

$$[\mathbf{u}^p_s]_f = \frac{1}{2}([\mathbf{u}^p_c]_o + [\mathbf{u}^p_c]_n)\cdot \mathbf{n}_f \tag{79}$$

Step 2b: $\tilde{\mathbf{p}}'_c = L^{-1}M\mathbf{u}^p_s$

To find the cell-centered correction pressure, $\tilde{\mathbf{p}}'_c$, we have to solve a Poisson equation, $L\tilde{\mathbf{p}}'_c = M\mathbf{u}^p_s$. To calculate the divergence of the staggered predictor velocities, $M\mathbf{u}^p_s$, sum the divergence of the predictor velocity at each face f, $[M\mathbf{u}^p_s]_f$. This is found by taking the staggered predictor velocity at face f, $[\mathbf{u}^p_s]_f$, the face area, A_f and the dot product of the outward pointing normal to face f and the normal to face f, $\mathbf{n}_{\overrightarrow{ij}}\cdot\mathbf{n}_f$:

$$[M\mathbf{u}^p_s]_f = [\mathbf{u}^p_s]_f A_f(\mathbf{n}_{\overrightarrow{ij}}\cdot\mathbf{n}_f) \tag{80}$$

Then sum these face values to find the total divergence of the staggered predictor velocities of cell i:

$$[M\mathbf{u}^p_s]_i = \sum_{f\in F_f(i)} [M\mathbf{u}^p_s]_f \tag{81}$$

Finally, solve for the Poisson equation implicitly to find the correction pressure at the cell centers for each cell i:

$$[\tilde{\mathbf{p}}'_c]_i = L^{-1}[M\mathbf{u}^p_s]_i \tag{82}$$

Where operation L^{-1} solves the Poisson equation implicitly for $\tilde{\mathbf{p}}'_c$, using one of the standard conventional standard OpenFOAM Poisson solvers, listed in the OpenFOAM User Guide [18].

Step 3a: Calculate $G\tilde{\mathbf{p}}'_c$

To find the gradient of the cell-centered correction pressure at face f, $[G\tilde{\mathbf{p}}'_c]_f$, substract the cell-centered correction pressure of the owner cell, $[\tilde{\mathbf{p}}'_c]_o$ from the cell-centered correction pressure of the neighbour cell, $[\tilde{\mathbf{p}}_c]_n$, and divide by the length of the normal component of the vector connecting the centroids of the owner and neighbour cells, δn_f:

$$[G\tilde{\mathbf{p}}'_c]_f = \frac{[\tilde{\mathbf{p}}'_c]_n - [\tilde{\mathbf{p}}'_c]_o}{\delta n_f} \tag{83}$$

Step 3b: $\mathbf{u}_c^{n+1} = \mathbf{u}_c^p - \Gamma_{s\to c} G\tilde{\mathbf{p}}'_c$

The cell-centered velocities at time step $n+1$ are calculated for each dimensional component of $\mathbf{u}_c$ separately. To find the cell-centered velocity at cell i at time step $n+1$ in dimensional component x_i, $[\mathbf{u}_c]_{i,x_i}^{n+1}$, interpolate the gradient of the cell-centered correction pressure at face f, $[G\tilde{\mathbf{p}}'_c]_f$, back to the cell centers, using $\Gamma_{s\to c}$, and substract the result from the cell-centered predictor velocity at cell i in dimensional component x_i as calculated in step 1e, $[\mathbf{u}_c^p]_{i,x_i}$. Operator $\Gamma_{s\to c}$ is defined by a sequence of operators, $\Omega_c^{-1}\Gamma_{c\to s}^T\Omega_s$, as seen in equation 25. We will follow this sequence to calculate the cell-centered correction pressure at cell i, $[G\tilde{\mathbf{p}}'_c]_i$. First, take the cell-centered correction pressure at face f and the staggered volume at face f, $[\Omega_s]_f = A_f\delta n_f$:

$$[\Omega_s G\tilde{\mathbf{p}}'_c]_f = [G\tilde{\mathbf{p}}'_c]_f A_f \delta n_f \tag{84}$$

Subsequently, interpolate these face values back to the cell centers, for each dimensional component as a result of the first two operations, $\Gamma_{c\to s}^T\Omega_s G\tilde{\mathbf{p}}'_c$. Calculate this for each face f and dimensional component x_i and then sum the resulting terms. To find the value of each face, take the x_i-component of the face normal and half of $[\Omega_s G\tilde{\mathbf{p}}'_c]_f$:

$$[\Gamma_{c\to s}^T\Omega_s G\tilde{\mathbf{p}}'_c]_{f,x_i} = \frac{n_{f,x_i}[\Omega_s G\tilde{\mathbf{p}}'_c]_f}{2} \tag{85}$$

Then sum these values at the faces to find the values for cell i in dimensional component x_i:

$$[\Gamma_{c\to s}^T\Omega_s G\tilde{\mathbf{p}}'_c]_{i,x_i} = \sum_{f\in F_f(i)} [\Gamma_{c\to s}^T\Omega_s G\tilde{\mathbf{p}}'_c]_{f,x_i} \tag{86}$$

The final step in the interpolation is to divide the cell-centered values by Ω_c, to find the cell-centered correction pressure at cell i in dimensional component x_i, $[G\tilde{\mathbf{p}}'_c]_{i,x_i}$

$$[G\tilde{\mathbf{p}}'_c]_{i,x_i} = [\Omega_c^{-1}]_i [\Gamma_{c\to s}^T\Omega_s G\tilde{\mathbf{p}}'_c]_{i,x_i} \tag{87}$$

To find the cell-centered velocity at cell i at time step $n+1$ in dimensional component x_i, $[\mathbf{u}_c]_{i,x_i}^{n+1}$, substract the cell-centered correction pressure at cell i in dimensional component x_i, $[G\tilde{\mathbf{p}}'_c]_{i,x_i}$, from the cell-centered predictor velocity at cell i in dimensional component x_i as calculated in step 1e, $[\mathbf{u}_c^p]_{i,x_i}$:

$$[\mathbf{u}_c]_{i,x_i}^{n+1} = [\mathbf{u}_c^p]_{i,x_i} - [G\tilde{\mathbf{p}}'_c]_{i,x_i} \tag{88}$$

This concludes the operations to find the cell-centered velocities at time step $n+1$. $\mathbf{u}_c^{n+1}$ will be used as input to calculate the cell-centered pressures and velocities at the next time step.

Step 4: $p_c^{n+1} = \tilde{\mathbf{p}}_c^n + \tilde{\mathbf{p}}_c'$

To find the cell-centered pressure for cell i at time step $n+1$, $[\tilde{\mathbf{p}}_c]_i^{n+1}$, add the cell-centered pressure of cell i at time step n, $[\tilde{\mathbf{p}}_c]_i^n$, and the correction pressure at cell i as calculated in step 2b, $[\tilde{\mathbf{p}}_c']_i$:

$$[\tilde{\mathbf{p}}_c]_i^{n+1} = [\tilde{\mathbf{p}}_c]_i^n + [\tilde{\mathbf{p}}_c']_i \tag{89}$$

This concludes the operations to find the cell-centered pressures at time step $n+1$. $\tilde{\mathbf{p}}_c^{n+1}$ will be used as input to calculate the cell-centered pressures and velocities at the next time step.

Summary

In summary, the fully discretised equations that need to be solved per step are presented, expressed in the basic variables that are necessary to solve them: $[\mathbf{u}_c]_i^n$, $[\tilde{\mathbf{p}}_c]_i^n$, $[\Omega_c]_i$, A_f, $\overrightarrow{c_o c_n}$, $\mathbf{n}_f$, $\mathbf{n}_{\overrightarrow{ij}}$, ν and Δt^n.

1a. Interpolate to find the staggered velocity at every face

$$[\mathbf{u}_s]_f^n = \frac{1}{2}([\mathbf{u}_c]_o^n + [\mathbf{u}_c]_n^n)\cdot \mathbf{n}_f \tag{90}$$

1b. Calculate the convective term at every cell for every component

$$[C(\mathbf{u}_s)\mathbf{u}_c]_{i,x_i}^n = \sum_{f\in F_f(i)} \frac{[\mathbf{u}_c]_{o,x_i}^n + [\mathbf{u}_c]_{n,x_i}^n}{2} A_f(\mathbf{n}_{\overrightarrow{ij}}\cdot \mathbf{n}_f)[\mathbf{u}_s]_f^n \tag{91}$$

1c. Calculate the diffusive term at every cell for every component

$$[D\mathbf{u}_c]_{i,x_i}^n = \sum_{f\in F_f(i)} ([\mathbf{u}_c]_{o,x_i}^n - [\mathbf{u}_c]_{n,x_i}^n)\frac{\nu A_f}{|\mathbf{n}_f\cdot \overrightarrow{c_o c_n}|}(\mathbf{n}_{\overrightarrow{ij}}\cdot \mathbf{n}_f) \tag{92}$$

1d. Calculate the cell-centered first predictor velocity at every cell for every component

$$[\mathbf{u}_c^{p1}]_{i,x_i} = [\mathbf{u}_c]_{i,x_i}^n - \Delta t^n [\Omega_c]_i^{-1}([C(\mathbf{u}_s)\mathbf{u}_c]_{i,x_i}^n + [D\mathbf{u}_c]_{i,x_i}^n) \tag{93}$$

1e. Calculate the cell-centered predictor velocity using the cell-centered pressure gradient at every cell for every component

$$[\mathbf{u}_c^p]_{i,x_i} = [\mathbf{u}_c^{p1}]_{i,xi} + [\Omega_c^{-1}]_i \sum_{f\in F_f(i)} \frac{n_{f,x_i} A_f(\tilde{\mathbf{p}}_{c,o}^n - \tilde{\mathbf{p}}_{c,n}^n)}{2} \tag{94}$$

2a. Interpolate to find the staggered predictor velocity at every face

$$[\mathbf{u}_s^p]_f = \frac{1}{2}([\mathbf{u}_c^p]_o + [\mathbf{u}_c^p]_n)\cdot \mathbf{n}_f \tag{95}$$

2b. Solve the Poisson equation to find the cell-centered correction pressure at every cell

$$[\mathbf{p}_c']_i = L^{-1} \sum_{f\in F_f(i)} [\mathbf{u}_s^p]_f A_f(\mathbf{n}_{\overrightarrow{ij}}\cdot \mathbf{n}_f) \tag{96}$$

3a. Calculate the gradient of the cell-centered correction pressure at each face

$$[G\tilde{\mathbf{p}}'_c]_f = \frac{[\tilde{\mathbf{p}}'_c]_n - [\tilde{\mathbf{p}}'_c]_o}{\delta n_f} \tag{97}$$

3b. Correct the cell-centered correction pressure with the gradient of the cell-centered correction pressure at every cell to find the cell-centered velocity at the next time step at every cell for every component

$$[\mathbf{u}_c]^{n+1}_{i,x_i} = [\mathbf{u}^p_c]_{i,x_i} - [\Omega_c^{-1}]_i \sum_{f \in F_f(i)} \frac{n_{f,x_i}[G\tilde{\mathbf{p}}'_c]_f A_f \delta n_f}{2} \tag{98}$$

4 Calculate the cell-centered pressure at the next time step at every cell

$$[\tilde{\mathbf{p}}_c]^{n+1}_i = [\tilde{\mathbf{p}}_c]^n_i + [\tilde{\mathbf{p}}'_c]_i \tag{99}$$

4 Results

In this section the solver that was built in the previous sections is tested and compared to standard OpenFOAM solvers and literature. First, the evolution of kinetic energy by physical and numerical dissipation is examined using a standard Taylor-Green vortex. The symmetry preserving solver is compared to a standard incompressible solver in OpenFOAM. Next, effects of the pressure prediction and spatial discretization, which are specific to the symmetry preserving scheme, are examined using the same case. Effects of temporal and spatial discretization will be discussed subsequently. Once a solid basic understanding of the solver is attained, several higher order temporal schemes, widely used in literature, are introduced and tested as well. Subsequently, temporal consistency is examined using a standard lid-driven cavity case and origins of errors in the solver are discussed. Finally, after gaining a thorough understanding of the symmetry preserving solver, numerous different simulations of a channel flow case are presented, for a representation of the solver's performance in more realistic problems.

4.1 Kinetic energy dissipation - Taylor Green vortex

To test solver performance a Taylor-Green vortex case was used to examine evolution of kinetic energy. The case consists of vortices in a domain with cyclic boundaries. The initial velocity and pressure profiles are described by continuous functions, which allows analytical calculation of the evolution of kinetic energy through time, to which several solvers are bench-marked. The analytical functions of the Taylor-Green vortex are given by:

$$u_x(x, y, t) = \sin(x)\cos(y)e^{-2\nu t} \tag{100}$$

$$u_y(x, y, t) = -\cos(x)\sin(y)e^{-2\nu t} \tag{101}$$

$$p(x, y, t) = \frac{1}{4}(\cos(2x) + \cos(2y))e^{-4\nu t} \tag{102}$$

The case was set up on a two-dimensional domain with $0 \leq x \leq 2\pi$, $0 \leq y \leq 2\pi$. The initial velocity field is depicted in figure 4. The Reynold's number was given by $\frac{1}{\nu}$. The case was simulated with in the inviscid limit with $\nu = 0$ and with $\nu = 0.001$, leading to $Re = \infty$ and $Re = 1000$ respectively. The inviscid case should stay steady over time, as no energy is dissipated by the diffusive term. In the viscous case the energy should decay according to the analytical solution given by:

$$E_a = E_0 e^{-4\nu t} \tag{103}$$

which is obtained from integrating $|\mathbf{u}|^2$ over the whole domain.

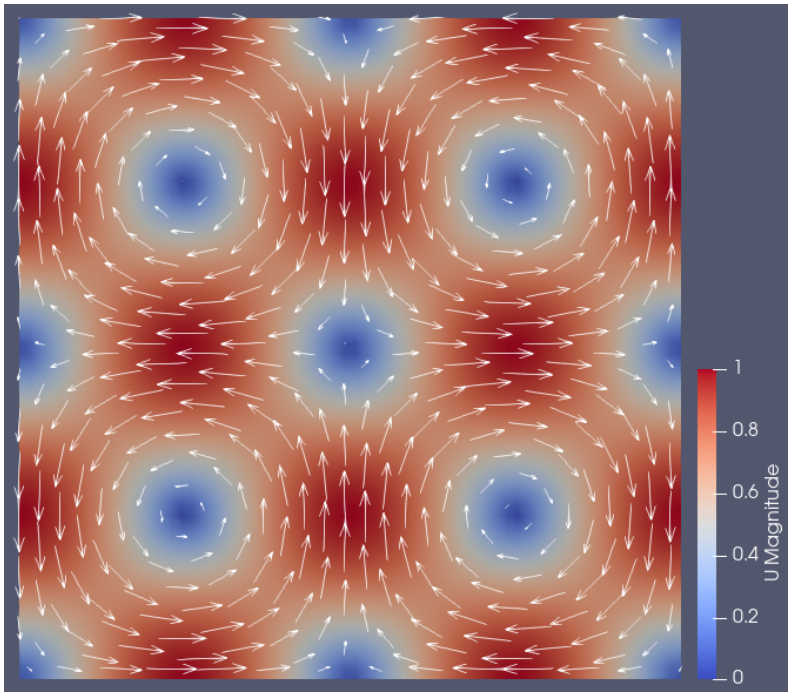


Figure 4: Initial velocity field of the Taylor-Green vortex case

The case was run on a two-dimensional regular Cartesian grid with 64x64 cells and on a skew grid, to view the effect of meshing on numerical dissipation. The skew grid was created such that the edges of the domain contain regular spaced grid cells and cell edge lengths decrease with a constant stretching ratio towards the middle of the domain in such a way that the largest edges of the domain are four times larger than the smallest ones, see figure 5 for an example of a skew 8x8 grid.

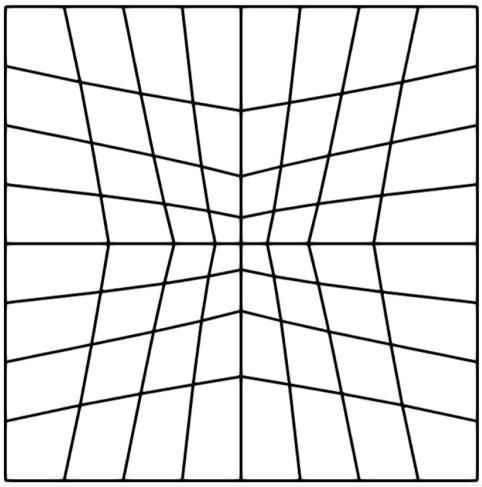


Figure 5: Example of an 8x8 skew grid

The case was run for three full rotational periods, corresponding to $2\pi * 3 \approx 18.85$, with $\Delta t = 0.01$, chosen such that, even on skew grids, the CFL-number remained low enough everywhere in the domain (CFL < 0.25).

In figure 6 a comparison is made between the implicit symmetry preserving method and a standard solver in OpenFOAM for incompressible flow, icoFoam. The case was set up on both a 64x64 Cartesian and skew grid, with either Re=∞ or Re=1000. The evolution of the kinetic energy through time of each solver is corrected by the analytical solution of the evolution of kinetic energy over time, to give a relative performance indication.

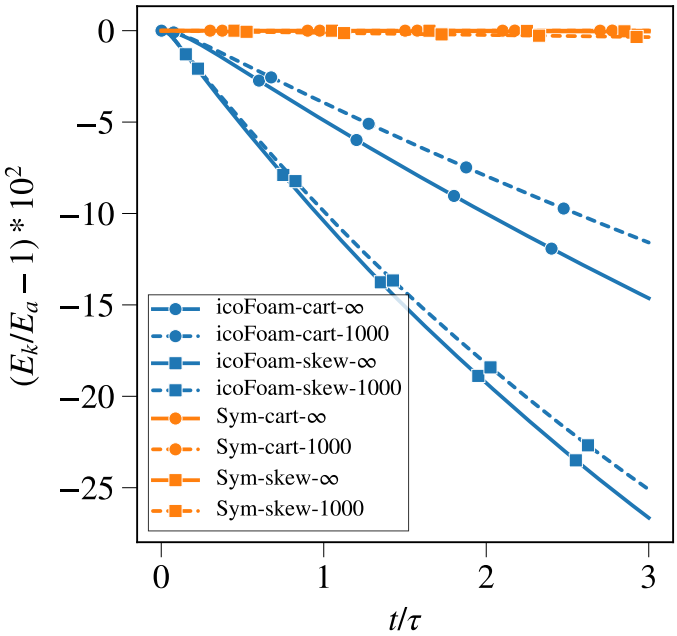


Figure 6: icoFoam solver and symmetry preserving solver for combinations of a [Cartesian, skew] grid and $[Re = \infty, Re = 1000]$

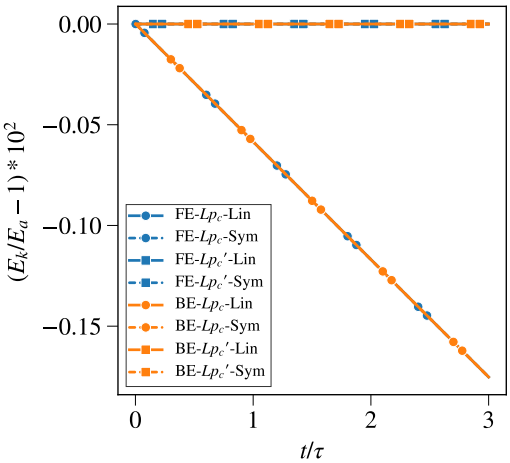
It can be seen that the symmetry preserving method performs much better in all different cases. The icoFoam solver seems especially inaccurate on the skew grids, with introduction of physical diffusion slightly improving performance. This is because the solver introduces a lot of numerical dissipation in favour of stability but in spite of accuracy. All symmetry preserving methods seem to perform very accurately, with some small dissipation in the viscid case on a skew grid.

To further investigate the solver, a deeper look is taken into two of the main differences between the symmetry preserving solver and a standard solving scheme. The first difference is the pressure predictor term, which is taken using the Van Kan method, with $\tilde{\mathbf{p}}_c^p = \tilde{\mathbf{p}}_c^n$, in contrast to the Chorin method which uses $\tilde{\mathbf{p}}_c^p = \mathbf{0}$ [3, 4]. The former method is referred to with Lp_c' (p_c' is the pressure correction), the latter is referred to with Lp_c (p_c is the full pressure). The second difference is the spatial discretization method. Midpoint interpolation is used as well as projected distances between centroids for calculation of the gradients, in contrast to using linear interpolation and the actual distances. These variables were examined using combinations of Cartesian and skew grids, with Re=∞ and Re=1000, using both Forward and Backward Euler time integration methods.

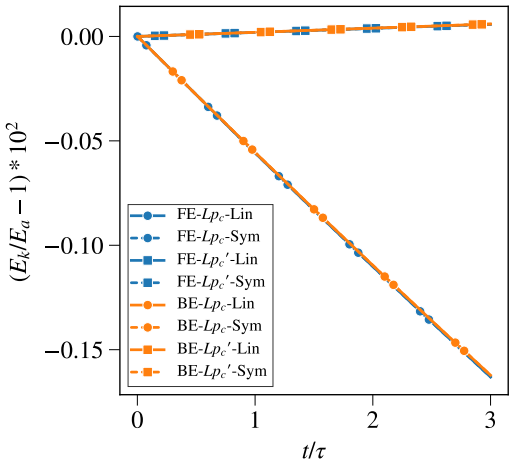
On the Cartesian grid without viscosity, figure 7a, it can be seen that the pressure prediction method is of the highest importance. The Van Kan method seems to give a good prediction of the pressure, leaving a relatively smaller term that is calculated using the com-

pact stencil Laplacian in the Poisson equation, leading to a more accurate prediction of the dissipation of kinetic energy. The time integration method does not seem to play any roll in this case. Both Forward Euler and Backward Euler methods are of a first-order temporal accuracy, with their biggest difference in the stability, which does not seem to play a role here. Finally, the graphs for the linear discretization method and the symmetry preserving method overlap because the symmetry preserving method simplifies to the linear method on regular Cartesian grids.

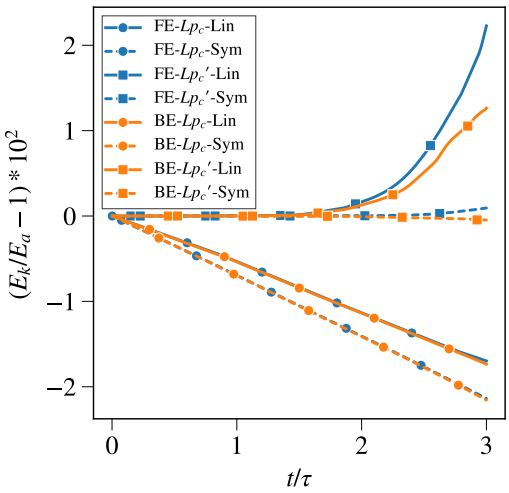
When viscosity is introduced, see figure 7b, the results are almost identical, with the biggest difference that the Van Kan method graphs all slightly tilt upwards. This implies a slight underestimation of the physical dissipation. Since the only difference with figure 7a is the introduction of viscosity, the difference in the output has to lie in the effect of the diffusive term. This term is based on estimations of the velocity gradients. The velocity gradients are calculated using a projected distance between centroids, $|n_f \cdot \overrightarrow{c_o c_n}|$, which varies from the actual distance depending on mesh orthogonality. Therefore, it is expected to see differences when the case is run on a skew grid. Also, if the upward tilt is a product of spatial discretization, positive changes are expected to be found in the grid-refinement study, where the gradients are more accurately captured.



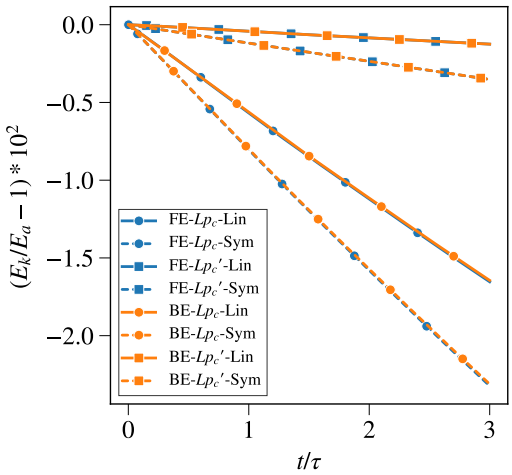
(a) Cartesian grid with $Re = \infty$



(b) Cartesian grid with $Re = 1000$



(c) Skew grid with $Re = \infty$



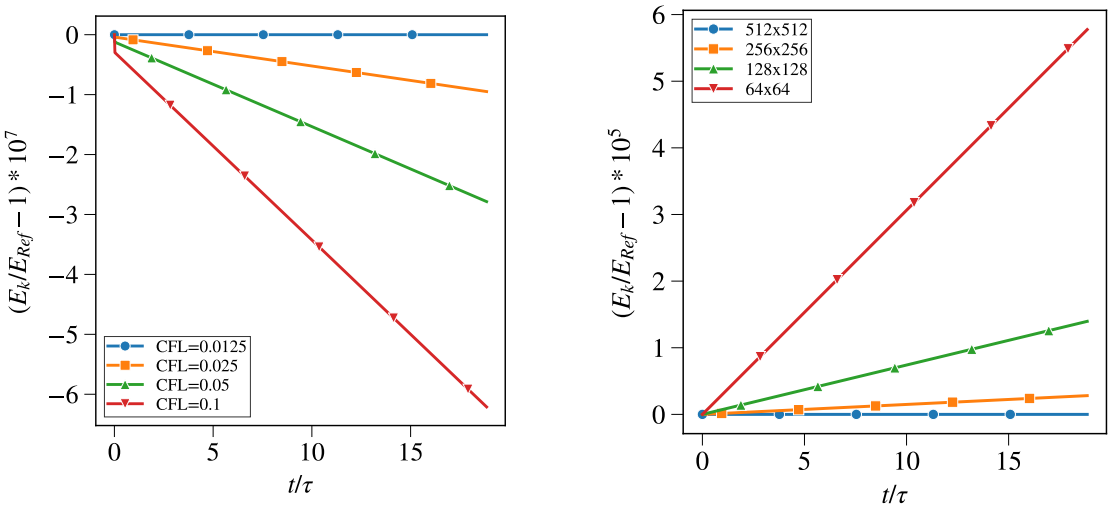
(d) Skew grid with $Re = 1000$

Figure 7: Effects of changing temporal scheme (Forward Euler, Backward Euler), Poisson equation (Lp_c', Lp_c) and interpolation method (symmetric, linear) on kinetic energy evolution on Cartesian and skew grids with $Re = \infty$ and $Re = 1000$

When the case is run on a skew grid without viscosity, see figure 7c, the effects of all components of the symmetry preserving solver can be examined even better. The Chorin methods still overestimate the dissipation of energy, however, some of the Van Kan methods seem to become unstable. The absence of dissipation in the solvers make them susceptible for the creation of spurious modes, which are not damped by any form of dissipation. The linear interpolation seems to introduce some kinetic energy in this case. In the Chorin method this leads to a more accurate solution. This seems to be a compensation of errors, because the Van Kan method with symmetry preserving interpolation is very accurate. In the Van Kan method, the linear interpolation seems to lead to instability. The symmetry preserving method seems to introduce more stability, which is one of the main reasons to use this spatial discretization technique, although a slight upward tilt can also be seen, hinting to some remaining instability. Lastly, a slight difference can be seen between Forward and Backward Euler time integration, where Backward Euler seems to lead to a more stable solution.

Finally, viscosity was introduced to the case on the skew grid, see figure 7d. With this combination, all solvers show overestimation of the dissipation of kinetic energy. The same pattern can be seen as in the previous figures, with the most notable difference that the linear discritisation leads to a better estimate in both the Chorin as the Van Kan methods. Again, the only difference with the previous figure is the introduction of viscosity, and therefore the effects of the diffusive term. The choice of method of calculation of the gradients in the linear discretization seems to give a better estimate and therefore lead to a more accurate solution. As mentioned before, the symmetry preserving method does not give the best estimate of dissipation on skew grids, it does however warrant stability.

In addition to these cases, the case on a Cartesian grid with Re=1000 was used again for temporal and spatial refinement, to get a feeling for the effects of the discretization of the case. To examine the temporal discretization error, the case was run with $\Delta t = \{0.01, 0.005, 0.0025, 0.00125\}$, respectively with CFL = $\{0.1, 0.05, 0.025, 0.125\}$. In figure 8a, the resulting evolutions of kinetic energy were plotted, relative to the reference case with $\Delta t = 0.00125$. It can be seen that higher temporal precision leads to a slight upward tilt in the solution, meaning temporal inaccuracy leads to overestimation of dissipation. This effect is, however, very small, as the relative differences are of order 10^{-7}.



(a) Temporal convergence study

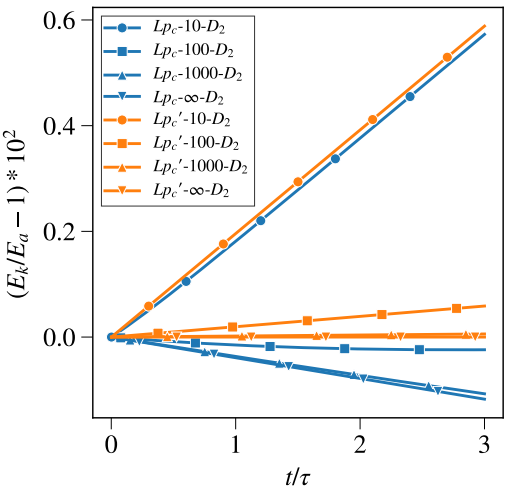
(b) Spatial convergence study

Figure 8: Temporal and spatial convergence studies for the Taylor-Green case on Cartesian grids, varying CFL-numbers and number of grid cells

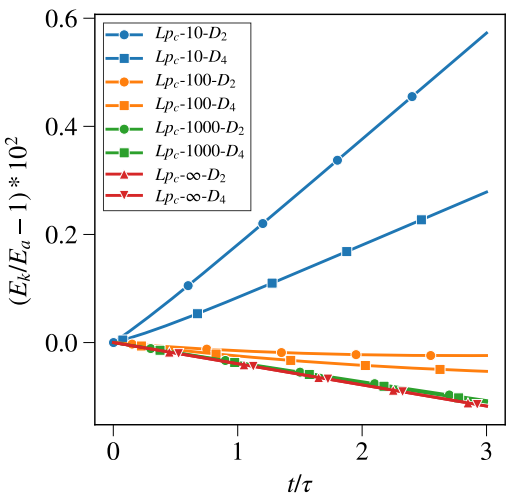
Additionaly, spatial refinement was done using grids of NxN = {64x64, 128x128, 256x256, 512x512}. To comply with the CFL-number requirements, the cases were all run with $\Delta t = 0.00125$, leading to $CFL \approx 0.1$ on the finest grid. In figure 8b, the different results were plotted relative to the reference case on the 512x512 grid. It can be seen that spatial refinement leads to a downward trend in dissipation of kinetic energy, suggesting coarseness in the spatial discretization leads to an underestimation of energy dissipation. This is in line with the findings of figures 7a and 7b, which suggested that the inaccuracy arises from the diffusive term and could be solved by better estimates of the velocity gradient, for example by using grid refinement. This effect is also approximately three orders of magnitude greater than the effect of temporal refinement. When sufficiently refined both temporally and spatially, the solution in figure 7b can be lowered down to the analytical solution (not shown).

The overestimation of dissipation by the Chorin method and the underestimation of dissipation caused by the diffusive term, could potentially cancel each other out. This was shown using a variety of cases with differing Reynold's numbers using the Van Kan and Choring method. The results are shown in figure 9a. Here, a higher order temporal scheme, Runge-Kutta 3, was used to decrease the effects of the temporal error, this scheme will be explained in the next paragraph. When looking at the Chorin method results, an increase in Reynold's number causes the ratio between the kinetic energy of the simulation and the analytical solution to shift down. At the inviscid limit, the underestimation of dissipation has completely dissapeared and overestimation is taking place. In contrast, the Van Kan method shows similar underestimation for low Reynold's numbers, but converges to a dissipation free solution in the inviscid limit. It can be seen that some cases of the Chorin method, for example with Re=100, give excellent results in the sense that they seem to be low-dissipative. However, upon decreasing viscosity in the case, Re=1000 and inviscid, it can be seen that the error in the diffusive term was in fact cancelling the error of the pressure term. This should be kept in mind, especially in the Channel Flow case, where the Reynold's number becomes

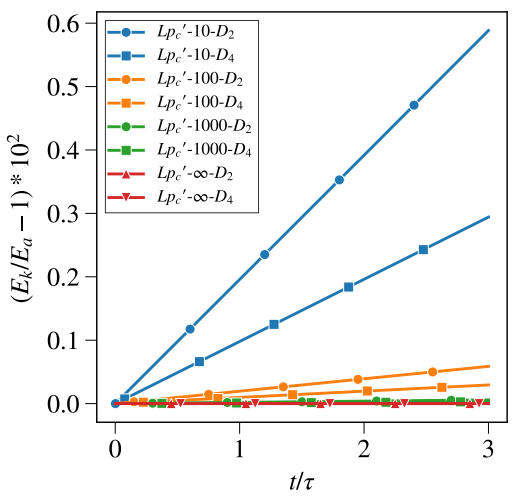
lower when approaching the wall. To further investigate the effects, a higher order diffusive term was discretized, see appendix, section D. This method effectively increases the accuracy of the gradients in the diffusive term from second (D_2) to fourth (D_4) order. The resulting tilts in the numerical dissipation for the Chorin method can be seen in figure 9b. It can be seen that all graphs creep towards the inviscid solution, thereby changing the error cancellation effect which was seen in figure 9a. For the Van Kan method, see figure 9c, the graphs also tilt downward towards the inviscid limit, but because the effect of error cancellation is not significant, the solutions give better predictions of the true dissipation of kinetic energy.



(a) D_2 diffusive term results for Lp_c and Lp_c'



(b) Effect of diffusive term for Lp_c



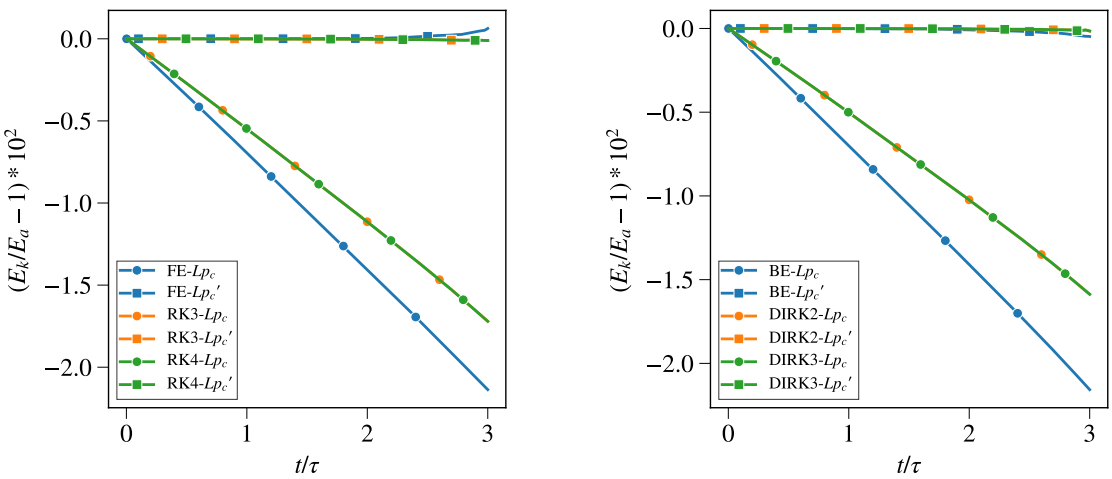
(c) Effect of diffusive term for Lp_c'

Figure 9: Balancing effect of the diffusive term and pressure prediction method, shown with regular (D_2) and higher order (D_4) diffusive terms at different Reynold's numbers.

Several other temporal schemes were also tested on these grids, in addition to the Forward and Backward Euler methods, to see the how the solver functions with higher order

temporal discretization schemes. These schemes can be subdivided into explicit and implicit Runge-Kutta schemes, with Runge-Kutta 3 and 4 (RK3 and RK4) in the former category and Diagonally Implicit Runge Kutta 2 and 3 (DIRK2 and DIRK3) in the latter. Similar to the Forward Euler method, the Runge-Kutta method uses information at the current time step to extrapolate the solution at the next overesttime step. Runge-Kutta, however, uses intermediate points to get a higher order estimation of the solution at the next overesttime step. To do so, the method makes use of so-called Butcher tableaus to assign weights to the intermediate steps, as first described by Butcher [19]. The difference between the RK methods and DIRK methods is that the diagonal of the RHS of the momentum equation is solved implicitly, similar to the PISO method. The number in the name of the method signifies the amount of intermediate steps taken. Further descriptions of the methods can be found in [7], [20] and [21]. In this thesis, an implementation method of RK in OpenFOAM was used similar to the one used by Vuorinen et al. [22].

In most of the previous test cases the differences between Forward Euler and Backward Euler were negligible, except on the skew grid with no viscosity. It was also found that the higher order schemes gave similar results in most cases, therefore only the skew case without viscosity is shown here, see figures 10a and 10b. Figure 10a shows that the difference between the Forward Euler and the higher order schemes are most pronounced when using the Chorin method. The differences between RK3 and RK4 appear to be of less significance. Lastly, the small instability that occurs in the Van Kan method with Forward Euler seems to be stabilised much more with the higher order schemes. However, the symmetry preserving method does not seems to be able to warrant stability on highly distorted grids when using Forward and Backward Euler. This stems from the pressure gradient interpolation, since the convective term should not introduce energy and the diffusive term is absent. The increase in temporal order of the solver does seem to positively affect this.



(a) Explicit solvers

(b) Implicit solvers

Figure 10: Taylor-Green case solved with higher order implicit and explicit temporal schemes on a skew grid with $Re = \infty$

In figure 10b almost identical results to figure 10a can be seen, showing again that the higher order shemes perform better mostly for the Chorin method. The difference between

Backward Euler and DIRK2 is much greater than the difference between DIRK2 and DIRK3. Finally, the inaccuracy that lead to the Backward Euler Van Kan method to give some slight numerical dissipation, seems to be absent in higher order schemes.

4.2 Temporal consistency - Lid-driven cavity

A self-convergence study was performed on a lid-driven cavity case to study the order of accuracy of a range of temporal schemes. The lid-driven cavity case is a widely used case to assess solver performance. It consists of a two-dimensional square domain with four walls of an arbitrary length L. The top wall, the lid, moves with a velocity of U_L and thereby creates a vortex inside the domain, a developed lid-driven cavity flow can be seen in figure 11. The boundary conditions are no-slip for the velocity and zero-gradient for the pressure. The temporal error of the simulation, taken at the final overesttime step, is given by the difference between the kinetic energy of the solution and the kinetic energy of the reference case, which is run using a very small time step. The relationship between this relative error and the size of the time step results in the order of accuracy of each method. In the used case the length of the domain was chosen as $L = 1$ and the velocity of the lid as $U_L = 1$. The viscosity was set to $\nu = 0.001$ leading to a Reynold's number of $Re = U_L L/\nu = 1000$. The case was run up to $2048 \cdot 10^{-6}$ time units (U_L/L) with time steps $\Delta t = 2^k \cdot 10^{-6}$ with $k = \{0, \ldots, 10\}$. The study was performed on both a 64x64 and a 512x512 regular Cartesian grid. Also, the effects of the Van Kan and Chorin methods for the pressure predictor were analysed. The same time schemes are used for this case as for the Taylor-Green Vortex, with their order of temporal accuracy between brackets: Backward Euler $\mathcal{O}(\Delta t^1)$, DIRK2 $\mathcal{O}(\Delta t^2)$, DIRK3 $\mathcal{O}(\Delta t^3)$, Forward Euler $\mathcal{O}(\Delta t^1)$, RK3 $\mathcal{O}(\Delta t^3)$, RK4 $\mathcal{O}(\Delta t^4)$ [5].

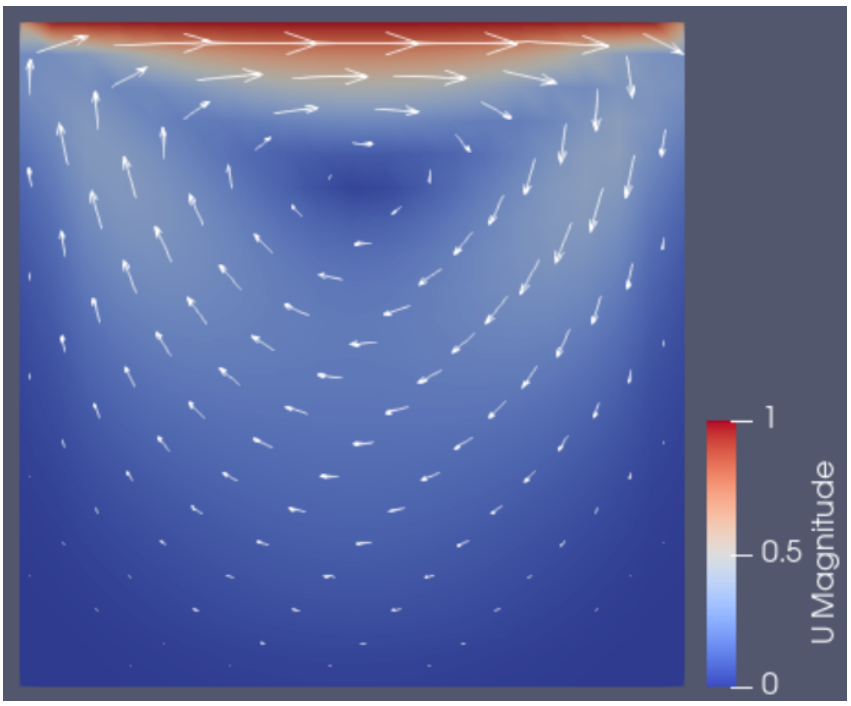
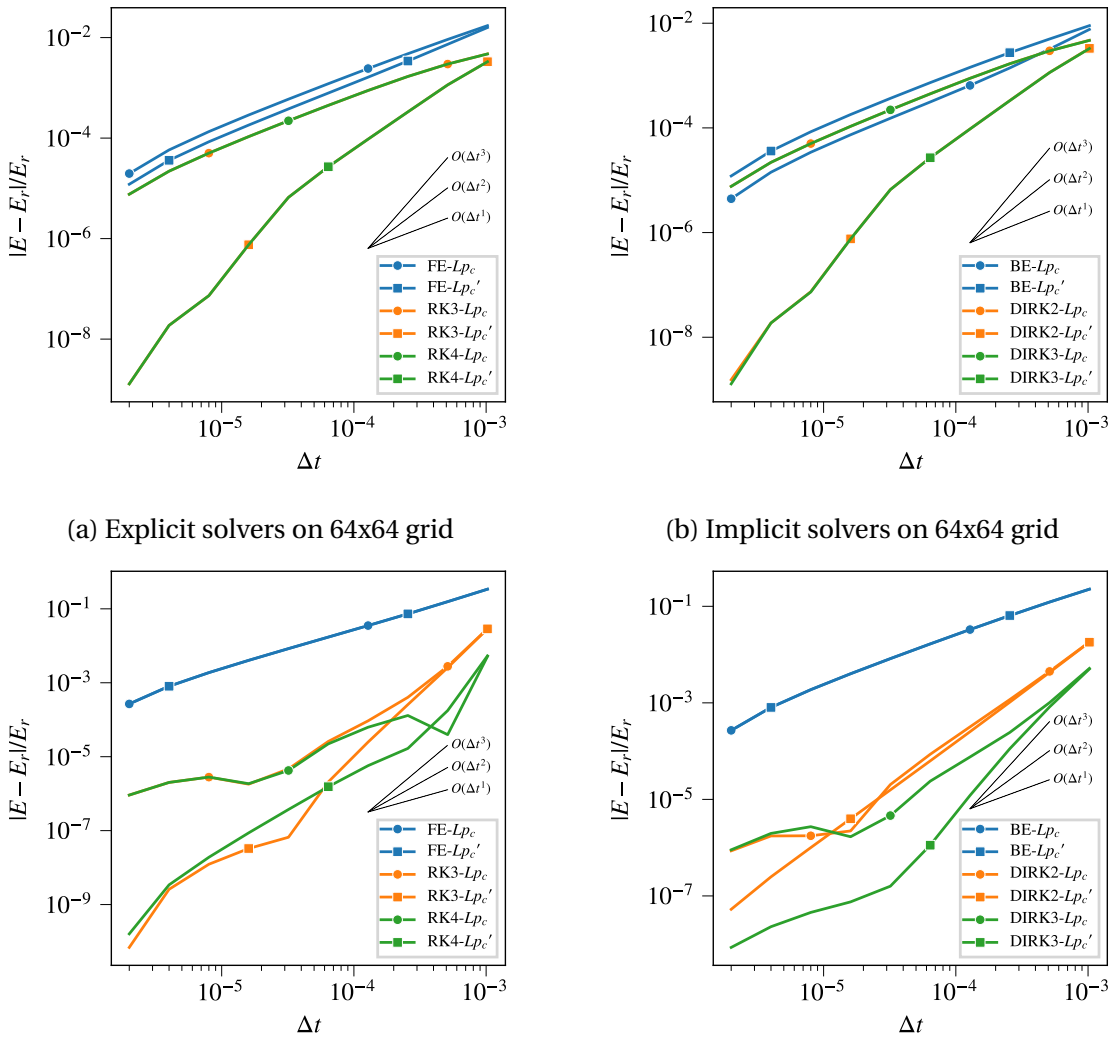


Figure 11: Velocity field of a developed lid-driven cavity flow

Two main causes of errors are expected from this study. Firstly, a temporal discretiza-

tion error is expected, $\mathcal{O}(\Delta t^n)$, with n the temporal order of the applied temporal integration scheme. Secondly, the pressure error, $\mathcal{O}(\Delta t \Delta h^2)$, that results from the non-exact conservation of mass by the cell center velocities [9]. This error is expected to be reduced to $\mathcal{O}(\Delta t^2 \Delta h^2)$ when the Van Kan method is applied, because a part of the total pressure is calculated on a wide stencil reducing the error [3]. Because of the Δh^2 term in the pressure error, the pressure error is expeted to be dominant on the coarser 64x64 grid and the temporal error to be more dominant on the 512x512 grid, especially when using the Chorin method, because the pressure error term is larger and therefore the effect of grid refinement should be more prevalent.

Figure 12a shows the relative error in the kinetic energy of the case for the explicit time integration schemes run on a 64x64 grid, comparing the Chorin and Van Kan method at $Re = 1000$. It can be seen that the results for the Chorin method lead to a first order temporal accuracy, which is to be expected as the pressure term dominates on a coarse grid. It is given by $\mathcal{O}(\Delta t \Delta h^2)$ for the Chorin method. For the Van Kan method a distinction can be seen between the Forward Euler scheme and the higher-order schemes. This is because the pressure term is in this case given by $\mathcal{O}(\Delta t^2 \Delta h^2)$. The Forward Euler result is dominated by its temporal order of $\mathcal{O}(\Delta t)$, whereas the higher order time schemes show a second order temporal accuracy. The results for the implicit schemes lead to the same conclusions, figure 12b, the Chorin method is restricting the solvers to first order because the pressure term is leading on a coarse grid, whereas the Van Kan method raises the accuracy to second order, given that the temporal order of the scheme is higher than first order as well. Schemes of a temporal order of accuracy higher than 2, still only show a second order of accuracy, suggesting that the pressure term is still dominant.



(a) Explicit solvers on 64x64 grid

(b) Implicit solvers on 64x64 grid

(c) Explicit solvers on 512x512 grid

(d) Implicit solvers on 512x512 grid

Figure 12: Temporal consistency in a lid-driven cavity case for a range of explicit and implicit solvers on a 64x64 and a 512x512 Cartesian grid. Reference energy (E_r) is taken as the solution with $\Delta t = 10^{-6}$.

Refining the grid to a 512x512 grid shows that the pressure term becomes less dominant. This can be seen in figures 12c and 12d. Even when using the Chorin method, it can be seen that the higher order temporal schemes, such as RK4 and DIRK3, tilt upwards to second order. While Forward and Backward Euler show their first order temporal accuracy. The fact that the higher order schemes are not fully showing their temporal order as the order of accuracy, suggests that the pressure term still plays a significant role. Finally, using the Van Kan method on a fine grid, the third or higher order schemes can be seen to become even third order of accuracy, whereas Forward and Backward Euler stay first order. However, fourth order of accuracy for RK4 was not fully attained, suggesting that both the pressure and temporal effects are still playing a part to reach a combined temporal order of accuracy.

4.3 Kinetic energy assessment - Channel Flow

As a more complex test case, a turbulent channel flow was examined. The characteristics of the flow were analysed by looking at the budget terms that contribute to the mean kinetic energy (MKE) and the mean turbulent kinetic energy (TKE) of the flow. As a reference, data of Vreman and Kuerten was used [23]. A similar method was used as in [5].

Case set-up

The channel flow case consists of two plates, separated by height $2h$, between which a lengthwise fully-developed turbulent flow moves. Boundaries at the beginning and the end of the channel as well as the front and the back are periodic. The length of the channel was chosen as 4π, the height as 2 and the depth as $4\pi/3$, to resemble the dimensions used by [5] and [23], see figure 13.

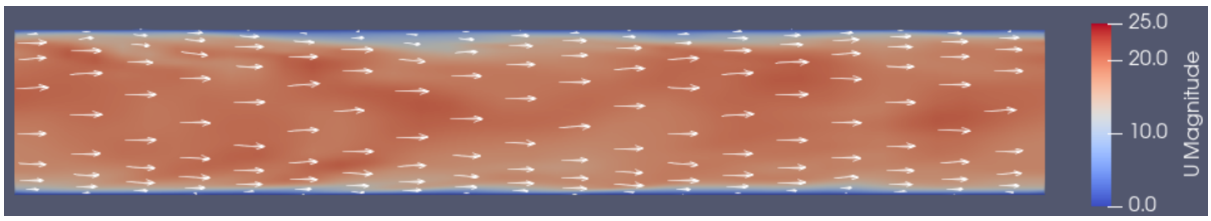


Figure 13: Velocity field in a fully developed channel flow

The flow has a frictional Reynolds number of $Re_\tau = u_\tau h/\nu = 180$. With ν the kinematic viscosity and the frictional velocity, u_τ, given by:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} = \sqrt{\nu\left(\frac{\partial\langle u\rangle}{\partial y}\right)_{wall}} \tag{104}$$

In which τ_w is the wall shear stress and $\langle u\rangle$ is the mean stream-wise velocity. To drive the flow, a fixed pressure gradient is used in the momentum equation. The initial field was established with the method of [24]. The orthogonal 60-grid mesh, as introduced by [5], was used for the present case. In this mesh, cell sizes are represented in wall units, calculated using:

$$\Delta_i^+ = \Delta_i \frac{u_\tau}{\nu}, \quad i = x, y, z \tag{105}$$

with $\Delta_i = x, y, z$ representing the cell sizes in stream-wise, wall normal and span-wise direction. The cell sizes are presented in table 1.

Δx^+	Δy^+_{wall}	Δy^+_{bulk}	Δz^+	N_x	N_y	N_z	N_{total}
60	0.35	10	20	38	108	38	155,952
30	0.35	5	10	75	116	75	652,500

Table 1: grid specifics for the turbulent channel flow at $Re_\tau = 180$ for the 60- and 30-grids. A stretch ratio of 1.05 is used between the cells at the wall and the bulk

The case was run using the RK3 time scheme, as it was found in the previous sections that it gave nearly identical results to RK4 and implicit schemes, while having a significantly lower computational cost.

Quantification methods

To quantify the results in terms of numerical dissipation, the kinetic energy of the flow averaged over time was studied. The averaged kinetic energy can be decomposed into MKE and TKE as follows [25]:

$$\langle KE \rangle = \frac{1}{2}\langle u^2 \rangle = \frac{1}{2}\big\langle \langle u \rangle \langle u \rangle \big\rangle + \big\langle \langle u \rangle u' \big\rangle + \frac{1}{2}\langle u'u' \rangle \tag{106}$$

where the velocity, u, is decomposed in a mean, $\langle u \rangle$, and a fluctuating component, u', following the Reynolds decomposition of instantaneous velocity, $u_i = \langle u_i \rangle + u'_i$. The first term on the RHS corresponds to the MKE, the middle term vanishes due to averaging and the last term corresponds to the TKE.

Quantification of the TKE was based on the turbulent kinetic energy per unit mass, $\langle k \rangle$, given by: $\langle k \rangle = \frac{1}{2}\Sigma_i \langle u'_i u'_i \rangle$, and its individual components $\langle u'u' \rangle$, $\langle v'v' \rangle$ and $\langle w'w' \rangle$. Following [26] and [27], a transport equation for the $\langle u'_i u'_j \rangle$ component of the Reynolds stress tensor for isothermal incompressible flow can be derived as:

$$\frac{D\langle u'_i u'_j \rangle}{Dt} = \frac{\partial \langle u'_i u'_j \rangle}{\partial t} + \langle u_k \rangle \frac{\partial \langle u'_i u'_j \rangle}{\partial x_k} = \langle P^t_{ij} \rangle + \langle \epsilon^t_{ij} \rangle + \langle T^t_{ij} \rangle + \langle \Pi^{s,t}_{ij} \rangle + \langle \Pi^{d,t}_{ij} \rangle + \langle D^t_{ij} \rangle, \tag{107}$$

leading to the following budget terms on the RHS:

Production rate, $$\langle P^t_{ij} \rangle = -\langle u'_j u'_k \rangle \frac{\partial \langle u_i \rangle}{\partial x_k} - \langle u'_i u'_k \rangle \frac{\partial \langle u_j \rangle}{\partial x_k}, \tag{108}$$

dissipation rate, $$\langle \epsilon^t_{ij} \rangle = -2\nu \Big\langle \frac{\partial u'_i}{\partial x_k} \frac{\partial u'_j}{\partial x_k} \Big\rangle, \tag{109}$$

turbulent transport rate, $$\langle T^t_{ij} \rangle = -\frac{\partial}{\partial x_k} \langle u'_k u'_i u'_j \rangle, \tag{110}$$

pressure strain rate, $\langle \Pi_{ij}^{s,t} \rangle = \left\langle \frac{p'}{\rho} \left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right) \right\rangle,$ (111)

pressure diffusion rate, $\langle \Pi_{ij}^{d,t} \rangle = -\frac{1}{\rho} \left(\frac{\partial \langle u_i' p' \rangle}{\partial x_j} + \frac{\partial \langle u_j' p' \rangle}{\partial x_i} \right),$ (112)

and molecular diffusion, $\langle D_{ij}^{t} \rangle = \frac{\partial}{\partial x_k} \nu \frac{\partial}{\partial x_k} \langle u_i' u_j' \rangle.$ (113)

From the continuity constraint, it follows that the pressure strain rate does not change the total amount of mean turbulent kinetic energy, it merely redistributes it in other directions. On a collocated grid arrangement, however, the velocity field is not exactly divergence free, therefore this term should also be assessed. All budget terms together should cancel each other out to be exactly zero, any residue will be due to numerical errors, such as spatial and temporal discretization errors, iteration errors, computational errors in determining the budget terms and finally due to Reynolds-averaging [25]. In summation, this numerical inaccuracy is then given by:

$$\langle P_{\Delta}^{t} \rangle + \langle \epsilon_{\Delta}^{t} \rangle + \langle T_{\Delta}^{t} \rangle + \langle \Pi_{\Delta}^{s,t} \rangle + \langle \Pi_{\Delta}^{d,t} \rangle + \langle D_{\Delta}^{t} \rangle = \Delta_{num}^{t} \tag{114}$$

Similarly, the MKE budget terms can derived from the transport equation for the $\langle u_i u_j \rangle$ terms of the total kinetic energy [28]:

$$\frac{D\langle u_i u_j \rangle}{Dt} = \frac{\partial \langle u_i u_j \rangle}{\partial t} + \langle u_k \rangle \frac{\partial \langle u_i u_j \rangle}{\partial x_k} = \langle P_u^m \rangle + \langle T_u^m \rangle + \langle D_u^m \rangle + \langle V_u \rangle \tag{115}$$

leading to the following budget terms on the RHS:

Production rate, $\langle P_u^m \rangle = -\frac{1}{\rho} \langle u_i \rangle \frac{\partial \langle p \rangle}{\partial x_i},$ (116)

transport rate, $\langle T_u^m \rangle = \frac{\partial}{\partial x_j} \left(-\langle u_i \rangle \langle u_i' u_j' \rangle \right),$ (117)

deformation, $\langle D_u^m \rangle = \langle u_i' u_j' \rangle \frac{\partial \langle u_i \rangle}{\partial x_j},$ (118)

and viscous diffusion, $\langle V_u^m \rangle = \nu \langle u_i \rangle \frac{\partial}{\partial x_j} \left(\frac{\partial \langle u_i \rangle}{\partial x_j} \right).$ (119)

Again, these terms should cancel out and sum to be zero, leading to the numerical error to be given by:

$$\langle P_{\Delta}^{m} \rangle + \langle T_{\Delta}^{m} \rangle + \langle D_{\Delta}^{m} \rangle + \langle V_{\Delta}^{m} \rangle = \Delta_{num}^{m} \tag{120}$$

Post-processing

To calculate the terms involved in the TKE and MKE, 100 flow through times (FTT) of the channel were considered, with $FTT = L/u_{mean}$, ranging between $t = 100$ and $t = 200$, to assure that the flow was fully developped. Every 0.25 temporal unit of the simulation the instantaneous fields were stored, so the required temporal and spatial averaging could be done as part of post-processing, this overesttime step is chosen such that the fields are taken as close to each other as possible without statistical correlation. This value is found by $t^+ = tu_\tau/h = 0.25$ [23]. The field was averaged in the x- and z- direction, because of periodicity, leaving the budgets as a function of channel width, in units of y^+.

It was found that calculating the gradient and Laplacian terms with first order accuracy lead to a significant numerical error. A set of higher order stencils was developped, using the method of Veldman for higher order spatial discretizations on non-uniform grids [16]. With the use of Taylor expansions, methods were developped for stencils with 3, 5, 7 and 9 points. Given below are the resulting schemes to calculate the gradient and the Laplacian at cell 0, using figure 14:

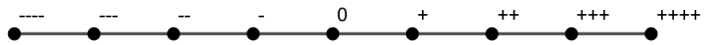


Figure 14: Supportive figure for determining higher order gradient and Laplacian schemes

3 point gradient:

$$\left(\frac{\partial u}{\partial x}\right)_0 = \frac{u_+ - u_-}{h_+ + h_-} \tag{121}$$

5 point gradient:

$$\left(\frac{\partial u}{\partial x}\right)_0 = \frac{-u_{++} + 8u_+ - 8u_- + u_{--}}{8(h_+ + h_-) - (h_{++} + h_{--})} \tag{122}$$

7 point gradient:

$$\left(\frac{\partial u}{\partial x}\right)_0 = \frac{u_{+++} - 9u_{++} + 45u_+ - 45u_- + 9u_{--} - u_{---}}{45(h_+ + h_-) - 9(h_{++} + h_{--}) + (h_{+++} + h_{---})} \tag{123}$$

9 point gradient:

$$\left(\frac{\partial u}{\partial x}\right)_0 = \frac{-3u_{++++} + 32u_{+++} - 168u_{++} + 672u_+ - 672u_- + 168u_{--} - 32u_{---} + 3u_{----}}{672(h_+ + h_-) - 168(h_{++} + h_{--}) + 32(h_{+++} + h_{---}) - 3(h_{++++} + h_{----})} \tag{124}$$

And similarly:
3 point Laplacian:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_0 = \frac{\frac{u_+ - u_0}{h_+} - \frac{u_0 - u_-}{h_-}}{\frac{1}{2}(h_+ + h_-)} \tag{125}$$

5 point Laplacian:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_0 = \frac{8(\frac{u_+ - u_0}{h_+} - \frac{u_0 - u_-}{h_-}) - (\frac{u_{++} - u_0}{h_{++}} - \frac{u_0 - u_{--}}{h_{--}})}{\frac{8}{2}(h_+ + h_-) - \frac{1}{2}(h_{++} + h_{--})} \tag{126}$$

7 point Laplacian:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_0 = \frac{45(\frac{u_+ - u_0}{h_+} - \frac{u_0 - u_-}{h_-}) - 9(\frac{u_{++} - u_0}{h_{++}} - \frac{u_0 - u_{--}}{h_{--}}) + (\frac{u_{+++} - u_0}{h_{+++}} - \frac{u_0 - u_{---}}{h_{---}})}{\frac{45}{2}(h_+ + h_-) - \frac{9}{2}(h_{++} + h_{--}) + \frac{1}{2}(h_{+++} + h_{---})} \tag{127}$$

9 point Laplacian:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_0 = \Big[672\left(\frac{u_+ - u_0}{h_+} - \frac{u_0 - u_-}{h_-}\right) - 168\left(\frac{u_{++} - u_0}{h_{++}} - \frac{u_0 - u_{--}}{h_{--}}\right) + 32\left(\frac{u_{+++} - u_0}{h_{+++}} - \frac{u_0 - u_{---}}{h_{---}}\right) - 3\left(\frac{u_{++++} - u_0}{h_{++++}} - \frac{u_0 - u_{----}}{h_{----}}\right)\Big] \Big/ \left[\frac{672}{2}(h_+ + h_-) - \frac{168}{2}(h_{++} + h_{--}) + \frac{32}{2}(h_{+++} + h_{---}) - \frac{3}{2}(h_{++++} + h_{----})\right] \tag{128}$$

where h_n gives the distance between 0 and point n. Basically, these stencils are all combinations of second order stencils with different widths, with weights following from Taylor expansions for gradients, as if performed on uniform grids. In figure 15, the convergence of the terms for the sum of all the TKE budgets is shown. It is clear that the difference is significant for the sum term. It was found that this difference is caused mainly by the calculation of the dissipation rate, with small differences also seen in calculation of the pressure strain rate. In the rest of the results, the widest (9 point) stencil is used.

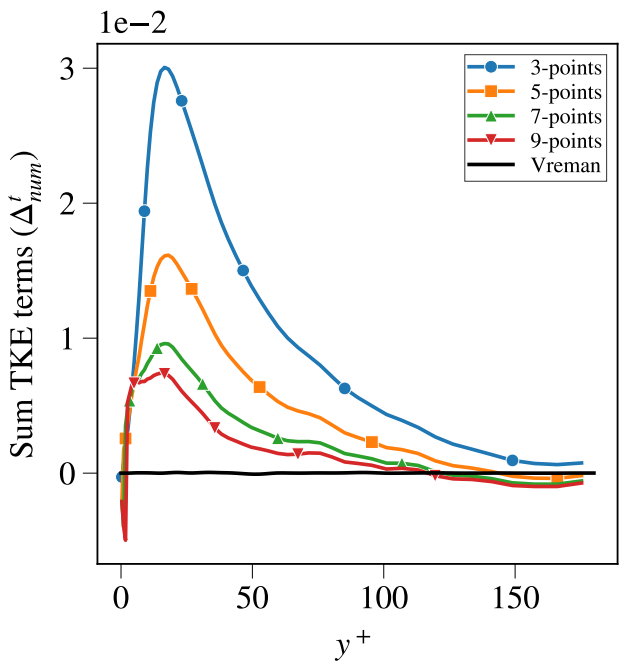


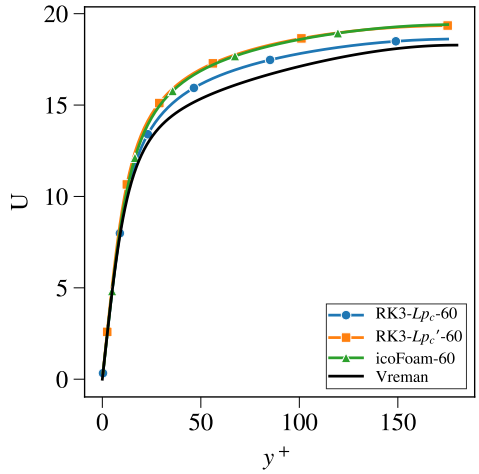
Figure 15: Sum of TKE terms for Lp'_c on the 60-grid, using different width stencils for the gradient and laplacian

Results

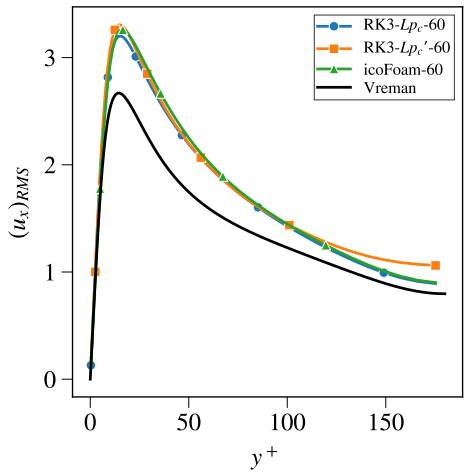
Below, the flow characteristics and budget terms are shown for the channel flow on the 60-grid with Lp_c and Lp'_c methods and OpenFOAM's icoFoam solver. Reference data is shown in black, labeled "Vreman", where pseudo-spectral DNS methods were used, retrieved from [23].

In figure 16 the results for the flow characteristics are shown. It can be seen that the mean velocity, figure 16a, is slightly overestimated by all solvers, with the Lp'_c giving the best estimation. In turn, the velocity fluctuations, figures 16b, 16c and 16d, show an overestimation

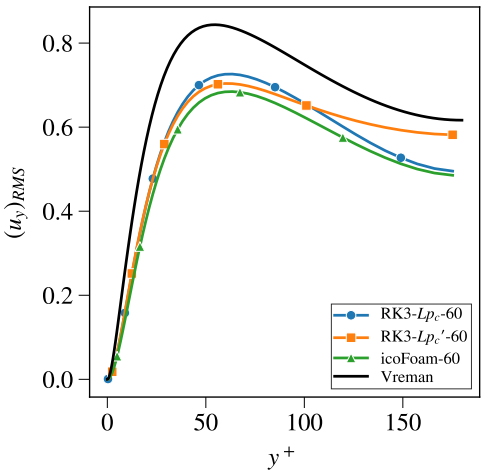
of the root mean square velocity in the x-direction for all solvers, as opposed to underestimation in both the y- and z-directions. Finally, figure 16e, shows that all solvers have an overestimation of the kinetic energy, with Lp_c' having the highest estimate and Lp_c and icoFoam showing similar values. It can also be seen that close to the wall, around $y^+ = 18$, the highest discrepancies in the characteristics can be seen, in the form of peaks, as this is the transition layer between the more viscous flow near the wall and more developed flow in the center of the channel.



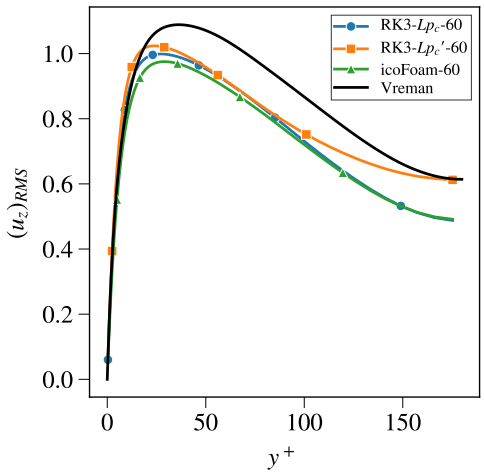
(a) Velocity profile



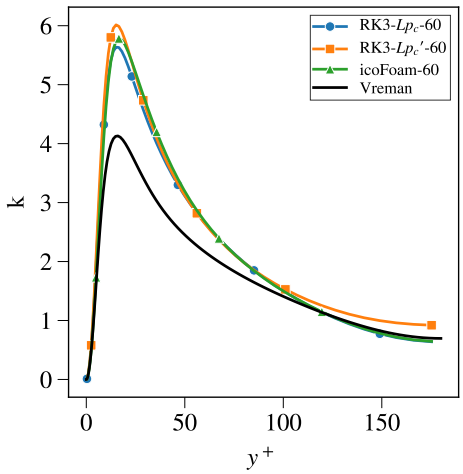
(b) $(\mathbf{u}_x)_{RMS}$



(c) $(\mathbf{u}_y)_{RMS}$



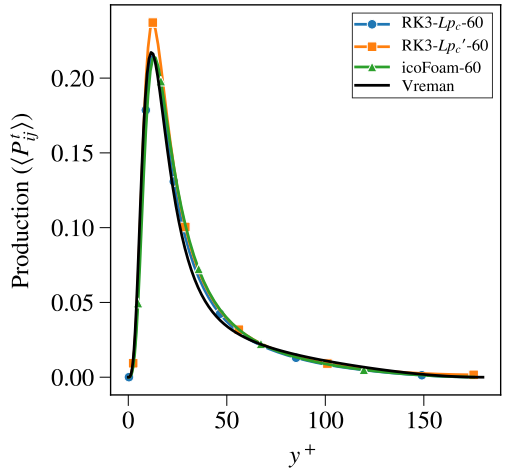
(d) $(\mathbf{u}_z)_{RMS}$



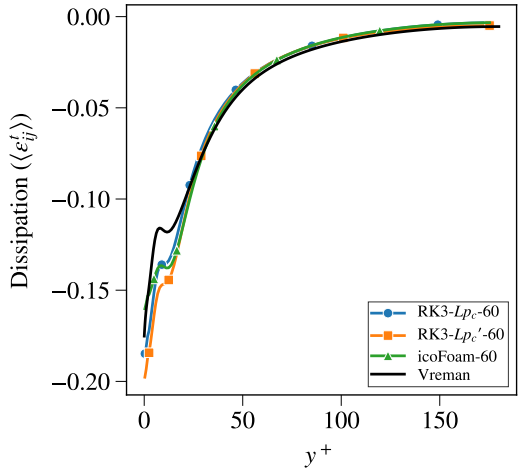
(e) Turbulent kinetic energy

Figure 16: Channel flow characteristics on the 60-grid, showing methods Lp_c, Lp_c' and icoFoam with reference data of Vreman

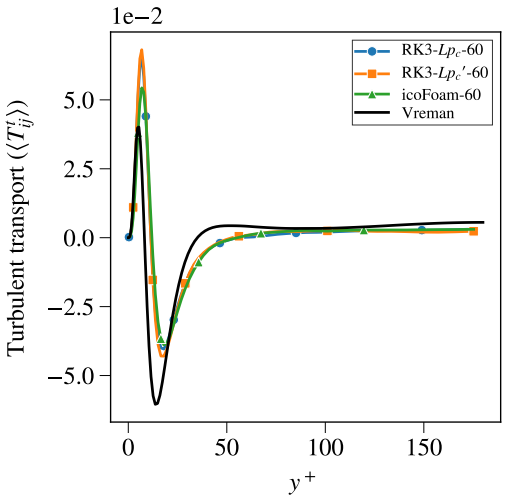
In figure 17 the different TKE budget terms are presented. In general, the results show curves similar to the reference case, except for the pressure strain rate. The production rate, figure 17a, is overestimated near the wall for the Lp'_c method, whereas all solvers underestimate the dissipation rate, figure 17b, with Lp'_c again showing the largest error. Similar results showing slight over- and underestimation can be seen for the turbulent transport, pressure diffusion and molecular diffusion rates, figures 17c, 17e and 17f. The pressure strain rate, figure 17d, is more interesting. As mentioned before, this term should be zero due to the incompressibility constraint. The graphs for all solvers show non-zero results, partially due to incompressibility of the velocity field, but also due to the method of calculating these terms. In post-processing, cell-centered field values are used to calculate budget terms involving transport terms. When using staggered values for these terms the results might give results closer to the reference case. This should be kept in mind when drawing conclusions from these results.



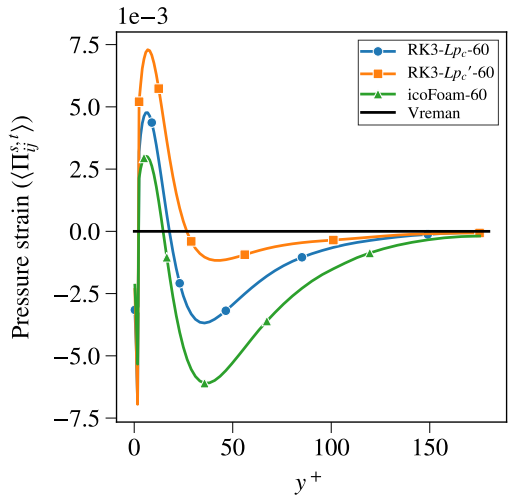
(a) TKE Production rate



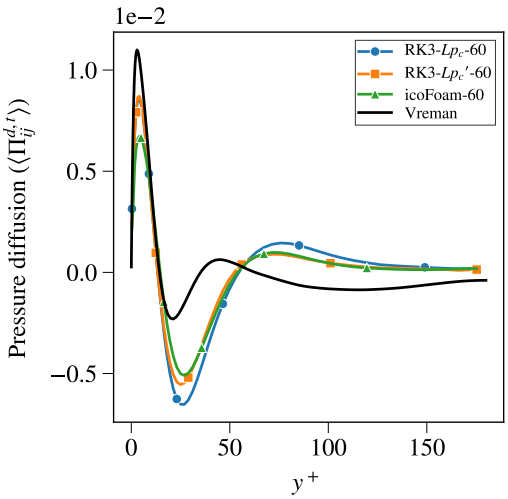
(b) TKE Dissipation rate



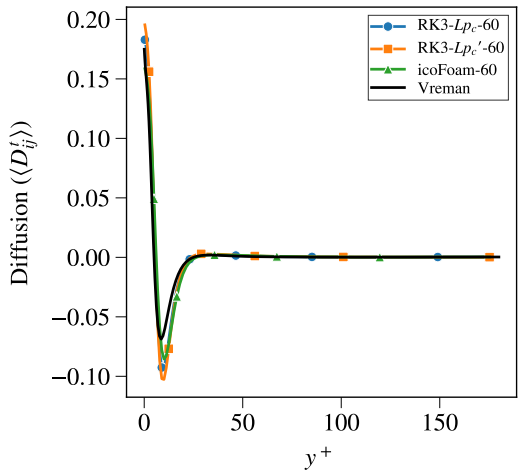
(c) TKE Turbulent transport rate



(d) TKE Pressure strain rate



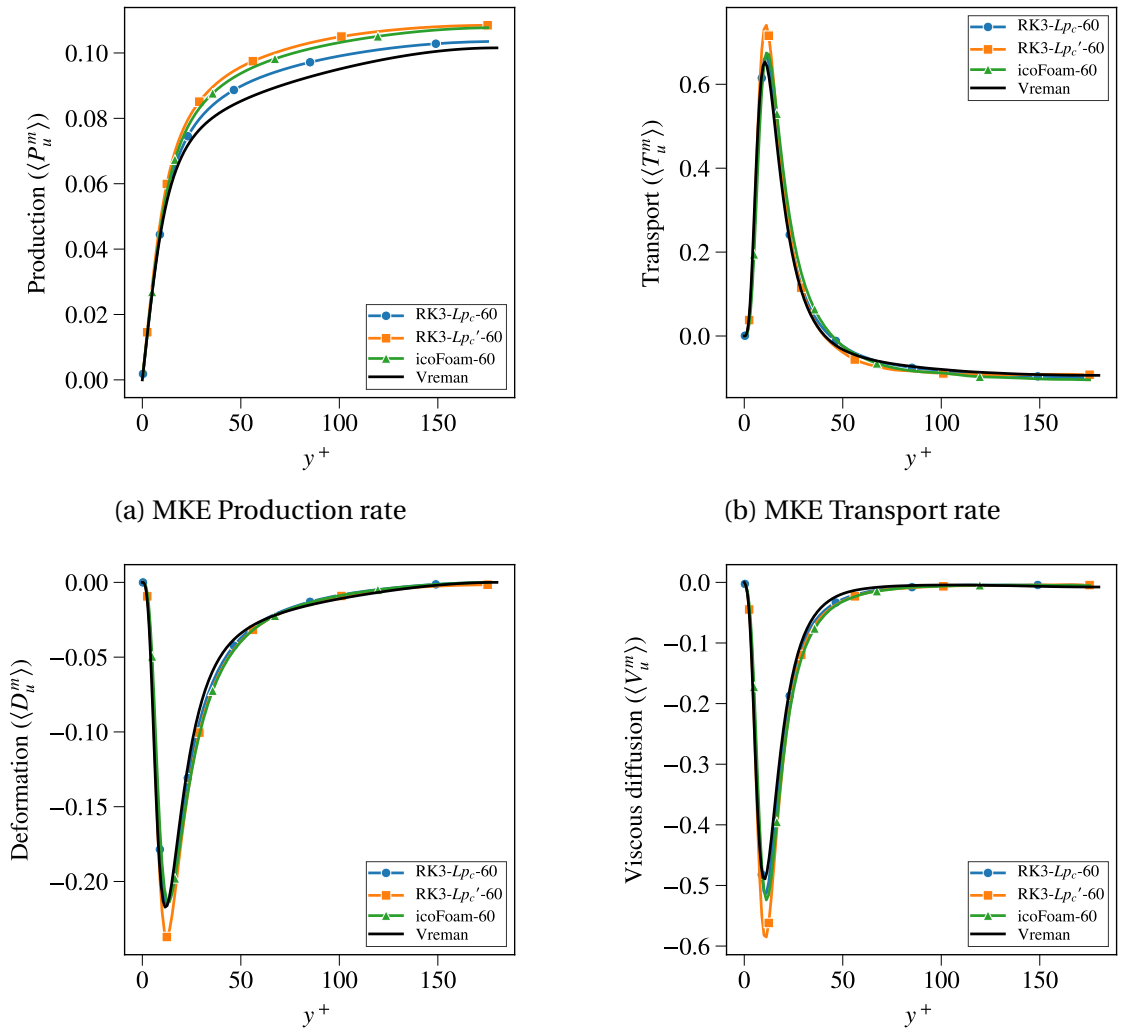
(e) TKE Pressure diffusion rate



(f) TKE Molecular diffusion

Figure 17: TKE budget terms on the 60-grid, showing methods Lp_c, Lp_c' and icoFoam with reference data of Vreman

In figure 18 the MKE budget terms are shown. The production rate, 18a shows that all solvers overestimate the production, especially in the center of the channel. Transport, deformation and viscous diffusion rates, figures 18b, 18c and 18d, show very good agreement for all solvers, although the Lp_c' method shows larger peaks in the transition layer for all budget terms.



(a) MKE Production rate

(b) MKE Transport rate

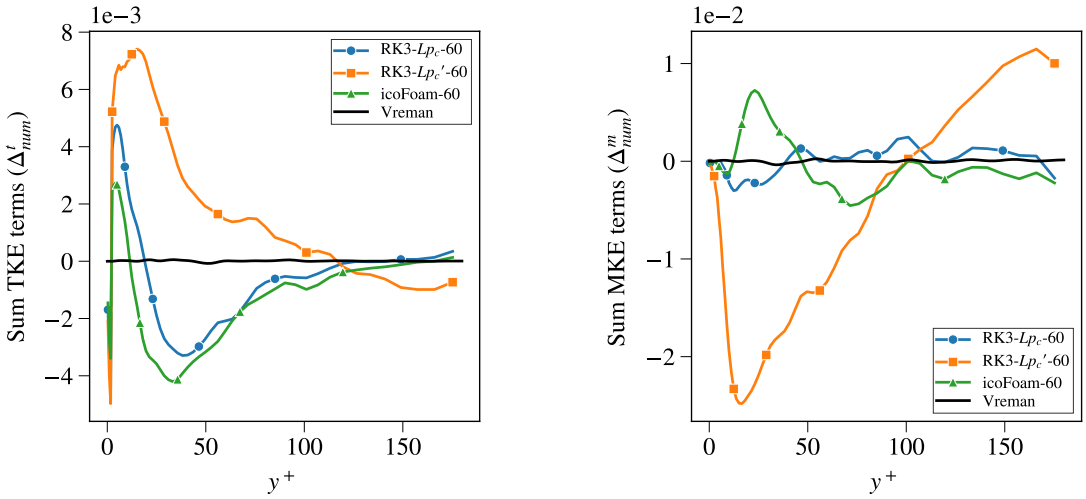
(c) MKE Deformation

(d) MKE Viscous diffusion

Figure 18: MKE budget terms on the 60-grid, showing methods Lp_c, Lp_c' and icoFoam with reference data of Vreman

Sums of all budget terms in TKE and MKE are shown in figure 19. All summing results are very close to zero, in the order of 10^{-2}, which means the budget terms overall cancel each other out to a good degree. However, when compared to the reference case, the inaccuracies are still quite significant. The Lp_c method and icoFoam show similar results close to zero, whereas the Lp_c' method shows a large peak in the TKE sum in the transition area, figure 19a, and a large valley in the same are for the MKE sum, figure 19b. However, when the sums are plotted together with the seperate budget terms, figure 20, a better indication of the relative inaccuracy can be obtained. The order of inaccuracy of the sum of the terms is almost non-

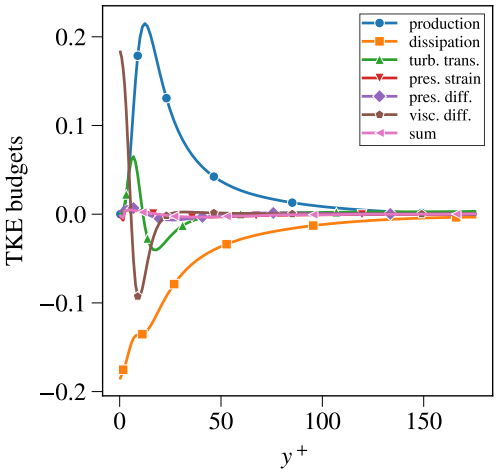
visible compared to the order of the separate terms. This is to show that slight errors in individual terms can heavily affect the sum. These slight errors include, for instance, the order of accuracy with which the gradients and Laplacians are calculated or the choice of method to calculate transport terms, as discussed before.



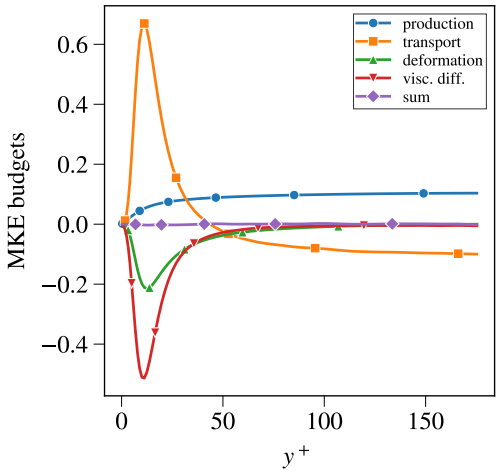
(a) TKE Sum of budget terms

(b) MKE Sum of budget terms

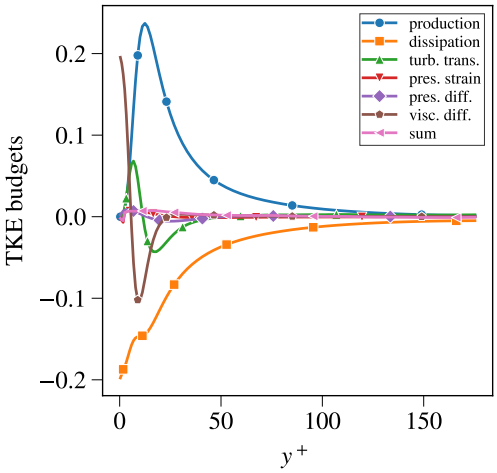
Figure 19: Sum of budget terms for TKE and MKE on the 60-grid, showing methods Lp_c, Lp'_c and icoFoam with reference data of Vreman



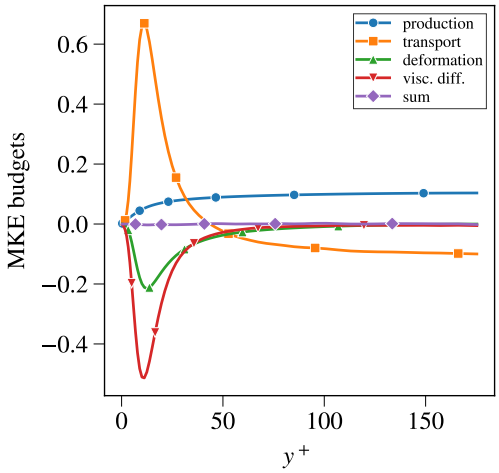
(a) TKE budget terms for Lp_c



(b) MKE budget terms for Lp_c



(c) TKE budget terms for Lp'_c



(d) MKE budget terms for Lp'_c

Figure 20: Seperate TKE and MKE budget terms compared for Lp'_c and Lp_c

To explain the reason why the Lp'_c method performed relatively poor, a qualitative look was taken at the pressure fields in the xz-plane in the transition area at $y^+ = 18$, using Paraview. The results are shown in figure 21. When looking at the pressure field for the Lp_c method, some checkerboarding can be seen in the form of alternating colours. These patterns are not present in the Lp_c case. To overcome the checkerboarding, grid refinement to the 30-grid was applied. Qualitative results for the pressure fields taken at the same height can be seen at the bottom of figure 21, where some checkerboarding can still be seen in the Lp'_c method, however in far less extent and in fewer areas of the plane.

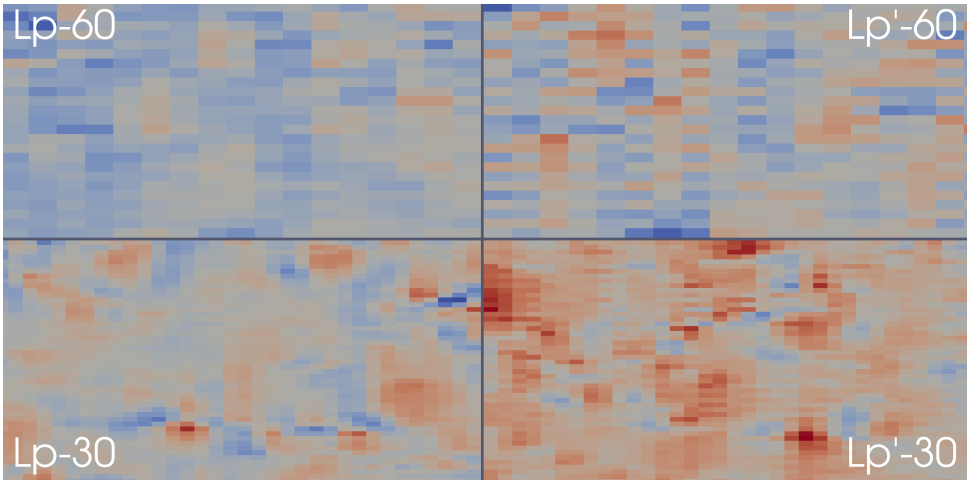
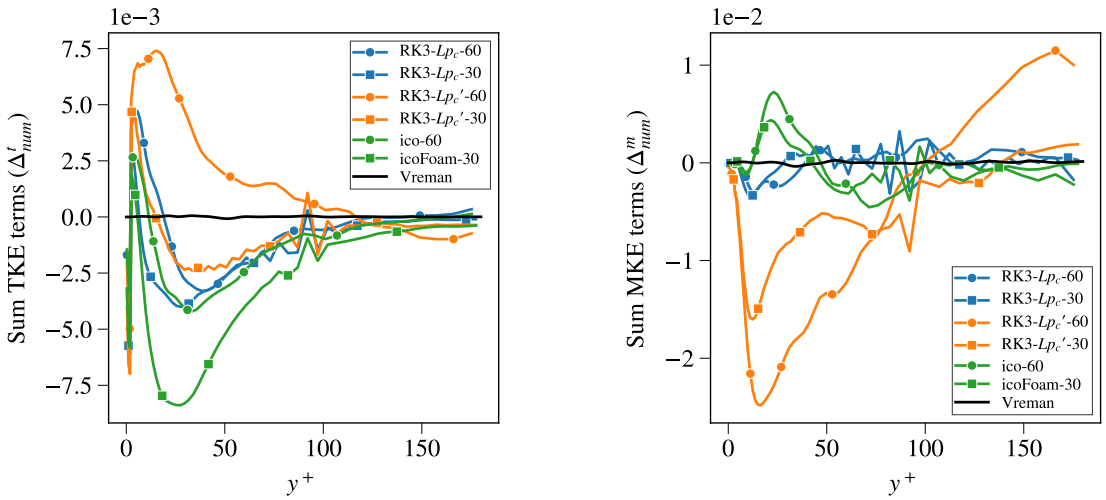


Figure 21: Slices of the channel flow cases at y=0.1 in the xz-plane, qualitatively showing pressure difference per cell. Checkerboarding can be seen in the Lp_c' case on the 60-grid in the z-direction

The budget analyses were again performed for the solvers on the 30-grid, the results for the sum of all terms are shown in figure 22. It can immediately be seen that the Lp_c' method now shows similar results to the Lp_c method for the TKE sum, whereas the icoFoam solver appears to move further away from zero, figure 22a. On the MKE side, figure 22b, it can be seen that the Lp_c' method still shows the least accurate results, however it does show improvement, whereas the other solvers stay similar to their results on the 60-grid.



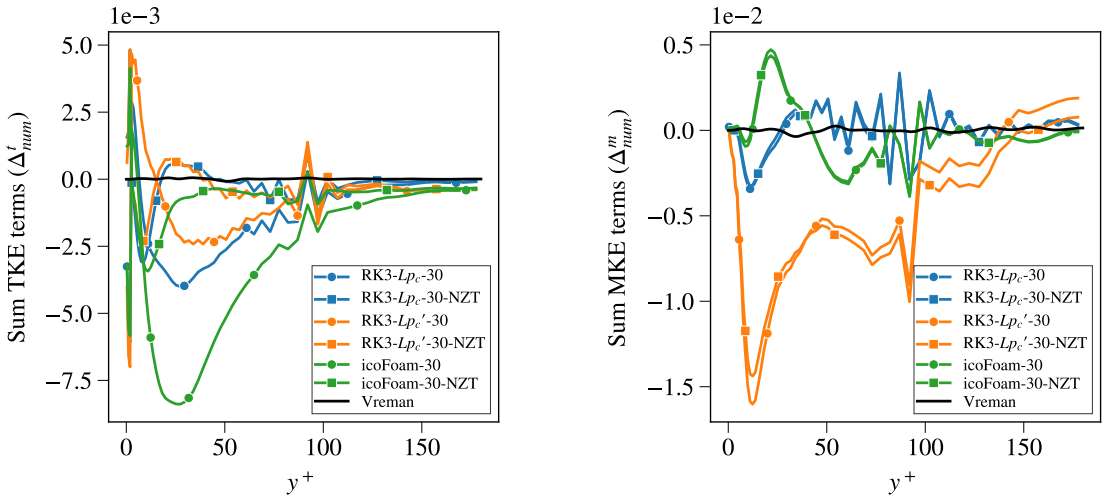
(a) TKE Sum of budget terms

(b) MKE Sum of budget terms

Figure 22: Comparison of sum of budget terms for TKE and MKE on the 60-grid and the 30-grid, showing methods Lp_c' and Lp_c with reference data of Vreman

Finally, a closer look was taken into the method of calculating the budget terms. In the definitions of the budget terms, equations 108-113 and 116-119, many iterations of the individual terms, such as $\langle u_x' \rangle$, should in principle be zero due to averaging, although in practice do not have to be exactly zero in the channel flow results. To see the effect of these terms,

post-processing was performed again, excluding these zero terms and only keeping the non-zero terms. Results for this method can be seen in figure 23, where the suffix "NZT" denotes post-processing with only the mathematical non-zero terms. It can be seen clearly that the TKE sums, figure 23a, move even closer to zero, mainly due to the pressure strain term dropping out, as it should mathematically be exactly zero. On the MKE side, 23b, only slight changes can be seen. The large influence of many post-processing choices that have to be made on the results of post-processing, raises the question of whether the results are depicting numerical artefacts of post-processing or true results from which hard conclusions can be drawn. This question becomes even more important when noting that the budget terms are already very close to zero and have entered the realm where statistical errors also start playing a larger role again. Although this section does show that the Lp_c method performs similar to icoFoam and the Lp_c' is hindered by instabilities due to checkerboarding, an important take-away should also be that the quantification methods of kinetic energy in the turbulent channel flow are of high significance and should be considered thoroughly before drawing hard conclusions from the results they produce.



(a) TKE Sum of budget terms

(b) MKE Sum of budget terms

Figure 23: Comparison of sum of budget terms for TKE and MKE summing all terms or only non-zero terms, on the 30-grid, showing methods Lp_c, Lp_c' and icoFoam with reference data of Vreman

5 Conclusions

The general formulation of the symmetry preserving method for collocated unstructured grids was successfully implemented into OpenFOAM. When comparing the solver to OpenFOAM's standard solver, icoFoam, in a Taylor-Green vortex case, the solver shows significantly less numerical dissipation and therefore higher accuracy. The choice of the pressure prediction method was shown to be the most important factor in this method, showing that the Van Kan method is much less dissipative than the Chorin method. Choices in spatial discritisation showed that the symmetry preserving method mostly favours stability in this case, whereas the linear interpolation method shows slightly higher accuracy due to better estimation of the diffusive term. The symmetry preserving method showed slight underestimation of the diffusive term. It was shown by improving the order of accuracy of the diffusive term that this can potentially cancel out dissipation caused by the pressure prediction method, which should be kept in mind when assessing evolution of kinetic energy. On very distorted grids with no diffusion, some instability was still seen across all symmetry preserving methods. The temporal consistency study in the lid-driven cavity flow showed that the pressure error term increased from $\mathcal{O}(\Delta t \Delta h^2)$ to $\mathcal{O}(\Delta t^2 \Delta h^2)$ by applying the Van Kan method for the pressure prediction. In this case, grid refinement showed that the pressure error becomes less dominant on fine grids, where a mixture between temporal and pressure error was seen. Finally, the turbulent channel flow showed that, in more realistic test cases, checkerboarding instabilities start hindering the symmetry preserving method on coarse grids, where a lower order pressure prediction was enough to avoid the spurious modes. By assessing kinetic energy budget terms, it was shown that the solver is in agreement with reference solutions for this case. However, it was found that these results are highly affected by choices in post-processing methods, including calculation of gradients and Laplacians and choice of calculating budget terms with collocated or staggered variables. In general, the symmetry preserving method has shown great value in academic test cases, due to its mathematically sound fundamentals. But its low-dissipative characteristics leading to its high accuracy inevitably lead to some instability in more realistic cases, where filtering of spurious modes becomes a necessity.

6 Future recommendations

For future work, some strategies to avoid checkerboarding could be considered. First of all, as already shortly discussed in this work, a symmetry preserving linear filter can be further developed to damp the spurious modes in the convective term, without introducing any non-physical dissipation. Another approach would be to modify the pseudo projection of the prediction velocity to remove the spurious modes that lie on its kernel, of which examples can be found in [29] and [30]. In this work, the Van Kan method and Chorin method were tested for the pressure prediction, showing two opposites in terms of accuracy and stability. Other definitions for the pressure predictor could be examined, to find a good balance between these two factors. When stability is warranted by removing spurious modes, even higher order predictors could potentially be of interest. Concerning the spatial discretizations of the symmetry preserving method, midpoint interpolation was found based on symmetry requirements for the convective term. However, no such requirements are strict for other interpolations in the scheme. Therefore other interpolation methods should be examined and tested for their accuracy and stability. Finally, the methods for analyzing kinetic energy in the turbulent channel flow should be further developed. Order of accuracy in calculation of the involved terms, as well as the choice to use collocated or interpolated staggered variables should be reviewed for each term, before more solid conclusions can be drawn from this method.

References

[1] R. W. Verstappen and A. E. Veldman, "Symmetry-preserving discretization of turbulent flow," *Journal of Computational Physics*, vol. 187, pp. 343–368, 5 2003.

[2] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W. Verstappen, "Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids," *Journal of Computational Physics*, vol. 258, pp. 246–267, 2 2014.

[3] J. Van Kan, "A second-order accurate pressure-correction scheme for viscous incompressible," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 870–891, 1986.

[4] A. J. Chorin and J. E. Marsden, *A Mathematical Introduction to Fluid Mechanics-Springer*. Springer-Verlag Berlin Heidelberg, 3 ed., 1993.

[5] E. M. J. Komen, E. M. A. Frederix, T. H. J. Coppen, V. D'alessandro, and J. G. M. Kuerten, "Analysis of the numerical dissipation rate of different Runge-Kutta and velocity interpolation methods in an unstructured collocated finite volume method in OpenFOAM," *Computer Physics Comunications*, 2020.

[6] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*. Prentice Hall, 1st ed., 1996.

[7] J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*. Spring, 3rd ed., 2002.

[8] C. M. Rhie and W. L. Chow, "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA Journal*, vol. 21, no. 11, pp. 1525–1532, 1983.

[9] F. N. Felten and T. S. Lund, "Kinetic energy conservation issues associated with the collocated mesh scheme for incompressible flow," *Journal of Computational Physics*, vol. 215, pp. 465–484, 7 2006.

[10] Y. Morinishi, T. S. Lund, O. V. Vasilyev, and P. Moin, "Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow," *Journal of Computational Physics*, 1998.

[11] N. N. Yanenko, *The Method of Fractional Steps*. Springer-Verlag Berlin Heidelberg, 1971.

[12] J. Blair Perot, "An analysis of the fractional step method," *Journal of Computational Physics*, vol. 108, no. 1, pp. 51–58, 1993.

[13] R. Verstappen, "On restraining the production of small scales of motion in a turbulent channel flow," *Computers and Fluids*, vol. 37, no. 7, pp. 887–897, 2008.

[14] F. Trias and R. Verstappen, "On the construction of discrete filters for symmetry-preserving regularization models," *Computers & Fluids*, vol. 40, pp. 139–148, 1 2011.

[15] A. Báez Vidal, O. Lehmkuhl, F. X. Trias, and C. D. Pérez-Segarra, "On the properties of discrete spatial filters for CFD," *Journal of Computational Physics*, vol. 326, pp. 474–498, 2016.

[16] A. E. P. Veldman, "Computational Fluid Dynamics - Lecture notes in applied mathematics," 2012.

[17] R. Issa, "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of Computational Physics*, vol. 62, pp. 40–65, 1 1986.

[18] The OpenFOAM Foundation, "OpenFOAM v7 User Guide," 2017.

[19] J. C. Butcher, "On Runge-Kutta Processes of High Order," *Journal of the Australian Mathematical Society*, vol. 4, no. 2, pp. 179–194, 1964.

[20] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I.* Springer, 3 ed., 2008.

[21] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, "Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations," *Applied Numerical Mathematics*, vol. 25, pp. 151–167, 1997.

[22] V. Vuorinen, J. P. Keskinen, C. Duwig, and B. J. Boersma, "On the implementation of low-dissipative Runge-Kutta projection methods for time dependent flows using OpenFOAM®," *Computers and Fluids*, vol. 93, pp. 153–163, 2014.

[23] A. W. Vreman and J. G. Kuerten, "Comparison of direct numerical simulation databases of turbulent channel flow at reτ = 180," *Physics of Fluids*, vol. 26, 1 2014.

[24] E. De Villiers, *The Potential of Large Eddy Simulation for the Modelling of Wall Bounded Flows.* PhD thesis, Imperial College of Science, Technology and Medicine, 2006.

[25] E. Komen, L. Camilo, A. Shams, B. Geurts, and B. Koren, "A quantification method for numerical dissipation in quasi-DNS and under-resolved DNS, and effects of numerical dissipation in quasi-DNS and under-resolved DNS of turbulent channel flows," *Journal of Computational Physics*, vol. 345, pp. 565–595, 9 2017.

[26] P. Durbin and B. Reif, *Statistical theory and modeling for turbulent flows.* Wiley & Sons, Ltd, 2 ed., 2011.

[27] J. Jiménez and S. Hoyas, "Turbulent fluctuations above the buffer layer of wall-bounded flows," *Journal of Fluid Mechanics*, vol. 611, pp. 215–236, 2008.

[28] F. T. M. Nieuwstadt, B. J. Boersma, and J. Westerweel, *Turbulence Introduction to Theory and Applications of Turbulent Flows.* Springer, 1998.

[29] L. DAVIDSON, "a Pressure Correction Method for Unstructured Meshes With Arbitrary Control Volumes," *International Journal for Numerical Methods in Fluids*, vol. 22, no. 4, pp. 265–281, 1996.

[30] K. Mahesh, G. Constantinescu, and P. Moin, "A numerical method for large-eddy simulation in complex geometries," *Journal of Computational Physics*, vol. 197, no. 1, pp. 215–240, 2004.

[31] F. X. Trias, A. Gorobets, A. Oliva, and C. D. Pérez-Segarra, "DNS and regularization modeling of a turbulent differentially heated cavity of aspect ratio 5," *International Journal of Heat and Mass Transfer*, vol. 57, no. 1, pp. 171–182, 2013.

Appendices

A Error term in pressure-velocity coupling due to back-and-forth interpolation

One of the most important relations that have to be defined is the relation between staggered and collocated variables, given by $\Gamma_{s\to c}$ and $\Gamma_{c\to s}$. As noted in section 2.4, interpolating from cell to face and back only approximately returns the original value:

$$\mathbf{u}_c \approx \Gamma_{s\to c}\Gamma_{c\to s}\mathbf{u}_c, \qquad \Gamma_{s\to c}\Gamma_{c\to s} \approx I \tag{129}$$

This can be shown most intuitively on a simple one-dimensional grid with three uniform cells and periodic boundaries, as seen in figure 24.

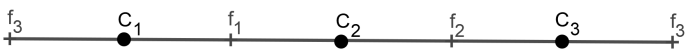


Figure 24: Simple one-dimensional structured grid with periodic boundaries

Since Ω_s and Ω_c are both a simple matrix given by: $\Omega_s = \Omega_c = \overrightarrow{C_1C_2} \otimes I_3$, equation 25 reduces to: $\Gamma_{s\to c} = \Gamma^T_{c\to s}$, when midpoint interpolation is chosen, with equal weights of $\frac{1}{2}$. Therefore, $\Gamma_{c\to s}$ and $\Gamma_{s\to c} = \Gamma^T_{c\to s}$ are given by:

$$\Gamma_{c\to s} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}, \qquad \Gamma_{s\to c} = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{130}$$

Given $\mathbf{u}_{c_1}$, $\mathbf{u}_{c_2}$ and $\mathbf{u}_{c_3}$ the following equations hold for interpolating $\mathbf{u}_{c_2}$ to faces and then back to the cell:

$$\mathbf{u}_{f_1,c\to s} = \frac{\mathbf{u}_{c_1} + \mathbf{u}_{c_2}}{2}, \qquad \mathbf{u}_{f_2,c\to s} = \frac{\mathbf{u}_{c_2} + \mathbf{u}_{c_3}}{2} \tag{131}$$

$$\mathbf{u}_{c_2,c\to s\to c} = \frac{\mathbf{u}_{f_1,c\to s} + \mathbf{u}_{f_2,c\to s}}{2} = \frac{\mathbf{u}_{c_1} + 2\mathbf{u}_{c_2} + \mathbf{u}_{c_3}}{4} \approx \mathbf{u}_{c_2} \tag{132}$$

Which is in line with equation 129, given that neighbouring cells have comparable velocities. When expanding to a three-dimensional unstructured grid, this approximation will remain and interpolating back and forth will always result in a small error, because it includes information from its neighbouring cells. This error leads to a difference in the terms for $\mathbf{u}_s^{n+1}$ and $\Gamma_{c\to s}\mathbf{u}_c^{n+1}$, which can be calculated using their definitions. Starting from equation 29 the error term can be derived:

$$\mathbf{u}_c^{n+1} = \mathbf{u}_c^p - \Gamma_{s\to c}G\tilde{\mathbf{p}}'_c \tag{133}$$

Using $\mathbf{u}_c^p = \mathbf{u}_c^n + \Delta t R(\mathbf{u}_c^n) - G_c\tilde{\mathbf{p}}_c^n$ and $\Gamma_{s\to c}G = G_c$ which gives:

$$\mathbf{u}_c^{n+1} + G_c\tilde{\mathbf{p}}'_c = \mathbf{u}_c^n + \Delta t R(\mathbf{u}_c^n) - G_c\tilde{\mathbf{p}}_c^n \tag{134}$$

Using $\mathbf{u}_s^{n+1} = \mathbf{u}_s^p - G\tilde{\mathbf{p}}_c'$ and $\mathbf{u}_s^p = \Gamma_{c\to s}\mathbf{u}_c^p$ to find:

$$\mathbf{u}_s^{n+1} = \Gamma_{c\to s}\left(\mathbf{u}_c^n + \Delta t R(\mathbf{u}_c^n) - G_c\tilde{\mathbf{p}}_c^n\right) - G\tilde{\mathbf{p}}_c' \tag{135}$$

Interpolating equation 134 using $\Gamma_{c\to s}$ and combining it with 135, gives:

$$\mathbf{u}_s^{n+1} = \Gamma_{c\to s}\mathbf{u}_c^{n+1} + \Gamma_{c\to s}G_c\tilde{\mathbf{p}}_c' - G\tilde{\mathbf{p}}_c' \tag{136}$$

The error term given by the difference in $\mathbf{u}_s^{n+1}$ and $\Gamma_{c\to s}\mathbf{u}_c^{n+1}$ is then given by:

$$\mathbf{u}_s^{n+1} - \Gamma_{c\to s}\mathbf{u}_c^{n+1} = (\Gamma_{c\to s}G_c - G)\tilde{\mathbf{p}}_c' = (\Gamma_{c\to s}\Gamma_{s\to c} - I)G\tilde{\mathbf{p}}_c' \tag{137}$$

This means that the cell-centered velocity field, $\mathbf{u}_c^{n+1}$ is not exactly divergence free, as there is a small difference with $\mathbf{u}_s^{n+1}$, which is divergence free.

B Symmetric linear filter

The convective term in the Navier-Stokes equation is a source of non-physical spurious modes. To stop the production of these modes the non-linearity can be approximated by regularization methods. These methods alter the convective term by restraining production of motion at the smaller scales. One method of regularization is the C_4 method, where the convective term is replaced by a fourth order accurate approximation [13]:

$$C_4(\mathbf{u}_s, \mathbf{u}_c) = C(\overline{\mathbf{u}_s}^p)\overline{\mathbf{u}_c} + F(C(\overline{\mathbf{u}_s}^p)\mathbf{u}_c' + C((\mathbf{u}_s^p)')\overline{\mathbf{u}_c}). \tag{138}$$

In which $\overline{\mathbf{u}_c} = F\mathbf{u}_c$ and $\mathbf{u}_c' = \mathbf{u}_c - \overline{\mathbf{u}_c}$, with F, the filter operation. The idea behind this filter is that it is also skew-symmetric, therefore the kinetic energy of this term is also conserved. Construction and application of this filter were accurately described by two algorithms presented by Trias et al. [31]. In this paper, the filter, F, is constructed in the first algorithm, after which $C_4(\mathbf{u}_s, \mathbf{u}_c)$ is calculated in the second one. Just like operator D, F is a block diagonal matrix and is constructed with square block filters, $F_\#$:

$$F = I_2 \otimes F_\# \tag{139}$$

In which the blocks are given by:

$$F_\# = I + \sum_{a=1}^{A} (\tilde{D}_a)^a \tag{140}$$

where Trias et al. found that $A = 2$ is sufficient to properly filter the convective term [31]. $\tilde{D}_a$ is given by:

$$\tilde{D}_a = -\Omega_c^{-1} M (\Lambda_a)^{1/a} J^2 \Omega_s^{-1} M^T \tag{141}$$

in which J and Λ_a are diagonal matrices in $\mathbb{R}^{m\times m}$ with entries $[J]_{f,f} = \delta n_f$ and $[\Lambda_a]_{f,f} = (d_a)_f$. It is evident from equation 141 that the symmetry of D is preserved in $\tilde{D}_a$, since both Λ_a and J are diagonal matrices. Values for d_a are in turn given by:

$$\begin{aligned} d_1 &= -\frac{\hat{G_{k_c}} - 1}{2(2\hat{G}_{k_c} + 1)}, \qquad d_2 = \frac{2\hat{G}_{k_c}^2 - 3\hat{G}_{k_c} + 1}{16(2\hat{G}_{k_c} + 1)}, \qquad \text{if } 0 \le \hat{G}_{k_c} < 1/2 \\ d_1 &= \frac{1}{4} - \frac{\hat{G}_{k_c}}{4}, \qquad d_2 = 0, \qquad \text{if } 1/2 \le \hat{G}_{k_c} \le 1 \end{aligned} \tag{142}$$

$\hat{G}_{k_c}$, finally, is the value of the transfer function at the smallest grid scale. In [31] this value is updated through time, however, [2] states that a sufficive choice of the filter length results in $\hat{G}_{k_c} = 0.1$. This value is assumed to be appropriate in this study as well, leading to $d_1 = \frac{3}{8}$ and $d_2 = \frac{3}{80}$.
The full expression for $F_\#$ can be written as:

$$F_\# = I + \frac{3}{8}\Omega_c^{-1} M J^2 G + \frac{3}{80}\Omega_c^{-1} M J^2 G \Omega_c^{-1} M J^2 G \tag{143}$$

The next step is to calculate the filtered convective term, C_4. To do so we follow the remaining steps of the second algorithm of [31], given by:

1. copmute $\overline{\mathbf{u}_c}$ and its residual: $\overline{\mathbf{u}_c} = F\mathbf{u}_c$ and $\mathbf{u}_c' = \mathbf{u}_c - \overline{\mathbf{u}_c}$.
2. Solve the following Poisson equation: $-M\Omega_s^{-1}M^T q_c = M\Gamma_{c\to s}\overline{\mathbf{u}_c}$.
3. Compute the projected (divergence-free, $M\overline{\mathbf{u}_s}^p = 0_c$) velocity field, $\overline{\mathbf{u}_s}^p = \Gamma_{c\to s}\overline{\mathbf{u}_c} + \Omega_s^{-1}M^T q_c$ and its residual $(\mathbf{u}_s^p)' = \Gamma_{c\to s}\mathbf{u}_c - \overline{\mathbf{u}_s}^p$.
4. Copmute C_4 in a discrete sense:

$$C_4(\Gamma_{c\to s}\mathbf{u}_c, \mathbf{u}_c) = C(\overline{\mathbf{u}_s}^p)\overline{\mathbf{u}_c} + F(C(\overline{\mathbf{u}_s}^p)\mathbf{u}_c' + C((\mathbf{u}_s^p)')\overline{\mathbf{u}_c}). \tag{144}$$

The computational costs of this filter are considerable, as F has to be calculated for both steps two and five. Moreover, in step 4, the convective term is calculated three times in stead of once. Finally, the most expensive step is step three, in which an additional Poisson equation has to be solved. All in all, this filtering method for regularization has been found to be twice more expensive [31].

C Total volume for staggered and collocated discretization

When interpolating a system from collocated to staggered notation or vice versa, physical attributes of the system should not be altered. One of the requirements for this is that the total volume of the system remains the same in both notations, i.e. the total volumes represented by Ω_c and Ω_s. When looking at the definition of the staggered to collocated shift operator, $\Gamma_{s\to c}$, equation 25, this makes sense intuitively. Since both matrices for volume are diagonal, their traces should be equal in any number of dimensions:

$$Tr(\Omega_c) = Tr(\Omega_s) \tag{145}$$

This can be shown by looking at an adaptation of figure 2, in which the staggered volume of one of the faces is shown, figure 25. The volume of $(\Omega_s)_{f,f}$ will be split in two parts, a part overlapping with the volume of cell i and a part overlapping with the volume of cell j, which are not necessarily of equal size. Let the part of the face volume of f that overlaps with i be given by $\Omega_{s_{f,i}} = d_{\perp_{f,i}} A_f$. Where $d_{\perp_{f,i}}$ is the perpendicular distance between centroid C_i and face A_f. By constructing this partial volume for every face adjacent to cell i, the sum of the face volumes can be compared to the volume of cell i. This sum is given by:

$$\Omega_{s_{\Sigma i}} = \sum_{f\in F_f(i)} d_{\perp_{f,i}} A_f \tag{146}$$

where $\sum_{f\in F_f(i)}$ sums over all faces adjacent to cell i. On the other hand, let cell i be divided by connecting the centroid to every vertex, into pyramids spanned by bases A_f and apex C_i. The volume of each pyramid is given by $\Omega_{c_{f,i}} = \frac{1}{3} d_{\perp_{f,i}} A_f$. The total volume of cell i can be expressed similar to the previous expression:

$$(\Omega_c)_{i,i} = \sum_{f\in F_f(i)} \frac{1}{3} d_{\perp_{f,i}} A_f \tag{147}$$

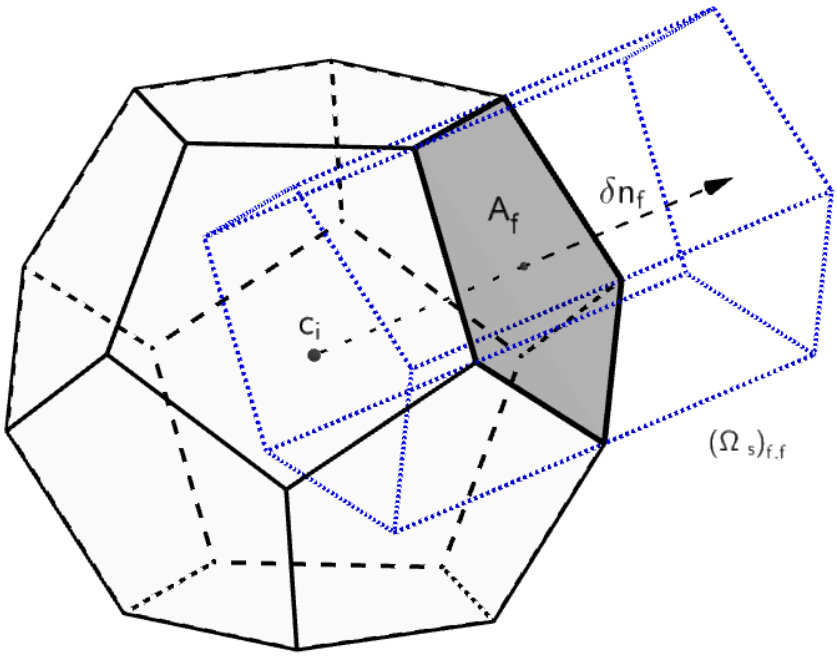


Figure 25: Construction of the staggered control volumes in three dimensions

When summing these volumes for all the cells, it is evident that the total volume in the staggered notation will be three times as big:

$$\Omega_{s_{\Sigma i}} = 3(\Omega_c)_{i,i} \tag{148}$$

But since Ω_c is a block diagonal matrix, which, in three dimensions, consists of three equal blocks that contain all cell volumes, it is evident that the sums of their traces are equal, i.e. equation 145 holds. This is under the trivial implication that the staggered volumes at the boundary of the domain are given by $d_{\perp_{f,i}} A_f$ and not $\delta n_f A_f$, since there is no δn_f when there is no neighbouring centroid. Moreover, this proof also holds in two dimensions, since the rectangles formed by $d_{\perp_{f,i}} A_f$, that construct the staggered volumes, have a surface two times bigger than the triangles spanned by the centroids and faces that construct the collocated volumes. Simultaneously, block diagonal matrix Ω_c consists of only two blocks, again resulting in equal traces of Ω_c and Ω_s

D Higher order discretizations

Higher order diffusive term

To find a higher order discretization of the diffusive term, a higher order gradient of the velocity is necessary. To do so, a one-dimensional case is considered with regular spaced grid points, as seen in figure 26.

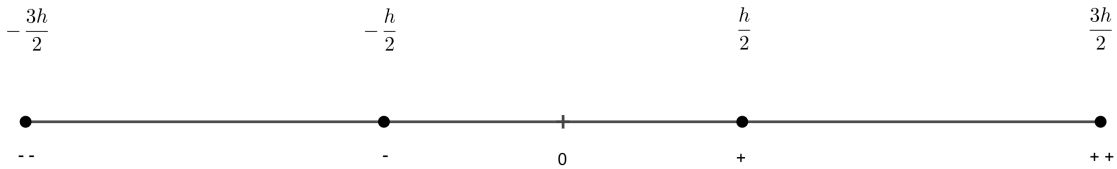


Figure 26: 1-dimensional regular grid

Taylor expansions is used on all four points:

$$u_{--} = u_0 - \frac{u_0^{(1)}}{1}\frac{3h}{2} + \frac{u_0^{(2)}}{2}\frac{9h^2}{4} - \frac{u_0^{(3)}}{6}\frac{27h^3}{8} + \frac{u_0^{(4)}}{24}\frac{81h^4}{16} \tag{149}$$

$$u_{-} = u_0 - \frac{u_0^{(1)}}{1}\frac{h}{2} + \frac{u_0^{(2)}}{2}\frac{h^2}{4} - \frac{u_0^{(3)}}{6}\frac{h^3}{8} + \frac{u_0^{(4)}}{24}\frac{h^4}{16} \tag{150}$$

$$u_{+} = u_0 + \frac{u_0^{(1)}}{1}\frac{h}{2} + \frac{u_0^{(2)}}{2}\frac{h^2}{4} + \frac{u_0^{(3)}}{6}\frac{h^3}{8} + \frac{u_0^{(4)}}{24}\frac{h^4}{16} \tag{151}$$

$$u_{++} = u_0 + \frac{u_0^{(1)}}{1}\frac{3h}{2} + \frac{u_0^{(2)}}{2}\frac{9h^2}{4} + \frac{u_0^{(3)}}{6}\frac{27h^3}{8} + \frac{u_0^{(4)}}{24}\frac{81h^4}{16} \tag{152}$$

Where $u^{(n)}$ gives the n-th spatial derivative of u. The following notation is then used:

$$\begin{aligned} u_0 &= T_0 \\ \frac{u_0^{(1)}}{1}\frac{h}{2} &= T_1 \\ \frac{u_0^{(2)}}{2}\frac{h^2}{4} &= T_2 \\ \frac{u_0^{(3)}}{6}\frac{h^3}{8} &= T_3 \\ \frac{u_0^{(4)}}{24}\frac{h^4}{16} &= T_4 \end{aligned} \tag{153}$$

To rewrite equations 149-152:

$$u_{--} = T_0 - 3T_1 + 9T_2 - 27T_3 + 81T_4 \tag{154}$$

$$u_{-} = T_0 - T_1 + T_2 - T_3 + T_4 \tag{155}$$

$$u_{+} = T_0 + T_1 + T_2 + T_3 + T_4 \tag{156}$$

$$u_{++} = T_0 + 3T_1 + 9T_2 + 27T_3 + 81T_4 \tag{157}$$

Which are summed with unknown weights A, B, C and D:

$$\begin{aligned} Au_{--} + Bu_{-} + Cu_{+} + Du_{++} = &(A+B+C+D)T_0 \\ &+ (-3A - B + C + 3D)T_1 \\ &+ (9A + B + C + 9D)T_2 \\ &+ (-27A - B + C + 27D)T_3 \\ &+ (81A + B + C + 81D)T_4 \end{aligned} \tag{158}$$

Only terms of order $u^{(1)}$ should remain in this sum, therefore:

$$\begin{aligned} A + B + C + D &= 0 \\ -3A - B + C + 3D &= 1 \\ 9A + B + C + 9D &= 0 \\ -27A - B + C + 27D &= 0 \\ 81A + B + C + 81D &= 0 \end{aligned} \tag{159}$$

Which is solved by:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -3 & -1 & 1 & 3 \\ 9 & 1 & 1 & 9 \\ -27 & -1 & 1 & 27 \\ 81 & 1 & 1 & 81 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{160}$$

Leading to: $A = \frac{1}{48}, \quad B = \frac{-27}{48}, \quad C = \frac{27}{48}, \quad D = \frac{-1}{48}$:

$$\begin{aligned}
\frac{1}{48}u_{--} - \frac{27}{48}u_{-} + \frac{27}{48}u_{+} - \frac{1}{48}u_{++} &= \frac{1}{48}T_0 - \frac{27}{48}T_0 + \frac{27}{48}T_0 - \frac{1}{48}T_0 && (=0T_0) \\
&- \frac{3}{48}T_1 + \frac{27}{48}T_1 + \frac{27}{48}T_1 - \frac{3}{48}T_1 && (=T_1) \\
&+ \frac{9}{48}T_2 - \frac{27}{48}T_2 + \frac{27}{48}T_2 - \frac{9}{48}T_2 && (=0T_2) \\
&- \frac{27}{48}T_3 + \frac{27}{48}T_3 + \frac{27}{48}T_3 - \frac{27}{48}T_3 && (=0T_3) \\
&+ \frac{81}{48}T_4 - \frac{27}{48}T_4 + \frac{27}{48}T_4 - \frac{81}{48}T_4 && (=0T_4) \\
&+ \mathcal{O}(h^5) \\
&= T_1 + \mathcal{O}(h^5) \\
&= \frac{1}{2}hu_0^{(1)} + \mathcal{O}(h^5)
\end{aligned}$$

The higher order gradient scheme is then found to be:

$$\frac{u_{--} - 27u_{-} + 27u_{+} - u_{++}}{24h} = u_0^{(1)} + \mathcal{O}(h^4) \tag{161}$$

To construct this scheme in OpenFOAM, possibly a combination between a cell-centered gradient and a face gradient can be used. In one dimension, a broad gradient, G_B, at + is defined to be:

$$[G_B u]_+ = \frac{u_{++} - u_{-}}{2h} \tag{162}$$

And similarly:

$$[G_B u]_- = \frac{u_{+} - u_{--}}{2h} \tag{163}$$

Using midpoint interpolation, $\Gamma_{c\to s}$, this gives:

$$[\Gamma_{c\to s} G_B u]_0 = \frac{-u_{--} - u_{-} + u_{+} + u_{++}}{4h} \tag{164}$$

So that:

$$\frac{-1}{6}[\Gamma_{c\to s}G_B u]_0 = \frac{u_{--} + u_- - u_+ - u_{++}}{24h} \tag{165}$$

Similarly the face gradient, G_s, at 0 is given by:

$$[G_s u]_0 = \frac{u_+ - u_-}{h} \tag{166}$$

So that:

$$\frac{28}{24}[G_s u]_0 = \frac{28u_+ - 28u_-}{24h} \tag{167}$$

Equations 165 and 167 are then summed to find:

$$\frac{7}{6}[G_s u]_0 + \frac{-1}{6}[\Gamma_{c\to s}G_B u]_0 = \frac{u_{--} - 27u_- + 27u_+ - u_{++}}{24h} = u_0^{(1)} + \mathcal{O}(h^4) \tag{168}$$

Which is a fourth order gradient scheme written in terms that can be implemented in OpenFOAM using a more general notation for unstructured grids. Using this notation on cartesian structured grids should however simplify to equation 168.
The cell-centered gradient that is used in OpenFOAM is a Gaussian gradient, it sums a gradient over each face similar to Gauss's theorem. The cell-centered velocity gradient at cell i, $[G_B(\mathbf{u}_c)]_i$, is given by the following expression:

$$[G_B(\mathbf{u}_c)]_i = \frac{1}{\Omega_{i,i}} \sum_{f\in F_f(i)} \left[\frac{\mathbf{u}_o + \mathbf{u}_n}{2}\right] \otimes \mathbf{n}_f A_f(\mathbf{n}_{\overrightarrow{ij}} \cdot \mathbf{n}_f) \tag{169}$$

To demonstrate that equation 168 holds with this definition of G_B, a two-dimensional regular cartesian grid is considered, as seen in figure 27.

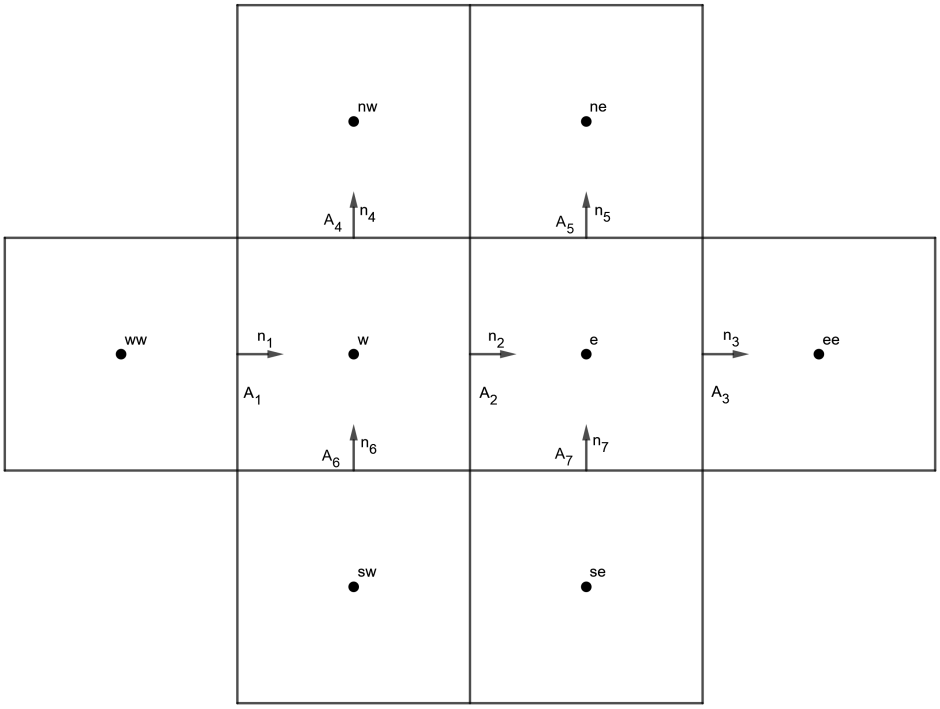


Figure 27: 2-dimensional cartesian grid containing the relevant cells for calculating the higher order gradient at face 2

First, the cell-centered gradients will be considered, $[G_B(\mathbf{u}_c)]_w$ specifically. Of this cell, the term considering face 4 is given by tensor:

$$
\begin{aligned}
[G_B(\mathbf{u}_c]_{w,4} &= \begin{pmatrix} \frac{u_{w,x}+u_{nw,x}}{2} \\ \frac{u_{w,y}+u_{nw,y}}{2} \end{pmatrix} \begin{pmatrix} n_{4,x} & n_{4,y} \end{pmatrix} A_4 \left[\begin{pmatrix} n_{\overrightarrow{w,nw},x} & n_{\overrightarrow{w,nw},y} \end{pmatrix} \cdot \begin{pmatrix} n_{4,x} \\ n_{4,y} \end{pmatrix} \right] \\
&= \begin{pmatrix} \frac{u_{w,x}+u_{nw,x}}{2} \\ \frac{u_{w,y}+u_{nw,y}}{2} \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} A_4 \left[\begin{pmatrix} 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right] \\
&= A_4 \begin{pmatrix} 0 & \frac{u_{w,x}+u_{nw,x}}{2} \\ 0 & \frac{u_{w,y}+u_{nw,y}}{2} \end{pmatrix}
\end{aligned} \tag{170}
$$

Similarly, the term for face 1 is given by:

$$
[G_B(\mathbf{u}_c]_{w,1} = -A_1 \begin{pmatrix} \frac{u_{ww,x}+u_{w,x}}{2} & 0 \\ \frac{u_{ww,y}+u_{w,y}}{2} & 0 \end{pmatrix} \tag{171}
$$

The cell-centered gradient is then given by the sum of the four face tensors divided by the cell volume:

$$
[G_B(\mathbf{u}_c)]_w = \frac{1}{\Omega_{w,w}} \begin{pmatrix} -A_1 \frac{u_{ww,x}+u_{w,x}}{2} + A_2 \frac{u_{w,x}+u_{e,x}}{2} & A_4 \frac{u_{w,x}+u_{nw,x}}{2} - A_6 \frac{u_{sw,x}+u_{w,x}}{2} \\ -A_1 \frac{u_{ww,y}+u_{w,y}}{2} + A_2 \frac{u_{w,y}+u_{e,y}}{2} & A_4 \frac{u_{w,y}+u_{nw,y}}{2} - A_6 \frac{u_{sw,y}+u_{w,y}}{2} \end{pmatrix} \tag{172}
$$

The cell-centered gradients of w and e are then interpolated to face 2:

$$
\begin{aligned}
[\Gamma_{c\to s} G_B(\mathbf{u}_c)]_2 &= \frac{[G_B(\mathbf{u}_c)]_w}{2}\cdot n_2 + \frac{[G_B(\mathbf{u}_c)]_e}{2}\cdot n_2 \\
&= \begin{pmatrix} -\frac{A_1}{\Omega_{w,w}}\frac{u_{ww,x}+u_{w,x}}{4} + \frac{A_2}{\Omega_{w,w}}\frac{u_{w,x}+u_{e,x}}{4} - \frac{A_2}{\Omega_{e,e}}\frac{u_{w,x}+u_{e,x}}{4} + \frac{A_3}{\Omega_{e,e}}\frac{u_{e,x}+u_{ee,x}}{4} \\ -\frac{A_1}{\Omega_{w,w}}\frac{u_{ww,y}+u_{w,y}}{4} + \frac{A_2}{\Omega_{w,w}}\frac{u_{w,y}+u_{e,y}}{4} - \frac{A_2}{\Omega_{e,e}}\frac{u_{w,y}+u_{e,y}}{4} + \frac{A_3}{\Omega_{e,e}}\frac{u_{e,y}+u_{ee,y}}{4} \end{pmatrix}
\end{aligned} \tag{173}
$$

This expression can be simplified by noticing $A_1 = A_2 = A_3 = A_f$, $\Omega_{e,e} = \Omega_{w,w} = A_f^2$ and $A_f = h$, with h the distance between two cell centers:

$$
[\Gamma_{c\to s} G_B \mathbf{u}_c]_2 = \frac{-\mathbf{u}_{ww} - \mathbf{u}_w + \mathbf{u}_e + \mathbf{u}_{ee}}{4h} \tag{174}
$$

which agrees with equation 164. Notice that the contributions from cells nw, ne, sw and se all vanish. This is because the faces over which these contributions have to be carried are orthogonal. The dot product of the interpolation step cancels the outer product that was taken when calculating the cell-centered gradient.
The face gradient in OpenFOAM is calculated by:

$$
[G_s(\mathbf{u}_c)]_2 = \frac{\mathbf{u}_e - \mathbf{u}_w}{|\mathbf{n}_2 \cdot \overrightarrow{w,e}|} \tag{175}
$$

which in this example simplifies to:

$$
[G_s(\mathbf{u}_c)]_2 = \frac{\mathbf{u}_e - \mathbf{u}_w}{h} \tag{176}
$$

just like equation 166. Both components of equation 168 have now been established. Using the right combination of weights, this leads to the higher order discretization of the diffusive term.

Higher order interpolation method

Similarly, a higher order interpolation (HOI) can be calculated using Taylor expansions. A derivation of u_0 then has to be made, in contrast with $u_0^{(1)}$ needed for the HOD. The derivation is similar, but equation 159 is replaced by:

$$
\begin{aligned}
A + B + C + D &= 1 \\
-3A - B + C + 3D &= 0 \\
9A + B + C + 9D &= 0 \\
-27A - B + C + 27D &= 0
\end{aligned} \tag{177}
$$

and:

$$
\begin{pmatrix} 1 & 1 & 1 & 1 \\ -3 & -1 & 1 & 3 \\ 9 & 1 & 1 & 9 \\ -27 & -1 & 1 & 27 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{178}
$$

Leading to: $A = \frac{-1}{16}$, $B = \frac{9}{16}$, $C = \frac{9}{16}$, $D = \frac{-1}{16}$, and:

$$\frac{-u_{--} + 9u_{-} + 9u_{+} - u_{++}}{16} = u_0 + \mathcal{O}(h^4) \tag{179}$$

This closely resembles a variant of the cubic interpolation scheme in OpenFOAM:

$$u_0 = u(1-\lambda) = \lambda u_- + (1-\lambda)u_+ + \lambda(1-\lambda)(1-2\lambda)(u_+ - u_-) - \lambda^2(\lambda-1)\nabla_- - \lambda(1-\lambda)^2\nabla_+ \tag{180}$$

To preserve symmetry in the interpolation, $\lambda = \frac{1}{2}$ is taken, so that this method simplifies to:

$$u_0 = \frac{1}{2}u_- + \frac{1}{2}u_+ + h_0\frac{1}{8}[\nabla u]_{cor,0} \tag{181}$$

Where the correcting gradient terms for the face, $[\nabla u]_{cor}$, are found by interpolating seperate gradients for owner and neighbour cells, for each dimensional component of u, x_i. These dimensional components are subsequently summed.

$$[\nabla u]_{cor} = \sum_i \left[[\nabla u]_{xi,o} - [\nabla u]_{xi,n}\right] \cdot n_f \tag{182}$$

With gradients of each cell, k, calculated as:

$$[\nabla u]_{x_i,k} = \sum_{f \in F_f(C_k)} \frac{u_{x_i,o} - u_{x_i,n}}{\overrightarrow{C_o C_n}} \tag{183}$$

On a three dimensional Cartesian grid, vectors $\overrightarrow{C_o C_n}$ in the y- and z-direction cancel out with n_{f_0}, leaving:

$$[\nabla u]_{cor} = \frac{1}{2}\left[\frac{(u_- - u_{--})}{h_-} - \frac{(u_{++} - u_+)}{h_+}\right] \tag{184}$$

When there is also no stretching, $h_- = h_0 = h_+$, the complete expression for the interpolated variable becomes:

$$u_0 = \frac{9}{16}u_- + \frac{9}{16}u_+ - \frac{1}{16}u_{--} - \frac{1}{16}u_{++} \tag{185}$$

Which is higher order midpoint.

It should be noted that $\lambda = \frac{\overrightarrow{C_- 0}}{h_0}$, equals $\frac{1}{2}$ on Cartesian grids. Therefore, this method is exactly the same as the cubic method on Cartesian grids, analogous to a linear interpolation on a Cartesian grid becoming equal to a midpoint interpolation. These higher order discretizations offer interesting alternatives to the ones used in this work, although first results did not prove these methods to lead to large differences in the examined cases.

E Disclaimer

EU DuC=N

Goods labeled with an EU DuC (European Dual-use Codification) not equal to ‘N’ are subject to European and national export authorization when exported from the EU and may be subject to national export authorization when exported to another EU country as well. Even without an EU DuC, or with EU DuC ‘N’, authorization may be required due to the final destination and purpose for which the goods are to be used. No rights may be derived from the specified EU DuC or absence of an EU DuC.