Velocity interpolation and streamline tracing on irregular geometries

R.A. Klausen A. F. Rasmussen A.F. Stephansen

Abstract

Streamline tracing on irregular grids requires reliable interpolation of velocity fields. We propose a new method for direct streamline tracing on polygon and polytope cells. While some numerical methods provide a basis function that can be used for interpolation, other methods provide only the fluxes at the faces of the elements. We introduce the concept of full field and raw field methods. Full field methods have built-in interpolation, but are often not defined on general grids such as polygonal and polyhedral grids which we examine here. Also, reliability issues may arise on non-simplicial meshes in terms of not being able to reproduce constant velocity fields. We propose an interpolation in H(div) and H(curl) valid on general grids that is based on barycentric coordinates and that reproduces uniform flow. The interpolation can be used to compute the streamline directly on the complex cell geometry. The method generalizes to convex polytopes in 3D, with a restriction on the polytope topology near corners that is shown to be satisfied by several popular grid types. Numerical results confirm that the method is applicable to general grids and preserves uniform flow.

Key words: streamlines, interpolation, H(div), H(curl), raw field and full field methods, general mesh, polygons, polyhedra, uniform flow, barycentric coordinates.

1 Introduction

Efficiency and accuracy are essential goals in all numerical simulations. In reservoir simulations, the size of the domain coupled with the necessity to incorporate local geological effects are core issues which must be tackled. Flexible gridding with a contained number of elements is therefore sought. In cases with thin layers and complex domains, one shape regular polygon may replace several shape regular simplexes. On polygonal grids existing numerical methods offer discrete edge fluxes, but the velocity field inside the cells is not defined. Obtaining a suitable velocity interpolation, the field may be visualized by tracing streamlines. Streamlines may also prove useful in solving hyperbolic differential equations. We look at what numerical methods might be better suited for general grids, how to interpolate the velocity field and how to trace streamlines.

When considering numerical methods used to obtain a velocity field it is natural to distinguish between those that provide a field that is defined inside each element and those that define the field only at the interfaces of the elements. We will name the former type of method as full field methods and the latter as raw field methods. The name 'raw field' is to indicate that the resultant vector field is in need of post-processing (an interpolation) in order to obtain a field that is everywhere defined. Raw field methods most commonly calculate either the flux or the circulation of the velocity field, dependent on the equation given. If the velocity raw field is given as fluxes we want our interpolated raw field to be in the $H(\text{div})$ space. If the velocity raw field is given as circulations, the final interpolated velocity field should be in $H(\mathrm{curl})$. Many numerical methods are based on the property of giving exact results for linear pressure fields, and it is natural then that this is reflected in the interpolation of the results. We will give more attention to the velocity interpolation in the $H(\mathrm{div})$ space since we will use it for streamline tracing.

One possible way to obtain a velocity interpolation on general grids that reproduces uniform flow is to divide the grid into simplices. The extension of the mixed finite element method to hexahedra follows this path. However, the finer grid introduces a more lengthy procedure which we would like to avoid. Also, raw field methods are defined directly on the grid and we would like to implement our velocity interpolation on this basis. The key to this approach is the use of generalized barycentric coordinates.

In reservoir simulation streamline tracing may be used to solve the transport equation for the saturation. Along streamlines the transport equation is one-dimensional which leads to fast solvers. The resulting saturation is subsequently mapped back to the grid. It is important that the streamlines are spatially accurate, so that for instance non-permeable regions are kept as such. As there is more to streamline tracing than just obtaining the velocity, we will also discuss some of the difficulties involved.

In the following we will, after introducing the mathematical setting, discuss some of the main differences between introducing a velocity interpolation in the definition of the numerical method and to see it as a post-processing issue. We will see that this has particular importance with regards to convergence properties of the numerical methods. We then introduce the velocity interpolation and show that it has the desired properties. The section that follows discuss how to use the interpolated velocity to obtain accurate streamlines. Numerical results show the validity of the velocity interpolation and the streamline tracing.

2 Preliminaries

Let Ω be a bounded domain in $\boldsymbol{R}^d$, d=2,3 with polygonal boundary $\partial\Omega$. We indicate with $\mathcal{L}_2(E)$ the space of functions whose square is Lebesgue-integrable, with inner product $(\cdot\,,\cdot)_{0,E}$ and norm $\|\cdot\|_E = (\cdot\,,\cdot)_{0,E}^{1/2}$. Let $H^1(E)$ denote the Sobolev space of first order differentiable functions in $\mathcal{L}_2(E)$. Furthermore we define the spaces

$$H(\mathrm{div}; E) = \{\boldsymbol{v} \in (\mathcal{L}_2(E))^2 : \mathrm{div}\,\boldsymbol{v} \in \mathcal{L}_2(E)\},$$

equipped with the norm $\|\boldsymbol{v}\|_{\mathrm{div},E} = (\|\boldsymbol{v}\|_{0,E}^2 + \|\,\mathrm{div}(\boldsymbol{v})\|_{0,E}^2)^{1/2}$, and

$$H(\mathrm{curl}; E) = \{\boldsymbol{v} \in (\mathcal{L}_2(E))^2 : \mathrm{curl}\,\boldsymbol{v} \in \mathcal{L}_2(E)\},$$

equipped with the norm $\|\boldsymbol{v}\|_{\mathrm{curl},E} = (\|\boldsymbol{v}\|_{0,E}^2 + \|\,\mathrm{curl}(\boldsymbol{v})\|_{0,E}^2)^{1/2}$. The notation $H(\mathrm{div})$ and $H(\mathrm{curl})$ is used when the domain considered is Ω.

As our focus will mostly be on $H(\mathrm{div})$, we introduce the model equation used to obtain the Darcy velocity $\boldsymbol{u}$ in reservoir simulation. We seek $(\mathbf{u}, p) \in H(\mathrm{div}) \times \mathcal{L}_2(\Omega)$ such that

$$\begin{cases} (K^{-1}\mathbf{u}, \mathbf{v})_{0,\Omega} - (p, \mathrm{div}\,\mathbf{v})_{0,\Omega} = 0, & \forall \mathbf{v} \in H(\mathrm{div}), \\ (\mathrm{div}\,\mathbf{u}, q)_{0,\Omega} = (f, q)_{0,\Omega} & \forall q \in \mathcal{L}_2(\Omega), \end{cases} \tag{1}$$

which implies homogeneous Dirichlet boundary conditions for the pressure p equal to zero. For the well posedness of the system let $f \in \mathcal{L}_2(\Omega)$ and let the permeability tensor K be a symmetric, positive definite field whose smallest eigenvalue is bounded from below by a positive constant.

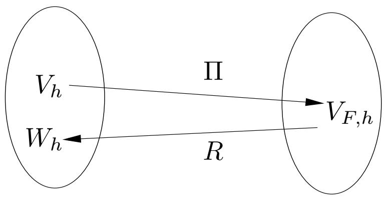

Figure 1: Projection and reconstruction

2.1 Raw field and full field methods

For equations where the velocity field is an unknown, we distinguish between discretizations that lead to either a raw (velocity) field or a full (velocity) field: by a raw field method we intend a discretization that specifies the velocity only at the interfaces of the elements and not inside the elements themselves. Conversely, a full field method will specify basis functions for the velocity on each element. Typical raw field methods are finite difference and finite volume methods, while finite element methods are full field methods.

For full field methods, we have a full field $V_h \subset H(D)$, $D =$ div or curl, represented by the finite basis $\{\phi_i\}$ on each element. We define the representation operator $\mu_h : V_h \to \mathbf{R}^{\#dofs}$ by

$$v = \sum_i (\mu_h v)_i \phi_i,$$

representing the degrees of freedom of the discrete methods. We denote the collection of edges for F. Since we are concerned with the discrete values associated with the edges, we also define a projection operator

$$\Pi_h : V_h \to V_{F,h}$$

mapping from the (everywhere defined) velocity field V_h into the raw field $V_{F,h}$, specified only at edges. Where fluxes are concerned, note that μ_h and Π_h may not necessary map to the same set of discrete values. The ABF-elements (see following section) for instance have more degrees of freedom than we have edges, so here $\Pi_h(V_h) \subset \mu_h(V_h)$.

Starting with the raw field, obtained either by a raw field method or from projection from a full field method, we define a reconstruction operator

$$\mathcal{R}_h : V_{\mathcal{E},h} \to W_h.$$

Note that by this definition $\mathcal{R}_h \Pi_h$ is not necessarily equal to the identity operator, as V_h and W_h may be different, see Figure 1. Examining the mixed finite element method, we shall see in the next section that $W_h = V_h$ may indeed not be the optimal choice where convergence of the method is concerned.

2.2 Convergence issues

Full field methods are mainly defined on simplicial grids and quadrilateral/hexahedral grids, due to the construction based on a mapping from a unit reference cell. In this section we discuss methods for the diffusion problem (1) defined on quadrilateral grids, which illustrates some difficulties that become even more evident on more complicated grids including 3D. We summarize some convergence results to illustrate the weaknesses

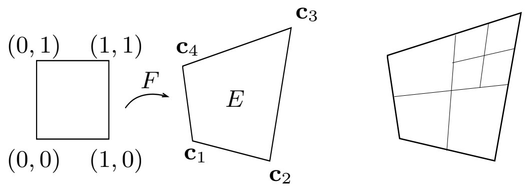

Figure 2: Mapping of quadrilateral cell from the unit square and uniform refinement

of some of the full field interpolations, and to motivate the definitions given in the next section.

The raw field methods we are considering are the mimetic finite difference (MFD) method, eg. [6], and the multi point flux approximation control volume (MPFA) methods. In particular we look at the MPFA O-method eg. [1] and its symmetric version eg. [2]. Amongst full field methods we examine the following mixed finite element (MFE) methods: the first order Raviart-Thomas (RT_0) [5], the Arnold-Boffi-Falk (ABF) [3] and the Barycentric Coordinate Interpolation (BCI) [16].

Let $\{\mathcal{T}_h\}$ denote a family of partitions of Ω into convex quadrilateral elements with vertices $\mathbf{c}_i$, $i = 1, \ldots, 4$, and h is the maximum element edge. Each cell may be mapped from a unit square by

$$F(\hat{x}, \hat{y}) = (1-\hat{x})(1-\hat{y})\mathbf{c}_1 + \hat{x}(1-\hat{y})\mathbf{c}_2 + \hat{x}\hat{y}\,\mathbf{c}_3 + (1-\hat{x})\hat{y}\,\mathbf{c}_4 = \sum_{i=1}^{4} \lambda_i(\hat{x},\hat{y})\,\mathbf{c}_i \qquad (2)$$

for $(\hat{x},\hat{y}) \in (0,1)\times(0,1)$, cf. Fig 2. It is possible to refine the mesh uniformly by dividing each cell edge in its midpoint at each refinement level. This will result in quadrilaterals that asymptotically approach parallelograms as h tends to zero. Such a family of meshes will be indicated by uniformly refined meshes or smooth meshes, cf. Fig 2. General quadrilateral grids without any asymptotic refinement condition on $\mathcal{T}_h$ are referred to as rough grids.

Another way of referring to uniformly refined meshes is to say that they are h^2–meshes, meaning that there exists a σ such that $|F_{\hat{x}\hat{y}}| \leq \sigma h^2$. The Piola transformation uses the mapping F and has the property that $H(\mathrm{div})$ functions are preserved as such. If we denote the Jacobian matrix of F by D and the Jacobian determinant of F by J, we can define the Piola transform $\mathcal{P}$ by

$$\boldsymbol{v}(\mathbf{x}) = \mathcal{P}\hat{\boldsymbol{v}}(\mathbf{x}) = \frac{1}{J} D\hat{\boldsymbol{v}} \circ F^{-1}(\mathbf{x}). \qquad (3)$$

Here $\hat{\boldsymbol{v}}$ is the vector function $\boldsymbol{v}$ defined on the unit reference cell. Note that the Piola mapping is a rational function. If the Jacobian is constant, then linear functions are mapped into linear functions. However, general quadrilaterals do not have a constant Jacobian, thus leading to grid convergence problems. Lack of rough grid convergence can therefore manifest itself for methods that make use of the Piola mapping. The full field methods RT and ABF, but also the symmetric MPFA method [2, 17] (a raw field method), use the Piola mapping in their definition.

From Table 1 we see however that the Piola mapping does not immediately indicate if the method is going to converge on rough grids or not. The BCI MFE method does not converge on rough grids even though it is defined directly on the physical grid. The BCI MFE method does not in fact satisfy the inf-sup condition on general grids. However,

Method	Velocity field	Piola mapping	Convergence	
			Smooth grids	Rough grids
MPFA symmetric	Raw	Yes	Yes	No
MPFA standard	Raw	No	Yes	Yes
MFD	Raw	No	Yes	Yes
RT_0 MFE	Full	Yes	Yes	Norm-dependent
ABF MFE	Full	Yes	Yes	Yes
BCI MFE	Full	No	Yes	No

Table 1: Some numerical methods for the diffusion problem and their characteristics

$\boldsymbol{u}_h \in$	$\mathcal{T}_{h,\text{rough}}$ $\Vert\boldsymbol{u}-\boldsymbol{u}_h\Vert_{\mathcal{L}_2}$	$\mathcal{T}_{h,\text{rough}}$ $\Vert\operatorname{div}(\boldsymbol{u}-\boldsymbol{u}_h)\Vert_{\mathcal{L}_2}$	$\mathcal{T}_{h^2}$ $\Vert\boldsymbol{u}-\boldsymbol{u}_h\Vert_{H(\text{div})}$
$\mathcal{R}_{\text{ABF}}\Pi_h u_{h,\text{ABF}}$	h	h	h
$\mathcal{R}_{\text{RT}_0}\Pi_h u_{h,\text{RT}_0}$	h	NO	h
$\mathcal{R}_{\phi}\Pi_h u_{h,\text{RT}_0}$	h	h	h
$\mathcal{R}_{\text{BCI}}\Pi_h u_{h,\text{BCI}}$	NO	NO	h
$\mathcal{R}_{\text{BCI}}\Pi_h u_{h,\text{RT}_0}$	h	h	h

Table 2: Order of convergence of differently interpolated velocity fields of the MFE methods on a family of rough grids $\mathcal{T}_{h,\text{rough}}$ and uniformly refined grids $\mathcal{T}_{h^2}$ in $H(\text{div})$ norms

on parallelogram grids the BCI MFE method coincides with the lowest order Raviart-Thomas, RT_0 MFE method, and this is why the convergence on smooth grids is obtained. We also note that though the ABF MFE method is defined using the Piola mapping, the rough grid convergence is obtained due to a careful choice of degrees of freedom. The definition of the element is however limited to 2D.

The entry 'Norm-dependent' in Table 1 needs to be further explained. It is in fact known that while the velocity field (as well as the pressure) of the RT_0 method converges in the $\mathcal{L}_2$ norm, this is not so for the divergence of the velocity field. As the velocity space is in $H(\text{div})$ this is problematic. The choice of degrees of freedom for the ABF element is made exactly to overcome this difficulty. It is however possible to obtain the same result by operating a postprocessing procedure, as is further explained below in conjunction with Table 2.

The results regarding convergence in H(div) norms are summarized in Table 2 where the subscript of $\mathcal{R}$ indicates the basis functions used for interpolation. The subscript of $\boldsymbol{u}$ indicates the method used, while the abbreviate Φ refers to a bubble function reconstruction, cf. [18]. The first line of Table 2 refers to the MFE solution with Arnold-Boffi-Falk elements, $\boldsymbol{u}_{\text{ABF}} = \mathcal{R}_{\text{ABF}}\Pi_h\boldsymbol{u}_{h,\text{ABF}}$, cf. [3]. The next lines refer to the MFE solution with Raviart-Thomas elements, $\boldsymbol{u}_{\text{RT}_0} = \mathcal{R}_{\text{RT}_0}\Pi_h\boldsymbol{u}_{h,\text{RT}_0}$, cf [5]. Due to the non constant Piola mapping twisting the character of interpolation, the $\boldsymbol{u}_{\text{RT}_0}$ does not converge in the full $H(\text{div})$ norm, cf. [3]. One way to overcome the problem is to add a different interpolation on the divergence term, as we will now see.

Let $\hat{RT}_0$ denote Raviart-Thomas space on the reference cell, with $RT_0 = \mathcal{P}\hat{RT}_0$. Define the bubble function $\boldsymbol{v}_{E,\phi}$ on each cell as $\boldsymbol{v}_{E,\phi} = \mathcal{P}\hat{\boldsymbol{v}}_{E,\phi}$, for all cells E and

$\boldsymbol{v} \in RT_0$, such that

$$\hat{\boldsymbol{v}}_{E,\phi} = \hat{\boldsymbol{v}} + \frac{\operatorname{div} \hat{\boldsymbol{v}}}{2(\Delta_1 + \Delta_3)} \begin{bmatrix} (\Delta_2 - \Delta_1)\hat{x}(\hat{x}-1) \\ (\Delta_4 - \Delta_1)\hat{y}(\hat{y}-1) \end{bmatrix}. \tag{4}$$

The symbol Δ_i indicates the area spanned by the adjacent cell edges of vertex i on cell E. Define

$$V_\phi = \{\boldsymbol{v} \in \mathrm{RT}_0 | \operatorname{div} \boldsymbol{v}|_E = \operatorname{div} \boldsymbol{v}_{E,\phi}, \forall E \in \mathcal{T}_h\},$$

and $\mathcal{R}_\phi : V_{F,h} \to V_\phi$, the bubble interpolation operator. The construction in (4) ensures that $\operatorname{div} \boldsymbol{v}_{E,\phi} = \operatorname{div}(\hat{\boldsymbol{v}})/J_c \in P_0(E), \forall E \in \mathcal{T}_h$ and $\boldsymbol{v} \in RT_0$. It can also be shown that the divergence of the reconstruction based on the lowest order Raviart-Thomas elements, $\operatorname{div} R_h$, converges in $\mathcal{L}_2$. More details can be found in [18] eq. 12. The problems of $H(\operatorname{div})$ convergence on rough grids is connected to the $\mathcal{R}_{RT_0}$ interpolation, while the RT_0 raw field is fine. Finally we look at two results using the velocity field of the Barycentric Coordinate Interpolation: $\boldsymbol{u}_{\mathrm{BCI}} = \mathcal{R}_{\mathrm{BCI}} \Pi_h \boldsymbol{u}_{h,\mathrm{BCI}}$. This velocity field does not converge on rough grids cf. [16] when implemented as a mixed finite element method. However, using the interpolation as a post-processing of the fluxes calculated with the RT_0 method, convergence is obtained in all norms considered here. For simplicity we just state the result given in the last line of the Table 2. The proof can be found in Appendix A.

Lemma 1

$$\|\boldsymbol{u} - \mathcal{R}_{\mathrm{BCI}} \Pi_h \boldsymbol{u}_{h,RT_o}\|_{H(\operatorname{div})} \le Ch|\boldsymbol{u}|_{H^2}, \qquad \textit{for all } \boldsymbol{u} \in H^2.$$

2.3 Local conservation and pointwise divergence

All the methods presented here are locally conservative, in the sense that

- the sum of the flux over each cell edge (or face in 3D) equals the accumulation, sink or source in the actual cell, and
- the flux is continuous across the cell edges (or faces in 3D).

It is worth noting that the definition here is for a discrete method, meaning that the definition of the integration area (the volume for which the conservation law is valid) is not arbitrary. The methods are in general not designed to be pointwise divergence free in an area with no sink, source, or accumulation term.

The isoparametric mapping (2) unfortunately does not preserve the local conservation property when mapping the reference space to the physical space. The Piola mapping, cf. eq. (3), is instead specifically designed with the local conservation property in mind. If the velocity field is defined by (3), then for each cell E, the Piola mapping is constructed such that

$$\int_E \operatorname{div} \boldsymbol{u} \, d\mathbf{x} = \int_{\hat{E}} \operatorname{div} \hat{\boldsymbol{u}} \, d\hat{\mathbf{x}}. \tag{5}$$

Furthermore, this gives the pointwise divergence relation

$$\operatorname{div} \hat{\boldsymbol{u}}(\mathbf{x}) = J(\mathbf{x}) \operatorname{div} \boldsymbol{u}(\mathbf{x}).$$

The Piola mapping also preserves the flux as one (continuous) quantity over each face F;

$$\int_F \boldsymbol{u} \cdot \mathbf{n} \, ds = \int_{\hat{F}} \hat{\boldsymbol{u}} \cdot \hat{\mathbf{n}} \, d\hat{s}.$$

In the light of the problem with the Piola mapping presented in the preceding section, we give an approach for interpolation which does not make use of the mapping and that

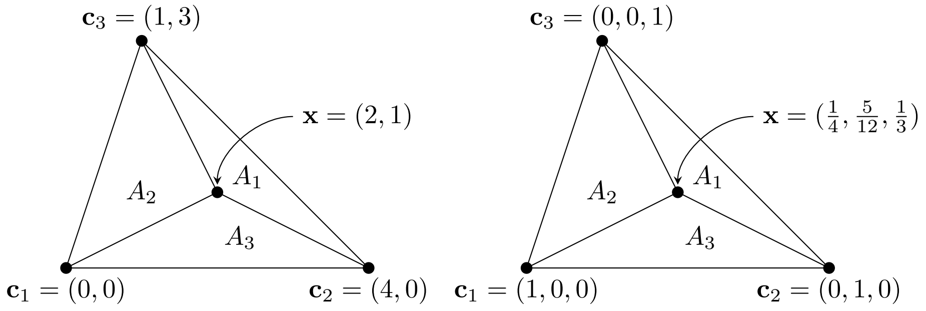

Figure 3: Cartesian vs. barycentric coordinates.

can be extended to simple polytopes. Another aspect that makes us focus on interpolation directly in the physical space is the wish to reproduce uniform flow. Numerical methods like the MPFA methods and the MFD method have been developed precisely so as to give exact fluxes when the pressure field is linear. For a general grid the reproduction of uniform flow in the reference space and in the physical space are two different things, as the Piola mapping is not constant anymore. This can be seen for instance in the difference between the definition of MPFA on the reference space (symmetric) and MPFA on the physical space (standard, non symmetric). For raw field methods we also note that the velocity field is defined only in terms of fluxes, which is an integrated quantity and not a pointwise one. This observation is interesting when considering the construction of interpolations, which try to fit a pointwise velocity field in accordance with average velocities (flux divided by length or area) given at the boundary. For uniform flow regimes however the average and the pointwise velocity coincide, giving a strong incentive to reproduce these fields exactly when interpolating.

3 Interpolation

3.1 Barycentric coordinates

For simplices, barycentric coordinates are well known, and may be defined as follows: let the corners of a simplex be denoted by $\{\mathbf{c}_1, \ldots, \mathbf{c}_{n+1}\}$, $\mathbf{c}_i \in \mathbf{R}^n$. Let $A_i(\mathbf{x})$ be the signed volume (or area) of the simplex created with the corner $\mathbf{c}_i$ replaced by $\mathbf{x}$, as shown in Figure 3. Then the barycentric coordinate functions $\lambda_1, \ldots, \lambda_n$ are uniquely defined by

$$\lambda_i(\mathbf{x}) = \frac{A_i(\mathbf{x})}{\sum_{i=1}^{n+1} A_i(\mathbf{x})}.$$

Figure 3 shows cartesian and barycentric coordinates for a simplex in 2D.

For a convex polytope $E \subset \mathbf{R}^n$ we will use generalized barycentric coordinates. We assume that the polytope is bounded by hyperplanes and that no curved boundaries are present. The latter limitation is due to our focus on reproducing uniform flow while maintaining only one unknown (flux) per face, cf. [20]. Let the corners of E be denoted by $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$. Then a set of functions $\{\lambda_1, \ldots, \lambda_k\}$ are generalized barycentric coordinates for E if the following holds:

B1 $\sum_{i=1}^{k} \lambda_i(\mathbf{x}) = 1$.

B2 $\lambda_i(\mathbf{c}_j) = \delta_{ij}$.

B3 $\lambda_i(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in E$.

B4 $\sum_{i=1}^k \lambda_i(\mathbf{x})\mathbf{c}_i = \mathbf{x}$.

(B5) For a point $\mathbf{x}$ on a face F_i of the polytope, we have that $\lambda_j(\mathbf{x}) = 0$ when $\mathbf{c}_j$ is not a corner of F_i. For convex polytopes this property is not independent, but follows from B3 and B4.

These properties also characterise the barycentric coordinates of a simplex. The one property of simplicial barycentric coordinates that is not carried over to polygons is linearity of λ_i.

Two important examples of generalized barycentric coordinate functions are Wachspress' coordinates described in [25, 19] and the Mean Value coordinates of [9]. We do not give any details of the computation of $\lambda_i(\mathbf{x})$ here, but refer the reader to the abovementioned papers. Both families produce smooth functions on the polytope E. While being initially described for the 2D case, both have generalizations to 3D. On quadrilaterals, one example of barycentric coordinates may be the bilinear coordinates defined by λ_i in equation (2). These turn out to coincide with the Wachspress coordinates for the rectangular case.

3.2 Interpolation of divergence free flow fields on simplices

As mentioned, barycentric coordinates are uniquely defined for simplices. For a divergence free flow the velocity interpolation in $H(\text{div})$ on such elements is also particularly simple as seen in the following lemma:

Lemma 2 *Let $S \subset \mathbf{R}^n$ be a simplex with corners $c_i, i = 1, \ldots, n+1$ and let S_i be the (codimension 1) subsimplex of S that does not contain the corner c_i. For a set of fluxes $\mathbf{f} = (f_i)$ defined on the S_i aligned with the outward normals and with the property that $\sum_{i=0}^n f_i = 0$, the constant velocity*

$$\mathbf{v} = -\mathbf{f}/(n|S|) \tag{6}$$

in barycentric coordinates is consistent with the normal fluxes $\mathbf{f}$. In the above, $|S|$ is the measure of S and n is the dimension.

Proof. We first note that the corner c_1 of the simplex has barycentric coordinates $(1, 0, \ldots, 0)$ and similarly for the other corners. We may therefore also write the velocity as

$$\mathbf{v} = -\frac{1}{n|S|} \sum_{i=1}^{n+1} f_i \mathbf{c}_i.$$

Without loss of generality, we show that $\hat{\mathbf{v}}$ is consistent with f_1. If f_1 is zero, we substitute $f_2 = -\sum_{i=3}^{n+1} f_i$ and obtain

$$\mathbf{v} = -\frac{1}{n|S|} \sum_{i=3}^{n+1} f_i(\mathbf{c}_i - \mathbf{c}_2).$$

Since all the vectors $\mathbf{c}_i - \mathbf{c}_2$ are tangent to S_1, clearly the normal component of $\hat{\mathbf{v}}$ and therefore the normal flux through S_1 is zero.

If f_1 is nonzero, we write

$$\mathbf{v} = -\frac{f_1}{n|S|}\big(\mathbf{c}_1 + \sum_{i=2}^{n+1} \frac{f_i}{f_1}\mathbf{c}_i\big) = -\frac{f_1}{n|S|}\left(\mathbf{c}_1 - \mathbf{r}\right).$$

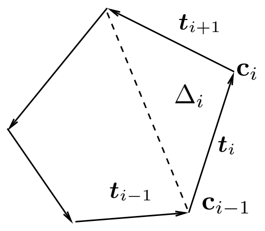

Figure 4: The vertices and edges of a polygonal cell.

Since $-\sum_{i=2}^{n} f_i/f_1 = 1$, we know that $\mathbf{r}$ must lie in the hyperplane tangent to S_1. Therefore the outward pointing normal component of $\mathbf{c}_1 - \mathbf{r}$ must equal $-d_1$, where d_1 is the distance from the hyperplane to $\mathbf{c}_1$. Since $d_1|S_1| = n|S|$, it follows that the normal flux is equal to $-f_1(-d_1)|S_1|/(n|S|) = f_1$. □

3.3 Interpolation of $H(\text{div})$ raw fields in 2D

In the following we will need the nomeclature as shown in Figure 4. Define the edge vector $\boldsymbol{t}_i$ associated with $F_i \in \partial E$, for $i = 1, \ldots, n$ numbered in an anticlockwise direction. The length of $\boldsymbol{t}_i$ is equal to the length of the edge F_i itself. Also define the triangle area Δ_i associated with the vertex $\mathbf{c}_i$ of E, spanned by the edges F_i and F_{i+1} and equal to $\frac{1}{2}\boldsymbol{t}_i \times \boldsymbol{t}_{i+1}$. Notation is circular, so that the index $i = n+1$ is to be read as $i = 1$.

On polygons we need to use generalized barycentric coordinates. We define edge-based basis functions that we will use to construct a velocity field given the fluxes:

Definition 1 *Let $\{\lambda_i\}$ be a set of barycentric functions on a n-polygon. For every edge $F_i \subset \partial E$ define the function*

$$\boldsymbol{\psi}_i(\mathbf{x}) = \frac{\boldsymbol{t}_{i-1}}{2\Delta_{i-1}}\lambda_{i-1}(\mathbf{x}) - \frac{\boldsymbol{t}_{i+1}}{2\Delta_i}\lambda_i(\mathbf{x}). \tag{7}$$

Definition 2 *Let $\{f_i\}$ be a discrete set of fluxes associated to the faces F_i of an element $E \in \mathcal{T}_h$. The interpolated velocity field v on $\mathcal{T}_h$ is defined as*

$$v(\mathbf{x})|_E = \sum_{F_i \in \partial E} f_i \boldsymbol{\psi}_i|_E(\mathbf{x}). \tag{8}$$

Theorem 1 *Let $\boldsymbol{n}_j$, $j = 1, \ldots, n$, be the outer normal vector to the edge F_j, with length equal to the length of F_j. Let $\mathbf{x}_{F_j} \in F_j$. Then*

$$\boldsymbol{\psi}_i(\mathbf{x}_{F_j}) \cdot \boldsymbol{n}_j = \delta_{ij}.$$

Proof. We have $\boldsymbol{t}_{i-1} \cdot \boldsymbol{n}_i = \boldsymbol{t}_{i-1} \times \boldsymbol{t}_i = 2\Delta_{i-1}$, and similarly $\boldsymbol{t}_{i+1} \cdot \boldsymbol{n}_i = \boldsymbol{t}_{i+1} \times \boldsymbol{t}_i = -2\Delta_i$. So

$$\boldsymbol{\psi}_i(\mathbf{x}_{F_i}) \cdot \boldsymbol{n}_i = \lambda_{i-1}(\mathbf{x}_{F_i}) + \lambda_i(\mathbf{x}_{F_i}) = 1$$

by properties B1 and B5 of the λ_i. Furthermore, the scalar product $\boldsymbol{t}_{i+1} \cdot \boldsymbol{n}_{i+1}$ is equal to zero, so

$$\boldsymbol{\psi}_i(\mathbf{x}_{F_{i+1}}) \cdot \boldsymbol{n}_{i+1} = \frac{\boldsymbol{t}_{i-1} \cdot \boldsymbol{n}_{i+1}}{\Delta_{i-1}}\lambda_{i-1}(\mathbf{x}_{F_{i+1}}) = 0,$$

by property B5. Similarly $\boldsymbol{\psi}_i(\mathbf{x}_{F_{i+1}}) \cdot \boldsymbol{n}_j = 0$ for all $j \neq i$. □

Theorem 1 shows that the basis functions form a field that is in $H(\text{div})$. The following Theorem shows that the basis functions $\{\boldsymbol{\psi}_i\}$ reproduce constant vector fields exactly on each cell E.

Theorem 2 *Let $\boldsymbol{e}_k = (\delta_{1k}, \delta_{2k})^t$, $k \in \{1, 2\}$, be the unit vector defined for any $\mathbf{x} \in E$, and let $\boldsymbol{n}_i$ be the outer normal vector associated with the edge F_i and with length equal to the edge length. Then*

$$\boldsymbol{e}_k = \sum_{i=1}^{n} f_{ki}\, \boldsymbol{\psi}_i, \tag{9}$$

with flux $f_{ki} = (\boldsymbol{e}_k \cdot \boldsymbol{n}_i)$.

Proof. Substituting the edge basis functions defined in (7), for the right hand side we have

$$\begin{aligned}
\sum_{i=1}^{n} f_{ki}\, \boldsymbol{\psi}_i &= \sum_{i=1}^{n} \frac{(\boldsymbol{e}_k \cdot \boldsymbol{n}_i)\, \boldsymbol{t}_{i-1}}{2\Delta_{i-1}} \lambda_{i-1} - \sum_{i=1}^{n} \frac{(\boldsymbol{e}_k \cdot \boldsymbol{n}_i)\, \boldsymbol{t}_{i+1}}{2\Delta_i} \lambda_i \\
&= \sum_{i=1}^{n} \frac{(\boldsymbol{e}_k \cdot \boldsymbol{n}_{i+1})\, \boldsymbol{t}_{i}}{2\Delta_{i}} \lambda_{i} - \sum_{i=1}^{n} \frac{(\boldsymbol{e}_k \cdot \boldsymbol{n}_i)\, \boldsymbol{t}_{i+1}}{2\Delta_i} \lambda_i \\
&= \sum_{i=1}^{n} \left[\, (\boldsymbol{e}_k \cdot \boldsymbol{n}_{i+1})\, \boldsymbol{t}_i - (\boldsymbol{e}_k \cdot \boldsymbol{n}_i)\, \boldsymbol{t}_{i+1} \,\right] \frac{\lambda_i}{2\Delta_i}.
\end{aligned}$$

For $\boldsymbol{e}_1 = (1, 0)^t$ and $\boldsymbol{n}_i = ((n_i)_x, (n_i)_y)^t = ((t_i)_y, -(t_i)_x)^t$

$$(\boldsymbol{e}_1 \cdot \boldsymbol{n}_{i+1})\, \boldsymbol{t}_i - (\boldsymbol{e}_1 \cdot \boldsymbol{n}_i)\, \boldsymbol{t}_{i+1} = \begin{pmatrix} t^2_{i+1} t^1_i - t^2_i t^1_{i+1} \\ t^2_{i+1} t^2_i - t^2_i t^2_{i+1} \end{pmatrix} = (\boldsymbol{t}_i \times \boldsymbol{t}_{i+1})\, \boldsymbol{e}_1 = 2\Delta_i \boldsymbol{e}_1.$$

A similar result is obtained for $\boldsymbol{e}_2$. Summing and using property B1,

$$\sum_{i=1}^{n} f_{ki}\, \boldsymbol{\psi}_i = \sum_{i=1}^{n} 2\Delta_i\, \boldsymbol{e}_k \, \frac{\lambda_i}{2\Delta_i} = \boldsymbol{e}_k \sum_{i=1}^{n} \lambda_i = \boldsymbol{e}_k.$$

□

The Corner Velocity Interpolation (CVI) [12] uses vertex velocities $\mathbf{v}_{cvi}(\mathbf{c}_j)$ that are interpolated by bi/tri-linear shape functions λ_j that have been defined on the unit square/cube:

$$\mathbf{v}_{cvi}(\mathbf{x})|_E = \sum_j \mathbf{v}_{cvi}(\mathbf{c}_j) \lambda_j(\mathbf{x}). \tag{10}$$

The functions λ_j are inverse bi/tri-linear mappings defined by (2) (for the 2D case). As already noted, these are generalized barycentric coordinates, and a special case of the Wachspress family of coordinates.

The vertex velocities $\mathbf{v}_{cvi}(\mathbf{c}_j)$ are obtained from the adjacent fluxes f_j and f_{j+1} by solving a 2×2 matrix with $\mathbf{v}(\mathbf{c}_j) \cdot \mathbf{n}_j = f_j$ and $\mathbf{v}(\mathbf{c}_j) \cdot \mathbf{n}_{j+1} = f_{j+1}$ for each vertex. In 3D this becomes a 3×3 matrix.

Lemma 3 *On a quadrilateral, the interpolated velocity field found from the basis $\boldsymbol{\psi}$ using equation (8) with λ_i being the inverse bilinear coordinates of (2), coincides with the CVI velocity interpolation.*

Proof. Assume we have a quadrilateral with edge fluxes f_j for $j = 1, \dots, 4$. We start from equation (8) and show that the interpolated velocity has the same form as (10).

$$\begin{aligned}\mathbf{v}(\mathbf{x}) &= \sum_1^4 f_i \boldsymbol{\psi}_i(\mathbf{x}) = \sum_1^4 f_i \frac{\boldsymbol{t}_{i-1}}{2\Delta_{i-1}} \lambda_{i-1}(\mathbf{x}) - f_i \frac{\boldsymbol{t}_{i+1}}{2\Delta_i} \lambda_i(\mathbf{x}) \\ &= \sum_1^4 \frac{f_{i+1}\boldsymbol{t}_i - f_i \boldsymbol{t} i + 1}{2\Delta_i} \lambda_i(\mathbf{x}).\end{aligned}$$

It remains to show that the corner velocities are the same. At the vertex $\mathbf{c}_i$ the CVI vertex velocity is the solution of $\mathbf{v}_{cvi}(\mathbf{c}_i) \cdot \mathbf{n}_i = f_i$ and $\mathbf{v}_{cvi}(\mathbf{c}_i) \cdot \mathbf{n}_{i+1} = f_{i+1}$, which gives

$$\begin{aligned}\mathbf{v}_{cvi}(\mathbf{c}_i) &= (\mathbf{n}_i, \mathbf{n}_{i+1})^{-T} \begin{pmatrix} f_i \\ f_{i+1} \end{pmatrix} = \frac{1}{2\Delta_i}(-\boldsymbol{t}_{i+1}, \boldsymbol{t}_i) \begin{pmatrix} f_i \\ f_{i+1} \end{pmatrix} \\ &= \frac{f_{i+1}\boldsymbol{t}_i - f_i \boldsymbol{t}_{i+1}}{2\Delta_i} \equiv \mathbf{v}(\mathbf{c}_i).\end{aligned}$$

□

3.4 Interpolation of $H(\mathrm{curl})$ raw fields in 2D

Definition 3 *Let $\{\lambda_i\}$ be a set of barycentric functions on a n-polygon. For every edge $F_i \subset \partial E$, define*

$$\boldsymbol{\varphi}_i(\mathbf{x}) = -\frac{\boldsymbol{n}_{i-1}}{2\Delta_{i-1}} \lambda_{i-1}(\mathbf{x}) + \frac{\boldsymbol{n}_{i+1}}{2\Delta_i} \lambda_i(\mathbf{x}) \tag{11}$$

where $\boldsymbol{n}_j$, $j = 1, \dots, n$, is the outer normal vector to the edge F_j with length equal to the length of F_j.

Theorem 3 *Let $\boldsymbol{t}_j$, $j = 1, \dots, n$, be the tangential vectors to the edge F_j of an element, directed from F_{j-1} to F_{j+1}, with length equal to the length of F_j. Let $\mathbf{x}_{F_j} \in F_j$. Then*

$$\boldsymbol{\varphi}_i(\mathbf{x}_{F_j}) \cdot \boldsymbol{t}_j = \delta_{ij}.$$

Proof. We have $\boldsymbol{n}_{i-1} \cdot \boldsymbol{t}_i = -\boldsymbol{t}_{i-1} \times \boldsymbol{t}_i = -2\Delta_{i-1}$, while $\boldsymbol{n}_{i+1} \cdot \boldsymbol{t}_i = -\boldsymbol{t}_{i+1} \times \boldsymbol{t}_i = 2\Delta_i$. So

$$\boldsymbol{\varphi}_i(\mathbf{x}_{F_i}) \cdot \boldsymbol{t}_i = \lambda_{i-1}(\mathbf{x}_{F_i}) + \lambda_i(\mathbf{x}_{F_i}),$$

which equals 1 along edge F_i by properties B1 and B5 of the λ. As $\boldsymbol{n}_{i+1} \cdot \boldsymbol{t}_{i+1} = 0$ we obtain

$$\boldsymbol{\varphi}_i(\mathbf{x}_{F_{i+1}}) \cdot \boldsymbol{t}_{i+1} = -\frac{\boldsymbol{n}_{i-1} \cdot \boldsymbol{t}_{i+1}}{2\Delta_{i-1}} \lambda_{i-1}(\mathbf{x}_{F_{i+1}}) = 0,$$

by property B5. Similarly, $\boldsymbol{\varphi}_i(\mathbf{x}_{F_j}) \cdot \boldsymbol{t}_j = 0$ for all $j \neq i$. □

Theorem 4 *Let $\boldsymbol{e}_k = (\delta_{1k}, \delta_{2,k})^t$, $k \in \{1, 2\}$, be the unit vector defined for any $\mathbf{x} \in E$. Then*

$$\boldsymbol{e}_k = \sum_{i=1}^n (\boldsymbol{e}_k \cdot \boldsymbol{t}_i)\, \boldsymbol{\varphi}_i. \tag{12}$$

Proof. For the right hand side, substituting the definition (11), we have

$$\begin{aligned}
\sum_{i=1}^{n}(\boldsymbol{e}_k\cdot\boldsymbol{t}_i)\boldsymbol{\varphi}_i &= \sum_{i=1}^{n}-\frac{(\boldsymbol{e}_k\cdot\boldsymbol{t}_i)\,\boldsymbol{n}_{i-1}}{2\Delta_{i-1}}\lambda_{i-1}+\sum_{i=1}^{n}\frac{(\boldsymbol{e}_k\cdot\boldsymbol{t}_i)\,\boldsymbol{n}_{i+1}}{2\Delta_i}\lambda_i\\
&= \sum_{i=1}^{n}-\frac{(\boldsymbol{e}_k\cdot\boldsymbol{t}_{i+1})\,\boldsymbol{n}_{i}}{2\Delta_{i}}\lambda_{i}+\sum_{i=1}^{n}\frac{(\boldsymbol{e}_k\cdot\boldsymbol{t}_i)\,\boldsymbol{n}_{i+1}}{2\Delta_i}\lambda_i\\
&= \sum_{i=1}^{n}-\left[\,(\boldsymbol{e}_k\cdot\boldsymbol{t}_{i+1})\,\boldsymbol{n}_i-(\boldsymbol{e}_k\cdot\boldsymbol{t}_i)\,\boldsymbol{n}_{i+1}\,\right]\frac{\lambda_i}{2\Delta_i}.
\end{aligned}$$

For $\boldsymbol{e}_1=(1,0)^t$ and $\boldsymbol{n}_i=((n_i)_x,(n_i)_y)^t=((t_i)_y,-(t_i)_x)^t$

$$(\boldsymbol{e}_1\cdot\boldsymbol{t}_{i+1})\,\boldsymbol{n}_i-(\boldsymbol{e}_1\cdot\boldsymbol{t}_i)\,\boldsymbol{n}_{i+1}=\begin{pmatrix} t^1_{i+1}t^2_i-t^1_it^2_{i+1}\\ -t^1_{i+1}t^1_i+t^1_it^1_{i+1}\end{pmatrix}=-2\Delta_i\boldsymbol{e}_1.$$

A similar result is obtained for $\boldsymbol{e}_2$. Summing and using property B1,

$$\sum_{i=1}^{n}(\boldsymbol{e}_k\cdot\boldsymbol{t}_i)\boldsymbol{\varphi}_i=\sum_{i=1}^{n}2\Delta_i\,\boldsymbol{e}_k\,\frac{\lambda_i}{2\Delta_i}=\boldsymbol{e}_k\sum_{i=1}^{n}\lambda_i=\boldsymbol{e}_k.$$

□

4 $H(\mathrm{div})$ interpolation in 3D

In [20] Nordbotten and Hægland prove that for general hexahedra in 3D with bilinear faces, it is not possible to obtain a velocity interpolation that simultaneously 1) is a local reconstruction of velocity based on the six face fluxes, 2) reproduces uniform flow and 3) lies in $H(\mathrm{div})$.

For cells with plane faces this problem does not present itself, and we can satisfy all three properties simultaneously. With the additional assumption that for each cell and vertex, no more than three of the cell's faces meet at the vertex, the $H(\mathrm{div})$ interpolation generalize straight away from 2D to 3D. Polytopes with this property are called simple polytopes. As before, $\{\lambda_j(\mathbf{x})\}$ will indicate the (3D) barycentric coordinates of the point $\mathbf{x}$, and the functions λ_i satisfy B1-B5.

In our implementation we have used the coordinates described in [26], which also require simple polytopes.

Definition 4 *Let $F_i\subset\partial E$ be a plane polygonal face with normal $\boldsymbol{n}_i$, the length of $\boldsymbol{n}_i$ being equal to the area $|F_i|$. Let exacly three edges meet in each vertex $\mathbf{x}_j$ of F_i, where two edges lie in the plane of F_i and the third edge is indicated by $\boldsymbol{\nu}^i_j$. Then we define the face based basis functions*

$$\boldsymbol{\psi}_i(\mathbf{x})=\sum_j\frac{\boldsymbol{\nu}^i_j}{\boldsymbol{\nu}^i_j\cdot\boldsymbol{n}_i}\lambda_j(\mathbf{x}). \qquad (13)$$

The face based basis functions have the same properties as those defined in 2D:

Theorem 5 *Let $\boldsymbol{n}_k$ be the outer normal vector to the face F_k and with length equal to the area $|F_k|$. Let $\mathbf{x}_{F_k}\in F_k$. Then*

$$\boldsymbol{\psi}_i(\mathbf{x}_{F_k})\cdot\boldsymbol{n}_k=\delta_{ik}.$$

Proof. The proof follows easily by considering three different cases: (i) either the normal n_k is the normal n_i ($i = k$), or (ii) the normal n_k is normal to a plane of which $\boldsymbol{\nu}_j^i$ is an edge ($i \neq k$) or (iii) the normal $n_k \neq n_i$ is the normal to a face which does not contain $\boldsymbol{\nu}_j^i$ ($i \neq k$). If (i) we have that $i = k$. The multiplication simplifies and we have the summation of the barycentric coordinates of the point $\mathbf{x}_{F_k}$ which gives 1 by B1. If (ii), the scalar product between $\boldsymbol{\nu}_j^i$ and n_k is zero. If we have (iii) the barycentric coordinate function $\lambda_j(\mathbf{x}_{F_k})$ is equal to zero by B5. This concludes the proof. □

Theorem 6 *Let $\boldsymbol{v}$ be a constant vector and let $\boldsymbol{n}_i$ be the outer normal vector associated with the face F_i and with length equal to the area $|F_i|$. Then*

$$\boldsymbol{v} = \sum_i (\boldsymbol{v} \cdot \boldsymbol{n}_i)\, \boldsymbol{\psi}_i. \tag{14}$$

Proof. Let $\mathcal{C}_i$ be the set of corners of face i, and $\mathcal{F}_j$ the set of faces of the corner j. By our assumptions, $\sharp\mathcal{F}_j = 3$, where $\sharp$ indicates the number of elements in the set. Examining the right hand side of (14) we obtain

$$\begin{aligned}\sum_i (\boldsymbol{v} \cdot \boldsymbol{n}_i)\, \boldsymbol{\psi}_i &= \sum_i (\boldsymbol{v} \cdot \boldsymbol{n}_i) \sum_{j \in \mathcal{C}_i} \frac{\boldsymbol{\nu}_j^i}{\boldsymbol{\nu}_j^i \cdot \boldsymbol{n}_i} \lambda_j(\mathbf{x}) \\ &= \sum_j \left(\sum_{i \in \mathcal{F}_j} \frac{\boldsymbol{v} \cdot \boldsymbol{n}_i}{\boldsymbol{\nu}_j^i \cdot \boldsymbol{n}_i} \boldsymbol{\nu}_j^i \right) \lambda_j(\mathbf{x}) = \sum_j \boldsymbol{r}_j \lambda_j.\end{aligned}$$

Now for any corner j, consider the product $\boldsymbol{r}_j \cdot \boldsymbol{n}_k$, $k \in L_j$:

$$\boldsymbol{r}_j \cdot \boldsymbol{n}_k = \sum_{i \in \mathcal{F}_j} \frac{\boldsymbol{\nu}_j^i \cdot \boldsymbol{n}_k}{\boldsymbol{\nu}_j^i \cdot \boldsymbol{n}_i} \boldsymbol{v} \cdot \boldsymbol{n}_i.$$

In the above sum, for $i = k$ we get $\boldsymbol{v} \cdot \boldsymbol{n}_k$. Since the sum is taken over only the set of faces of the corner j, for $i \neq k$ the vector $\boldsymbol{\nu}_j^i$ lies in the plane of F_k, and the term is therefore zero. Since $\boldsymbol{r}_j \cdot \boldsymbol{n}_k = \boldsymbol{v} \cdot \boldsymbol{n}_k$ for three linearly independent $\boldsymbol{n}_k$ (by the $\sharp\mathcal{F}_j = 3$ assumption and convexity) $\boldsymbol{r}_j = \boldsymbol{v}$ and the theorem follows. □

In 3D the equation may not always have a solution, even for strictly convex E, since the $\sharp\mathcal{F}_j = 3$ assumption (simple polytopes) does not hold in general. However, many types of grids do have this property. Example cell shapes include hexahedra, simplices, arbitrary prisms and even dodecahedra. On the other hand pyramids (one corner has four faces), octahedra (all corners have four) and icosahedra (all corners have five) do not. Pebi grids in 3D and 2.5D (extruded) will in general work well with the method. We prove the assertion for 3D Voronoi diagrams under a uniqueness assumption.

Lemma 4 *If there are no five cospherical Voronoi sites, corresponding to uniqueness of the dual Delaunay tetrahedral grid, for any Voronoi cell each of its corners will have exactly three adjacent faces belonging to that cell.*

Proof. Let E be the cell generated by the Voronoi site $\mathbf{x}_E$. Assume that there is a corner with four or more adjacent faces. On each such face, all its points, including the corner itself, are equidistant between $\mathbf{x}_E$ and one other site. So the corner must be equidistant to $\mathbf{x}_E$ and at least four more sites. Then those (at least) five sites are cospherical, which contradicts our assumption. □

5 Streamlines

Streamline methods for reservoir simulation have been studied extensively. For a comprehensive overview see the book by Datta-Gupta and King, [8]. Streamline tracing is a critical part of any such method, and consists of solving the streamline ODE on some computational grid:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = \mathbf{v}(\mathbf{x}), \tag{15}$$

where $\mathbf{x}$ is a point in space and $\mathbf{v}$ is the velocity field. The parameter τ is known as the streamline "time-of-flight".

For our purposes the velocity field is considered fixed in time. Consequently, the streamlines will not correspond to path lines if the actual velocity field $\mathbf{v}$ evolves in time. In streamline-based simulation an operator-splitting approach is used, and the velocity field will be recomputed (relatively) rarely, the underlying assumption being that it changes only slowly. This gives rise to an operator-splitting error. This error may be reduced by recomputing the velocity field more often, but that may be computationally expensive. Also, after each such update all streamlines must be recomputed.

Typically, a flow solver provides a discrete velocity or flux field which is then interpolated to obtain an approximation of $\mathbf{v}$ in (15).

In Pollock's method the velocity field interpolation is given on the unit square by linear interpolation in each component separately. The method leads to a discoupling of the components of the streamline equation (15) and the solution can be found analytically. Variants by Cordes and Kinzelbach [7], Prevost et al. [24] and Jimenez et al. [14] are often used for grids with general hexahedral cells.

In [12], Hægeland etc. describe the Corner Velocity Interpolation (CVI) on quadrilateral and hexahedral grids. This is a streamline tracing method aiming to produce streamlines that are more accurate than those produced by the methods in [23], [7], [24] and [14]. The CVI tracing can be done both in the referance cell and in the pysical cell, cf. [13]. Juanes and Matringe [15] examine streamline tracing using the general order Raviart-Thomas (RT_k) and Brezzi-Douglas-Marini (BDM_k) velocity fields and the stream function defined on the corresponding pointwise divergence free subspace. The stream function is defined on the reference element and is mapped to to the physical cell according to the Piola mapping. Note that the time of flight have to be adjusted by the Jacobian to get the same streamlines you would else get on the quadrilateral RT_k or BDM_k fullfield. We refer to Section 2.3 for a discussion of the Piola mapping. Zhang et al. [27] present an extension of streamline tracing to polygonal cells, also with focus on pointwise divergence free fields in reference cells, and they discuss the Juanes and Matringe method as well as the CVI method.

We aim at streamline tracing directly on the physical space, where we ensure reproduction of uniform flow. We do not involve the Piola mapping. The streamline tracing presented here is an extension in physical space of the CVI interpolation, and we denote it epCVI. In our case, we assume that a discrete flux field is given in terms of a flux on each grid face which is then interpolated using (8). For simplicity, we will assume homogeneous porosity per grid cell, which simply scales the resulting cell time-of-flight, and leave it out of the equations. The equations of (15) are here coupled, which means that we need to solve the system by numerical means.

5.1 Streamline tracing on simplicial grids

For the special case of incompressible flow on simplicial grids, that is, triangulations in 2D and tetrahedral grids in 3D, there exists a cell-wise constant velocity field that is consistent with the face fluxes. In n dimensions a constant velocity has n degrees of freedom, and on a simplex we have $n+1$ faces at which we need to satisfy consistency. However, incompressibility implies that the sum of the fluxes is equal to zero (away from wells), which reduces the number of independent consistency equations by one. The constant velocity field for a cell is given by equation (6).

Tracing a streamline through the element E may now be done by following this procedure:

1. Find the barycentric coordinates $\mathbf{x}_0 = \mathbf{x}(\tau_0)$ of the entry point with respect to E. If the entry point is given in barycentric coordinates for the intercell boundary subsimplex (a triangle in 3D, a line in 2D), this step consists of a simple shuffling of coordinates.

2. Compute the velocity $\mathbf{v}$ in barycentric coordinates using equation (6).

3. Equation (15) has the solution $\mathbf{x}(\tau) = \mathbf{x}_0 + (\tau - \tau_0) \cdot \mathbf{v}$. Here τ_0 is the entry time-of-flight. Find the exit point and time-of-flight as follows:

   (a) For all i such that $v_i < 0$, compute the intersection times $\tau_i = -x_i(\tau_0)/v_i$. For all other i, set $\tau_i = -\infty$.

   (b) The minimum non-negative intersection time τ_m is the time-of-flight through E. The index m at which this minimum is attained indicates the boundary subsimplex that was hit. If there are multiple candidates, choose the one for which the corresponding subsimplex outward flux is greatest.

   (c) The exit point is given in barycentric coordinates with respect to E by $\mathbf{x}(\tau_m) = \mathbf{x}_0 + \tau_m \cdot \mathbf{v}$.

4. Transform $\mathbf{x}(\tau_m)$ to barycentric coordinates on the intercell boundary subsimplex by removing the coordinate for which $\tau_m = \tau_i$ (which must be zero).

5. E now becomes the neighboring simplex that is adjacent through subsimplex m. Repeat the procedure.

The above procedure is robust. It cannot be trapped on an edge or vertex, and will always exit the element E through an outflow face. However when using floating point numerics, it is prudent to add a rescaling of the coordinates after step 4, to ensure that they sum to one. This is because the result of the computation $\mathbf{x}(\tau_m) = \mathbf{x}_0 + \tau_m \cdot \mathbf{v}$ above may not sum exactly to one when using floating point.

5.2 Streamline tracing on arbitrary convex polytopes

The procedure is as follows:

1. For each cell E, pre-compute the barycentric basis functions.

2. Upon a streamline reaching a cell boundary entry point, integrate it with the interpolated velocity function given by (8) until reaching the cell boundary again. That is the exit point, which will be the entry point for the next segment of the streamline, traced through the neighbouring cell.

In order to actually trace a streamline through E, we must use some numerical ODE solver, and this is the primary drawback of the epCVI method, compared to semi-analytic methods.

Assuming that tracing starts at a point $\mathbf{x}_0$ on some inflow boundary face of the cell, we must trace forwards by solving (15) numerically. A simple scheme like Euler's method may or may not be sufficient, depending on accuracy requirements and the application. We have used the second-order Heun's method, which is an explicit second-order method also known as the Euler-Trapezoidal predictor-corrector method. The scheme is given by

$$\begin{aligned} \mathbf{x}_{n+1}^{p} &= \mathbf{x}_n + h\mathbf{v}(\mathbf{x}_n) \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{2}\left(\mathbf{v}(\mathbf{x}_n) + \mathbf{v}(\mathbf{x}_{n+1}^{p})\right) \end{aligned}$$

Choosing the step-size h can be challenging. An initial step-size may be chosen proportional to some characteristic length divided by a characteristic velocity. One can for instance use the square root (cubic root in 3D) of the cell volume divided by a cell average velocity, although this is insufficient for anisotropic cell shapes.

The difference between $\mathbf{x}_{n+1}^{p}$ and $\mathbf{x}_{n+1}$ may be used to control the error, exploiting the fact that Heun's method is a second-order method with an embedded first-order method. Similar embedded method pairs of higher degree such as the Runge-Kutta-Fehlberg 4(5) pair or the Dormand-Prince 5(4) pair may be used for increased accuracy.

Another difficulty may be to detect the point at which the streamline exits the cell. Given the generalized barycentric coordinates $\{\lambda_i\}$ for a position it is easy to check this, since a point lies inside the cell if and only if all coordinates are nonnegative. We need to compute these coordinates for the purpose of calculating $\mathbf{v}$, so there is no extra computational cost.

Once we know that the integration has passed outside the cell, we need to find the point of intersection with the cell boundary. A simple approach is to use a function ϕ that is defined on the line between $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$. The indicator function ϕ should be negative on the outside and positive on the inside of E. For example one may use $\phi(\mathbf{x}) = \min_i \lambda_i(\mathbf{x})$. Then one can use a nonlinear equation solver for bracketed zeros to find the intersection, for example one of the modified regula falsi solvers described by [10] or the method of [4]. The regula falsi variants are easier to implement correctly, but Brent's method may give faster convergence. The latter approach only gives first order accuracy at the intersection point, however. If higher accuracy is needed one should use a continuous Runge-Kutta method to define the curve between $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$ along which the zero of ϕ is sought, see for example [21]. For order up to three, Hermite interpolation of the solution curve will yield sufficient accuracy.

6 Numerical examples

In the following numerical tests we show that the velocity interpolation proposed leads to smooth streamlines, comparable to a higher order method as seen in for instance in article [15]. In particular, our first test case shows a homogeneous quarter-five-spot case which was also used for the tests used to compare streamline tracing with RT_0 and BDM_1 in [15]. The streamlines obtained with the extended physical space CVI (epCVI) are smooth, as those obtained with BDM_1, as can be seen in Figure 5. As the grid is Cartesian, and we have employed Wacshpress Coordinates, the epCVI method reduces to the regular CVI method. We also note the smoothness in the variation of time of flight as can be seen to the right in Figure 5. When tracing from a source to a sink, we choose to terminate the streamlines on the source/sink cell boundaries. This is a simple approach, but may not yield sufficiently accurate time-of-flight, especially on coarse grids with many wells. For such cases a more accurate model of the near-well region is needed. We do not consider this issue within the scope of this paper.

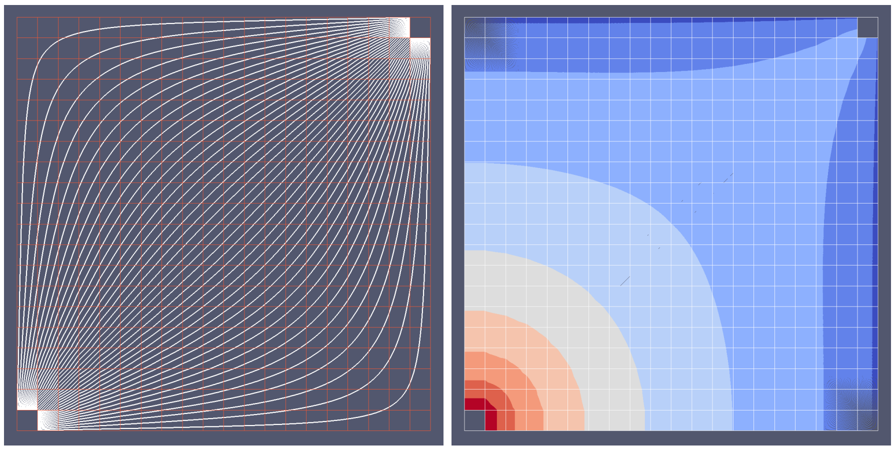

Figure 5: Streamlines for homogenous quarter-five-spot case with 20×20 cells. Left: Sparse streamlines. Right: Dense streamlines that have been coloured according to logarithmic time-of-flight with a discrete set of colours to show isocontours.

In the following two test cases we look at streamline tracing on polygonal grids. The original CVI method has not been defined for such a grid. To apply the tracing using RT_0 or BDM_1 velocity fields, the grid would have to be refined so as to consist of either quadrilaterals or triangles, which we do not consider here. As our interpolation is constructed such as to reproduce uniform flow, our first test case on polygonal grids shows that this is indeed the case. The left hand side of Figure 6 shows streamlines that are straight lines and parallel, as expected. The right hand side of Figure 6 shows streamlines for the case where we have injection in a cell in the lower central part of the domain, while the producer is placed towards the top central part of the domain. We see that the produced streamlines are smooth and quite robust to grid effects. Near the source and sink there is some distortion, but that can mostly be attributed to the coarse spatial resolution and its effect on the accuracy of the flow solver. The injector cell is of an irregular shape and is not symmetrical, which leads to the non-symmetry also in the velocity field.

The time-of-fligth contours as well as the effect of grid refinement can be examined in Figure 7. While the contours are quite smooth even for the coarser grid, the smoothness and the symmetry are improved on the fine grid as the impact of the shape of the injector diminishes. If we do a close up of the area around the injection cell, Figure 8, we see that the contours are distorted into the shape of the injector, but then become smoother as they advance.

All the tests so far have been on a grid cells that are, though distorted, more or less isotropic. We have also tested the effect of using a grid with anisotropic cell shapes, as seen in Figure 9. The grid consists of 18711 hexagonal cells, compressed in the y-direction to obtain a 10:1 anisotropy ratio. A mimetic pressure solver has been used to obtain the discrete fluxes (raw field). It can be seen that in spite of the strongly distorted velocity field near the source, the time-of-flight behaves well on the larger scale.

Finally we test our streamline tracing on a 3D test case, with the results shown in Figure 10 and Figure 11. The test case is similar to the 2D test case with an injector and a producer placed on opposite sides of the domain, close to the symmetry axis. The left hand side of Figure 10 shows a sparse set of streamlines. The grid, shown on the right in the same Figure, is a 5-layer extrusion of a 2D voronoi cell grid. The 2D grid is

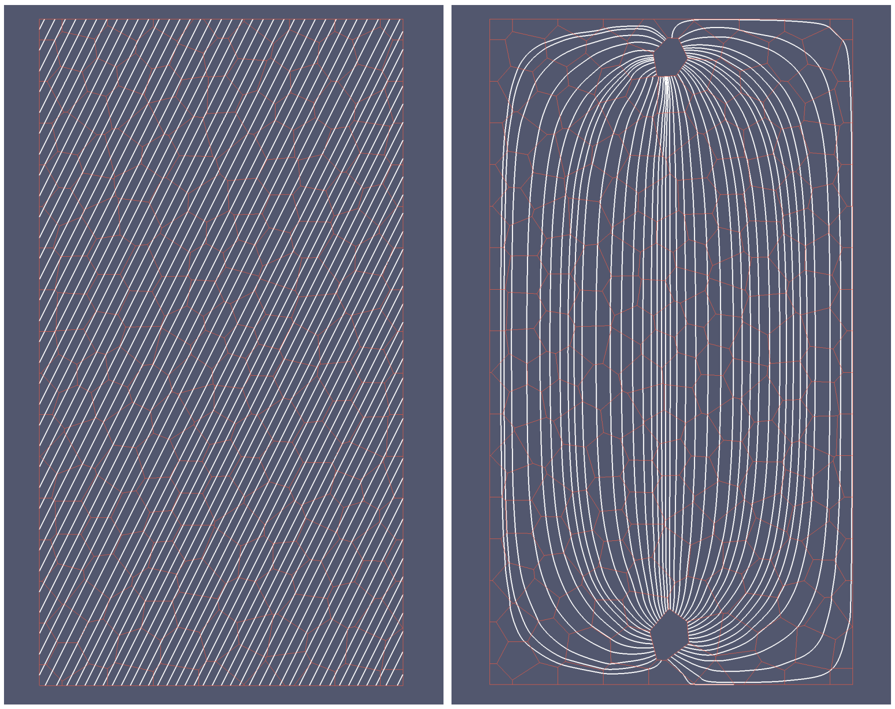
Figure 6: Sparse streamlines for a homogenous polygonal grid case with 232 cells. Left: uniform flow. Right: streamlines traced from injector cell to producer cell.

the dual of a triangulation created with DistMesh [22]. To the left in Figure 11 it can again be seen that the time-of-flight contours become quite smooth as they advance. The somewhat uneven distribution of the streamlines is an artifact of the very simple seed point distribution algorithm used, which is generating random barycentric coordinate tuples for each face of the source cell. To the right in Figure 11 we demonstrate that uniform flow is also preserved in 3D.

7 Conclusions

A new method has been described for direct streamline tracing on simple polytope cells. The method extends the CVI method of [12, 13] in the physical space. Numerical testing shows quite good performance of the method, with smoothness of the streamlines and time-of-flight contours in addition to robustness with respect to grid effects.

We introduced the differentiation between full field methods, which have element basis functions, and raw field methods that define a velocity field only at the element interfaces. We argue that the element basis functions used in full field methods often have shortcomings that can be overcome by instead using a raw field method and adopt a suitable interpolation successively. We present new interpolation basis functions for $H(\text{div})$ and $H(\text{curl})$ which reproduce constant vector fields on general meshes.

Some of the issues that may be investigated in future work include relaxation of convexity and topological requirements and sensitivity of interpolation with respect to cell shape.

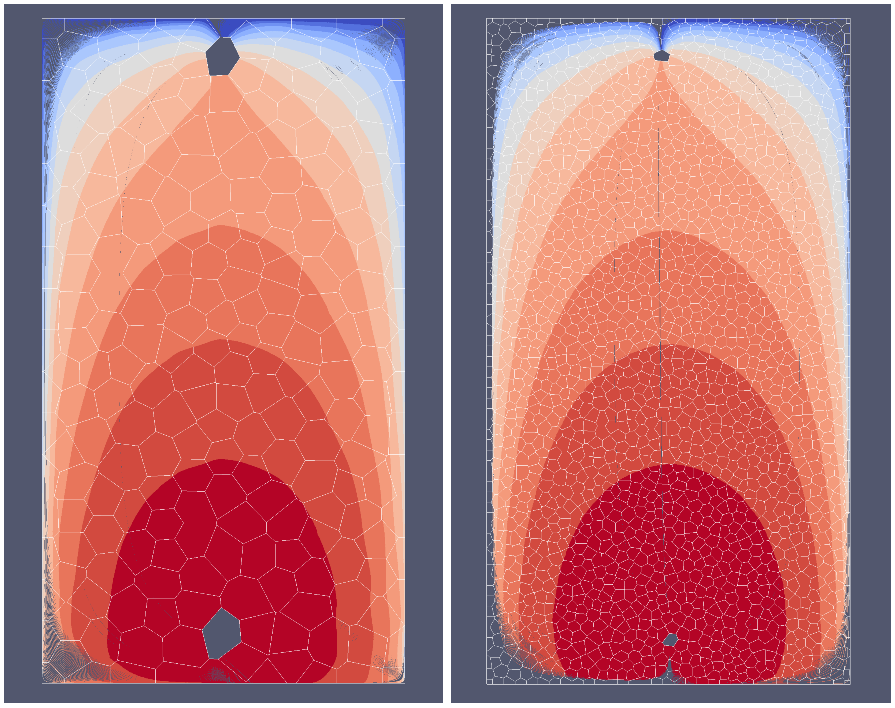

Figure 7: Dense streamlines for injection in homogenous polygonal grids. Streamlines have been coloured according to time-of-flight with a discrete set of colours to show isocontours. Left: 232 cells. Right: 2079 cells.

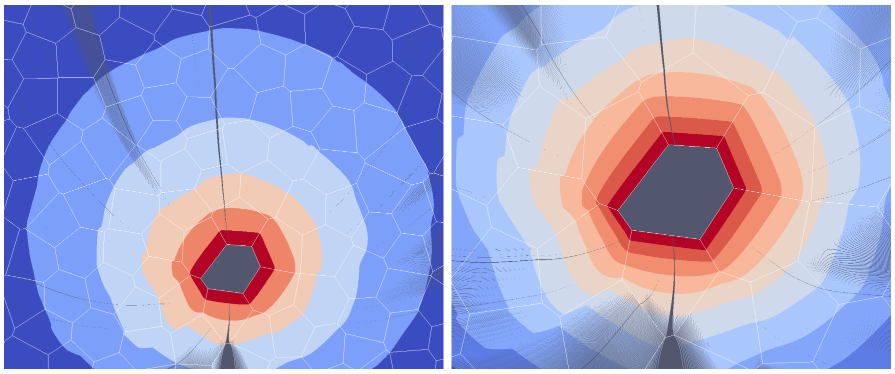

Figure 8: Closeups near source of the 2079 cell case. Coloured according to logarithmic time-of-flight.

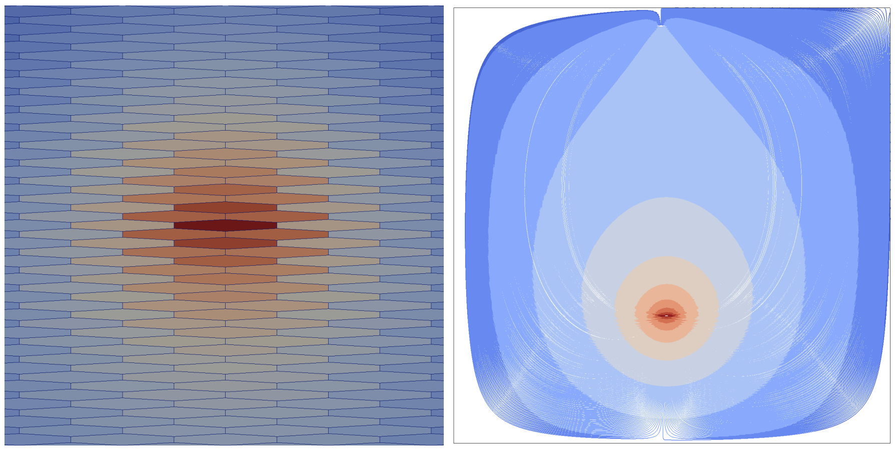

Figure 9: Left: closeup of pressure field near source of anisotropic case. Right: dense set of streamlines from anisotropic case, coloured according to logarithmic time-of-flight

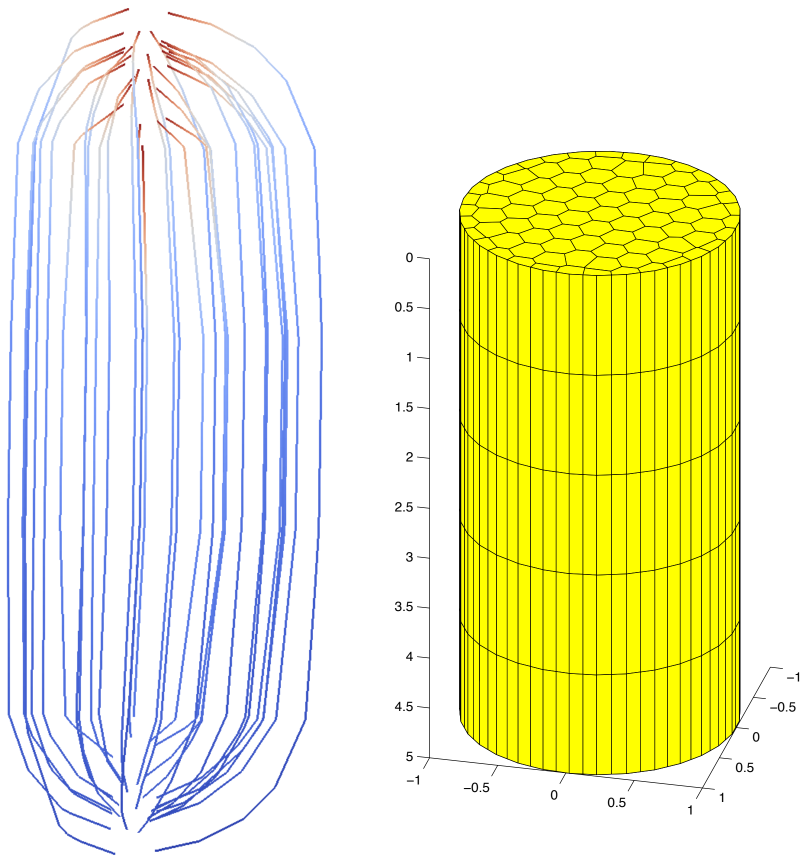

Figure 10: Left: sparse set of streamlines from 3d case, coloured according to logarithmic time-of-flight. Right: extruded voronoi grid used for 3d test.

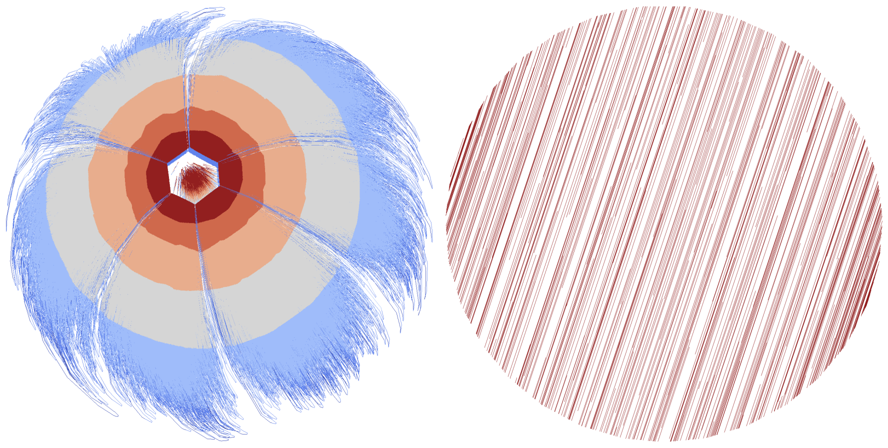

Figure 11: Left: denser set of streamlines from 3d case, coloured according to logarithmic time-of-flight. Right: streamlines from 3d case, uniform flow, from above.

A Proof of Lemma 1

In [11], the following error estimate with generalized barycentric coordinates interpolation I on polyhedra is established for Wachspress and Sibson coordinates, given sufficient geometric restrictions on the domain E;

$$\|u - Iu\|_{H^1(E)} \leq Ch|u|_{H^2(E)}, \qquad \text{for all } u \in H^2(E). \tag{16}$$

If we interpolate $\Pi_h \boldsymbol{u}_h$ component-wise with Wachspress coordinates from vertex values reconstructed from the flux, Lemma 3 shows that this gives the reconstructed barycentric coordinate field $\mathcal{R}_{\text{BCI}}\boldsymbol{u}_{F,h}$. From (16) we can also deduce that

$$\|\boldsymbol{u} - \mathcal{R}_{\text{BCI}}\Pi_h \boldsymbol{u}\|_{H^1(E)} \leq Ch|\boldsymbol{u}|_{H^2(E)}, \qquad \text{for all } u \in H^2(E)$$

under the assumption of constant values of $\boldsymbol{u}\cdot\mathbf{n}$ on ∂E. Since we know that the raw field, $\Pi_h \boldsymbol{u}_{h,RT_0}$, will converge in a discrete $\mathcal{L}_2$ norm, it follows that also

$$\begin{aligned}\|\boldsymbol{u} - \mathcal{R}_{\text{BCI}}\Pi_h \boldsymbol{u}_{h,RT_o}\|_{H(\text{div})} &\leq \|\boldsymbol{u} - \mathcal{R}_{\text{BCI}}\Pi_h \boldsymbol{u}\|_{H(\text{div})} + \|\mathcal{R}_{\text{BCI}}\Pi_h(\boldsymbol{u} - \boldsymbol{u}_{h,RT_o})\|_{H(\text{div})}\\ &\leq \|\boldsymbol{u} - \mathcal{R}_{\text{BCI}}\Pi_h \boldsymbol{u}\|_{H(\text{div})} + |\mathcal{R}_{\text{BCI}}|\|\Pi_h(\boldsymbol{u} - \boldsymbol{u}_{h,RT_o})\|_{\mathcal{L}_2,h}\\ &\leq Ch|\boldsymbol{u}|_{H^2}, \qquad \text{for all } u \in H^2.\end{aligned}$$

References

[1] I. Aavatsmark, T. Barkve, Ø. Bøe, and T. Mannseth. Discretization on unstructured grids for inhomogeneous, anisotropic media. I. Derivation of the methods. *SIAM J. Sci. Comput.*, 19(5):1700–1716, 1998.

[2] I. Aavatsmark, G. T. Eigestad, R. A. Klausen, M. F. Wheeler, and I. Yotov. Convergence of a symmetric MPFA method on quadrilateral grids. *Comput. Geosci.*, 11(4):333–345, 2007.

[3] D. N. Arnold, D. Boffi, and R. S. Falk. Quadrilateral H(div) finite elements. *SIAM J. Numer. Anal.*, 42(6):2429–2451, 2005.

[4] R. P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, 1973.

[5] F. Brezzi and M. Fortin. *Mixed and Hybrid finite element methods.* Springer series in computational mathematics. Springer-Verlag, 15 edition, 1991.

[6] F. Brezzi, K. Lipnikov, and M. Shashkov. Convergence of the mimetic finite difference method for diffusion problems on polyhedral meshes. *SIAM J. Numer. Anal.*, 43(5):1872–1896, 2005.

[7] C. Cordes and W. Kinzelbach. Continuous groundwater velocity fields and path lines in linear, bilinear and trilinear finite elements. *Water Resources Research*, 28(11):2903–2911, 1992.

[8] A. Datta-Gupta and M. J. King. *Streamline Simulation: Theory and Practice*, volume 11 of *SPE Textbook Series.* Society of Petroleum Engineers, 2007.

[9] M. S. Floater. Mean value coordinates. *Comp. Aided Geom. Design*, 20:19–27, 2003.

[10] J. A. Ford. Improved algorithms of illinois-type for the numerical solution of nonlinear equations. Technical Report CSM-257, University of Essex, 1995.

[11] A. Gillette, A. Rand, and C. Bajaj. Error estimates for generalized barycentric interpolation. Technical Report arXiv:1010.5005, University of Texas at Austin, 2010.

[12] H. Hægland, H. K. Dahle, G. T. Eigestad, K.-A. Lie, and I. Aavatsmark. Improved streamlines and time-of-flight for streamline simulation on irregular grids. *Adv. Water Resour.*, 30(4):1027–1045, 2007.

[13] H. Hægland, H. K. Dahle, K.-A. Lie, and G. Eigestad. Adaptive streamline tracing for streamline simulation on irregular grids. In P. Binning, P. Engesgaard, H. Dahle, G. Pinder, and W. Gray, editors, *Proceedings of the XVI International Conference on Computational Methods in Water Resources*, Copenhagen, Denmark, 18–22 June 2006.

[14] E. Jimenez, K. Sabir, A. Datta-Gupta, and M. King. Spatial error and convergence in streamline simulation. In *SPE Reservoir Simulation Symposium, Houston, TX, January 31-February 2, 2005*, SPE 92873, 2005.

[15] R. Juanes and S. F. Matringe. Unified formulation for high-order streamline tracing on two-dimensional unstructured grids. *J. Sci. Comput.*, 38(1):50–73, 2009.

[16] R. Klausen, S. Mundal, and H. Dahle. Barycentric coordinate based mixed finite elements on quadrilateral/hexahedral mesh. *Inter. J. of Num. Analysis and Modeling*, 8(4):584–598, 2011.

[17] R. A. Klausen and R. Winther. Convergence of multipoint flux approximations on quadrilateral grids. *Numer. Methods Partial Differential Equations*, 22(6):1438–1454, 2006.

[18] R. A. Klausen and R. Winther. Robust convergence of multi point flux approximation on rough grids. *Numer. Math.*, 104(3):317–337, 2006.

[19] M. Meyer, A. Barr, H. Lee, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

[20] J. M. Nordbotten and H. Hægland. On reproducing uniform flow exactly on general hexahedral cells using one degree of freedom per surface. *Advances in Water Resources*, 32(2):264–267, 2009.

[21] B. Owren and M. Zennaro. Derivation of efficient continuous explicit runge-kutta methods. *SIAM J. Sci. Stat. Comput*, 13:1488–1501, 1992.

[22] P.-O. Persson and G. Strang. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, 2004.

[23] D. Pollock. Semi-analytical computation of path lines for finite-difference models. *Ground Water*, 26(6):743–750, 1988.

[24] M. Prevost, M. G. Edwards, and M. J. Blunt. Streamline tracing on curvilinear structures and unstructured grids. *SPE Journal*, 7(2):139–148, 2002.

[25] E. L. Wachspress. *A rational finite element basis.* Academic Press, NY, 1975.

[26] J. Warren, S. Schaefer, A. N. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. Technical report, Advances in Computational and Applied Mathematics, 2004.

[27] Y. Zhang, M. King, and A. Datta-Gupta. Robust streamline tracing using intercell fluxes in locally refined and unstructured grids. In *SPE Reservoir Simulation Symposium, 21-23 February 2011, The Woodlands, Texas, USA*, SPE 140695, 2011.