A High Resolution PDE Approach to Quadrilateral Mesh Generation

Julian Marcon\textsuperscript{a}, David A. Kopriva\textsuperscript{b,\ast}, Spencer J. Sherwin\textsuperscript{a}, Joaquim Peiró\textsuperscript{a}

\textsuperscript{a} Imperial College London, South Kensington Campus, London SW7 2AZ, UK
\textsuperscript{b} The Florida State University, Tallahassee, FL 32306, USA and San Diego State University, San Diego, CA 92182, USA

Abstract

We describe a high order technique to generate quadrilateral decompositions and meshes for complex two dimensional domains using spectral elements in a field guided procedure. Inspired by cross field methods, we never actually compute crosses. Instead, we compute a high order accurate guiding field using a continuous Galerkin (CG) or discontinuous Galerkin (DG) spectral element method to solve a Laplace equation for each of the field variables using the open source code \textit{Nektar++}. The spectral method provides spectral convergence and sub-element resolution of the fields. The DG approximation allows meshing of corners that are not multiples of $\pi/2$ in a discretization consistent manner, when needed. The high order field can then be exploited to accurately find irregular points, and accurately integrated using a high order streamline integration method to avoid features like limit cycles. The result is a mesh with naturally curved quadrilateral elements that do not need to be curved \textit{a posteriori} to eliminate invalid elements. The mesh generation procedure is implemented in the open source mesh generation program \textit{NekMesh}.

Keywords:
Cross field; quad meshing; high order; spectral element method; singularity characterization

\ast Corresponding author

Email addresses: julian.marcon14@imperial.ac.uk (Julian Marcon), kopriva@math.fsu.edu (David A. Kopriva), s.sherwin@imperial.ac.uk (Spencer J. Sherwin), j.peiro@imperial.ac.uk (Joaquim Peiró)
1. Introduction

A quadrilateral subdivision of a two dimensional domain into a minimal number of subdomains can serve as the starting point from which to generate meshes for finite element [1], finite volume, (block structured) finite difference [2] and spectral element methods [3, 4]. To generate finite element and finite volume meshes, one would subdivide each subdomain into quadrilateral cells sufficiently small to resolve the geometry and expected solution features. The resulting mesh would have the desired property of only a small number of irregular nodes where the valence, i.e., the number of elements sharing a point, is not equal to four. The subdivision of the subdomains could also be used as nodes for high order block-structured finite difference methods if the subdivision within each block is structured. Finally, the subdomains could be used as is with high order geometry information intact, or subdivided in a structured or unstructured manner, to be used as elements for a quadrilateral based spectral element method.

Fully automatic quadrilateral mesh generation is still difficult [5], especially for high order (curved) elements. Most previous work has focussed on generating large numbers of small straight sided elements. The extension to high order is usually done by curving the elements a posteriori, e.g. [6]. Unstructured quadrilateral mesh generators tend to generate meshes with large numbers of irregular points [7]. Extraneous irregular points are generated even for simple geometries, where mesh topologies can be generated by hand with a minimum number of them. As an example, Fig. 1 shows two meshes generated with a subdivision algorithm [8, 9], which generates more subdivisions than necessary.

One approach for generating quadrilateral subdivisions in arbitrary domains has come out of the computer graphics community. It uses cross fields [1, 5] in a field-guided procedure. Cross fields resemble cross hatching used in drawings. Crosses, which are composed of two direction vectors and their negatives at a point, are invariant to rotations of $\pi/2$. Cross field methods generate meshes with fewer irregular points. Fig. 2 shows the same two geometries as Fig. 1 decomposed with a cross field method. The method produces the same topology as an experienced user might generate by hand, and has a minimum number of irregular points.

Details of cross field procedures vary, but the basic idea is that the crosses are oriented, usually at the nodes of an existing triangular mesh, according to a minimization or smoothing procedure. Triangles in which the crosses can-
Figure 1: Examples of quadrilateral meshes in simple geometries generated by a subdivision algorithm.

Figure 2: Subdivision of the triangle and circle geometries using a cross field approach.
not smoothly vary contain singularities, which serve as the irregular nodes of the quadrilateral decomposition to be computed. The internal edges of the quadrilaterals are found by integrating streamlines (separatices, which are streamlines that start at a singularity) as directed by the crosses \[10\], starting from the singularities until they reach another singularity or a physical boundary. The quadrilateral subdivision is usually further subdivided into smaller quadrilateral elements.

Low order techniques do not accurately locate singular points within the domain. Spurious singularities can be generated, which may be coalesced in an *ad hoc* fashion or retained to produce a mesh with unneeded irregular points. The strength or valence of the singularities computed with low order techniques can lead to the incorrect number of streamlines and hence the incorrect valence of the associated interior quadrilateral mesh node.

Low order tracing of the streamlines also often leads to multiple curves that need to be coalesced. Associated with this is the well-known limit cycle problem where streamlines, instead of meeting as they should, spiral indefinitely \[11\].

Finally, boundary conditions are difficult to apply at corners where a jump discontinuity in the cross field will occur when a standard continuous Galerkin (finite element) approximation is used for smoothing. Rather than putting a singularity there, the corner may be effectively smoothed by averaging the cross field at such points \[11\], at the expense of adding more interior singular points.

We address the issues found in the use of low order techniques by using a high resolution approach to compute a guiding field, accurately locating the interior singularities, and accurately integrating the streamlines used to subdivide the domain. We use a high order continuous or discontinuous Galerkin spectral element method (SEM or DGSEM) on a triangular mesh, the approximation depending on the regularity of the boundary conditions needed to solve Laplace problems. Unlike previous work, we perform all operations on the original highly-resolved guiding field rather than a cross field so that accuracy is not lost. We use an accurate locator for the singular points that exploits the high resolution solution, and an accurate method for computing their valence. The streamline integration then uses the high order information available to obtain precise streamlines, which reduces the need to coalesce extraneous lines.
2. Mathematical Formulation

The problem is to subdivide any two dimensional domain, $\Omega$, that has piecewise smooth boundaries into quadrilateral subdomains. The domain might be simply or multiply connected. The decomposition will be regular if all the corners of the quadrilaterals have valence four, that is, each corner node connects four edges. In general, it will not be possible to generate a purely regular mesh, especially in multiply connected domains. Instead, some of the nodes will be irregular, where the valence will be larger or smaller than four, like those observed in Fig. 3.

![Figure 3: Portions of quadrilateral meshes, where irregular nodes are encircled with red. Left: Valence three. Right: Valence five. Interior degrees of freedom for fourth order spectral elements are represented by dots.](image)

2.1. Crosses and Cross Field

To motivate the high resolution approach to the quadrilateral decomposition of a domain, $\Omega$, and to put it into context, we recall the notion of a cross. A cross can be represented by

$$C(\psi) = \left\{ \vec{v}_k = (u, v)_k = \left( \cos \left( \psi + \frac{(2k + 1)\pi}{4} \right), \sin \left( \psi + \frac{(2k + 1)\pi}{4} \right) \right) \right\},$$

for $k = 0, 1, 2, 3$. Here, the tangent angle, or phase, $\psi$, is computed from the four quadrant inverse tangent,

$$\psi = \frac{1}{4}\text{atan2}(v, u) \in \left[ -\frac{\pi}{4}, \frac{\pi}{4} \right],$$
so that a cross is represented by four unit vectors $\vec{v}_k$ at any point $\vec{x} \in \Omega$ given a vector field $\vec{v} = (u(\vec{x}), v(\vec{x}))$ at which $\vec{v} \neq 0$ from the principal direction. See Fig. 4. The important property of a cross is its 4-way rotational symmetry. Rotation by an angle $\pi/2$ does not change the cross.

The function $\psi$ is tangent or orthogonal (due to the jump in the atan2 function) to the streamlines of the field $\vec{v}$. Hence, it is parallel to one of the branches of a cross. It is undefined at $\vec{v} = (0, 0)$. Since $\psi$ is computed from the arctangent, there will be a jump of value $\pi/2$ in $\psi$ depending on the signs of $u$ and $v$ even if $\vec{v}$ is smooth. We call lines in the field across which $\psi$ jumps jump lines.

A full discussion of cross fields and methods based on them is beyond the scope of this paper. See, for example, references [1, 11, 12, 13] for more comprehensive discussions.

![Figure 4: A cross defined in a field $\vec{v}$. The axes of the cross lie in the tangent and normal directions to the field lines.](image)

2.2. The Guiding Field, $\vec{v}$

It is necessary, then, to find the guiding field $\vec{v}$ from which to find the singularities and trace streamlines. The only values that can be set a priori are on the boundary $\partial \Omega$ of $\Omega$. We align the field with the boundary to ensure that the mesh is aligned there. Therefore, we set

$$\vec{v}_b = (u_b, v_b) = (\cos(4\theta_b), \sin(4\theta_b)), \tag{3}$$

where $\theta_b$ is the tangent angle of the boundary where the field is being computed. The vector $\vec{v}_b$ at any point along the boundary defines a cross at that point. The factor of four in eq. (3) ensures the same $u, v$ values for each
Figure 5: Boundary crosses shown on a quarter circle domain. Along the circular boundary, $u_b$ has two zeros and one zero contour, while $v_b$ has three zeros.

$90^\circ$ rotation of the angle, and hence the rotational symmetry. Fig. 5 shows example crosses on the boundary of a quarter circle domain.

We make three observations from eq. (3) about the boundary field $\vec{v}_b$:

- Along any smooth closed boundary curve (interior or exterior), there are eight zeros each for $u$ and $v$, and those zeros are not coincident.

- Along smooth portions of the boundary curves and at corners whose angle is a multiple of $\pi/2$, $u$ and $v$ are continuous.

- At boundary corners with angles not divisible by $\pi/2$ there is a jump discontinuity in $\vec{v}_b$ and the cross field.

Once the boundary values are specified using expression (3), they are smoothly propagated to the interior. Smoothing procedures that have been used in the past include marching the associated crosses to the interior [14], minimizing an energy functional [1], or solving a Laplace problem [15]. Solving a Laplace problem for $u$ and $v$

\[
\begin{align*}
\nabla^2 \vec{v} &= 0, & \vec{x} \in \Omega \\
\vec{v} &= \vec{v}_b, & \vec{x} \in \partial \Omega
\end{align*}
\]  

(4)

 guarantees that the field $\vec{v}$ is smooth in the interior of the domain. The solution of the Laplace problem is equivalent to the minimization of the Dirichlet energy. We choose to solve (4) because it allows us to use a high resolution approach such as the spectral element methods. In particular, the
use of a discontinuous Galerkin approximation allows us to handle geometries where the corner angles generate jump singularities in the guiding field, see Sec. 2.2.1.

2.2.1. Continuous and Discontinuous Galerkin Spectral Element Methods

A key difference between our work here and that of previous work is that we compute high resolution solutions for the BVPs (4) with either a continuous Galerkin (CG) or a discontinuous Galerkin (DG) spectral element method on a triangular mesh [3]. Spectral element methods are spectrally accurate, meaning that the convergence rate depends only on the smoothness of the solution. They are high resolution in that they use a large number of degrees of freedom within an element. Unlike traditional low order finite element methods, the high order polynomial expansion of the solution inside each element allows us to locate singularities, identify their valence, and finally trace streamlines with high order accuracy.

In this work, we use the spectral/hp element methods formulation described in detail in reference [3] and implemented in Nektar++ [16]. We briefly describe the fundamentals of this method in what follows. We consider the numerical solution of PDEs of the form

\[ L \mathbf{u} = 0 \]

over a domain \( \Omega \). The domain \( \Omega \) is taken as a set of finite elements, \( \Omega_e \) — the mesh — such that \( \Omega = \bigcup \Omega_e \) and \( \Omega_{e_1} \cap \Omega_{e_2} = \partial \Omega_{e_1 e_2} \) is either an empty set or the interface between two elements and is of one dimension less than the mesh. We solve the PDE problem in the weak sense and require that \( \mathbf{u} |_{\Omega_e} \) is in the Sobolev space \( W^{1,2}(\Omega_e) \). In a CG formulation, we also require the solution to be continuous from one element to the other.

We formulate the solution to (4) in weak form: find \( \tilde{\mathbf{v}} \in H^1(\Omega) \) such that

\[ a(\tilde{\mathbf{v}}, \tilde{\mathbf{w}}) = l(\tilde{\mathbf{w}}) \quad \forall \tilde{\mathbf{w}} \in H^1(\Omega) \quad (5) \]

where \( a(\cdot, \cdot) \) is a symmetric bilinear form, \( l(\cdot) \) is a linear form and \( H^1(\Omega) \) is formally defined as

\[ H^1(\Omega) := W^{1,2}(\Omega) = \{ \tilde{w} \in L^2(\Omega) \mid D^\alpha \tilde{w} \in L^2(\Omega) \forall |\alpha| \leq 1 \} \quad (6) \]

We want to solve this problem numerically and therefore consider solutions in a finite dimensional subspace \( V_N \subset H^1(\Omega) \) and cast the problem: find \( \tilde{\mathbf{v}}^\delta \in V_N \) such that

\[ a(\tilde{\mathbf{v}}^\delta, \tilde{\mathbf{w}}^\delta) = l(\tilde{\mathbf{w}}^\delta) \quad \forall \tilde{\mathbf{w}}^\delta \in V_N, \quad (7) \]
augmented with boundary conditions. In the CG formulation, we also enforce the condition $V_N \subset C^0$.

We take a weighted sum of $N$ trial functions $\Phi_n(\vec{x})$ defined on $\Omega$ so we can represent $\vec{v}^g(\vec{x}) = \sum_n \hat{u}_n \Phi_n(\vec{x})$. This transforms the problem to that of finding the coefficients $\hat{u}_n$ that define $\vec{v}^g(\vec{x})$ within an element. To obtain a unique choice of coefficients $\hat{u}_n$, we place a restriction on $R = L \vec{v}^g$ that its $L^2$ inner product, with respect to the test functions $\Psi_n(\vec{x})$, is zero. In the Galerkin projection we choose the test functions to be the same as the trial functions, i.e. $\Psi_n = \Phi_n$.

The contributions of each element in the domain must be taken into account to construct the global basis $\Phi_n$. A parametric mapping $X_e : \Omega_e \rightarrow \mathcal{E}$ exists from each element $\Omega_e$ to a standard reference element $\mathcal{E} \subset [-1,1]^d$. This mapping is given by $\vec{x} = X_e(\vec{\xi})$. It is important to distinguish $\vec{x}$, the physical coordinates, from the $\vec{\xi}$, the coordinates in the reference space. In Nektar++, triangular, tetrahedral, prismatic, and pyramidal elements are created by collapsing one or more of the coordinate directions to create singular vertices. This allows it to support, for this work, easy-to-generate triangular meshes.

We construct a local polynomial basis on the reference element to represent the solution. A one-dimensional order- $P$ basis is a set of polynomials $\Phi_p(\xi), 0 \leq p \leq P$, defined on the reference segment $-1 \leq \xi_1 \leq 1$. In two and three dimensional reference regions, a tensor basis is used, where the polynomial space is constructed as the tensor product of one dimensional bases on segments, quadrilaterals, and hexahedral reference regions.

Finally, the discrete solution in a physical element $\Omega_e$ can be expressed as

$$\vec{v}^g(\vec{x}) = \sum_n \hat{v}_n \phi_n(X_e^{-1}(\vec{x})), \quad (8)$$

with $\hat{v}_n$ the coefficients computed by the Galerkin procedure. We restrict our solution space to

$$V := \{ \vec{v} \in H^1(\Omega) \mid \vec{v}|_{\Omega_e} \in \mathcal{P}_P(\Omega_e) \}, \quad (9)$$

where $\mathcal{P}_P(\Omega_e)$ is the space of order $P$ polynomials on $\Omega_e$.

An assembly operator can then be designed to assemble the element contributions to the global solution. In the CG formulation, elemental contributions of neighbours are summed to enforce $C^0$-continuity. In the DG
formulation, such mappings transfer flux values from the element interfaces into the global solution vector.

2.3. Critical (Singular) Points

Points where \( v = u = 0 \) are called critical points in the guiding field and singular points in the cross field. They are points where \( \psi \) cannot be determined uniquely. Such points become the interior irregular points (valence \( \neq 4 \)) in the quadrilateral decomposition.

The form of the boundary values usually implies that singular points will exist in the interior of the domain. If there are zeros in \( u \) and \( v \) on the boundary, such as occur when the boundary is smooth and \( \theta_b \) varies continuously, the fact that the interior field is smooth implies that zero contours of \( u \) and \( v \) must cross somewhere in the interior, thereby creating a singular point. Zeros in \( u \) will exist along the boundary at any point where \( \theta_b \) smoothly passes through \( \theta_b = k\pi/8 \), \( k \) odd, and in \( v \) for \( k \) even. Therefore, one can infer the existence of interior irregular points in the final quadrilateral mesh from the boundary curves. A semi-circular section of a boundary, for instance will have four zeros in \( u_b \) and five in \( v_b \) and, in the absence of other nearby features, will create two interior singular points. See Fig. 6. We note that although the cross field directions are undefined at singularities, the field \( \vec{v} \) is smooth because the solutions of the Laplace equation (4) are regular.

Singularities in the cross field occur as recognizable topologies in the \( \vec{v} \) field [17]. They can be categorized analytically by the Poincaré index

\[
i\gamma = \frac{1}{2\pi} \oint_\gamma d\phi,
\]

(10)

where

\[
\phi = \arctan\left( \frac{v}{u} \right).
\]

(11)

When \( \gamma \) encloses a singular point, \( i\gamma = \pm 1 \), where \(-1\) corresponds to a saddle point. Other critical points have index \(+1\). If there is no critical point inside the contour \( \gamma \) then the index is zero. The categorization of the critical point lets one determine the number of streamlines to emanate from the singularity, and hence the valence of the node in the mesh. When \( i\gamma = -1 \) there are five separatrices. When \( i\gamma = +1 \) there are three [1, Lemma 5.1], [10].

Using the definition (2), we similarly define the integral

\[
I_c = \oint_c \frac{d\psi}{d\theta} d\theta,
\]

(12)
Figure 6: Solution $u$ (top left) and $v$ (top right) fields and computed $\psi$ field (bottom) on the half disc geometry. Isocontours of $u = 0$ and $v = 0$ shown in white and black respectively.
Table 1: Value of $I_c$, (12), and the associated valence.

| $I_c$   | Valence ($\mathcal{V}$) |
|---------|-------------------------|
| $-\frac{\pi}{2}$ | 5                        |
| 0       | 4                        |
| $+\frac{\pi}{2}$ | 3                        |

where $c$ is a (small) counter-clockwise circular contour centered on the singular point. Table 1 shows the values of $I_c$ for three streamline topologies in the neighborhood of a singularity. It is equal to the sum of the jumps in $\psi$ around the contour. The valence $\mathcal{V}$ of the critical point is then

$$\mathcal{V} = \frac{2\pi - I_c}{\pi/2}.$$  

(13)

Note, for consistency, that if there is no critical point inside the contour then $I_c = 0$ and the point is regular with valence four.

If one generalizes the integral $I_c$ to be over a part of the circle, one can characterize boundary vertices, too [13]. Let

$$I(\theta_0, \theta_f) = \int_{\theta_0}^{\theta_f} d\psi \frac{d\psi}{d\theta} d\theta,$$  

(14)

so that $I_c = I(0, 2\pi)$. At a boundary point, then, the number of quadrilaterals attached to the point (the boundary itself already being a streamline by construction) is

$$\mathcal{V} = \frac{\Delta \theta - I(\theta_0, \theta_f)}{\pi/2},$$  

(15)

where $\Delta \theta = \theta_f - \theta_0$. $I(\theta_0, \theta_f)$ can be viewed as a correcting factor to $\Delta \theta$ towards a multiple of $\pi/2$.

2.4. Streamline Integration

The guiding field is also used to trace streamlines that will form the separatrices of the block decomposition. We can formulate the problem as finding the trajectory $\vec{x}(t)$ that satisfies

$$\frac{d\vec{x}}{dt} = \vec{v}(\psi'(|\vec{x}|))$$  

(16)
where $t$ is the integration parameter and $\tilde{v}(\psi'(\vec{x}))$ the adjusted guiding field vector. Equation (2) gives a guiding field $\psi \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$, but crosses are invariant to rotations of $\pi/2$. It is therefore necessary to account for the branches of the cross that don’t lie within $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$. These adjusted guiding directions can be obtained by adding a certain number $k = 0, 1, 2, 3$ of $\pi/2$ so that $\psi'$ lies within the appropriate quadrant $\left[-\frac{\pi}{4} + k\frac{\pi}{2}, \frac{\pi}{4} + k\frac{\pi}{2}\right]$. This adjustment operation is especially important when streamlines cross jump lines where $\psi$ abruptly changes by $\pm \pi/2$, but the streamline has to continue in the same overall direction.

The ordinary differential equation (16) requires initial conditions in the form of

$$\vec{x}(t_0) = \vec{x}_0$$  \hspace{1cm} (17)

where $t_0$ is the initial integration time and $\vec{x}_0$ the location of the start of the streamline. In this case, the start of the streamline will either be a singularity or a corner from which a non-boundary separatrix emerges. The ODE (16) can then be integrated in a traditional numerical manner as will be described below in Section 3.3.

3. The Mesh Generation Process

From the discussion above, we see that the decomposition of a domain into quadrilateral subdomains has four stages:

1. Computation of a guiding field. This includes the generation of a triangular mesh and solution of the Laplace problem (4).
2. Finding singular points and their valences, including the valences of corners at boundary vertices.
3. Integration of streamlines to generate the interior boundaries of the quadrilateral decomposition.
4. Cutting of the domain into quadrilaterals using the streamlines as subdomain boundaries, then further subdivision into elements, as necessary.

This section presents details of the implementation of the process and walks through the four stages needed to generate a quadrilateral mesh. The procedure is illustrated using a geometry commonly used in the cross field literature [10, 11], that of a half disc as shown in Fig. 7.
3.1. Solution of the Field Equations

To solve the Laplace problem (4) we use the open source spectral element program Nektar++ [16]. We first generate a triangular finite element mesh which is then made high order and curved by projecting interior nodes onto the curved boundaries [18]. This procedure is carried out in NekMesh, which also has the capability to optimize the high order mesh should some elements be of low quality or simply invalid [19]. Such a mesh is shown in Fig. 8 for the reference geometry.

For the purposes of generating the quadrilateral decomposition, we generate as coarse a triangular mesh as possible and then use high order polynomials to approximate the solution within the elements to get the desired
accuracy. First, we want to take advantage of the spectral accuracy of the spectral element Laplace solver, where a fine mesh is not necessary. Second, as will be discussed in Sections 3.2 and 3.3, a small number of elements is desirable to simplify and speed up the singularity detection and streamline tracing.

Once the curvilinear triangular mesh is generated, the Laplace problem (4) is solved. A special implementation of the Laplace solver has been implemented in the spectral element code Nektar++ [16] for the boundary value problem (4), where the boundary conditions are automatically computed using (3).

Nektar++ supports both continuous Galerkin and discontinuous Galerkin discretizations. The CG approximation is used when the boundary conditions (BCs) are continuous along all boundary curves. As can be seen in the discussion in Sec. 2, boundary conditions are continuous if the boundaries satisfy one of two conditions:

- The curve at each point is $C^2$-continuous, i.e. it is a smooth curve; or
- The boundary curve is only $C^1$-continuous, i.e. a corner, and the angle is a multiple of $\pi/2$.

In this paper, we employ CG whenever possible and switch to DG only when the geometry requires discontinuous BCs. For example, the reference geometry of Fig. 7 contains only smooth curves and $\pi/2$ corners. We therefore use a CG formulation for it. The solution for $\vec{v}$ on the half disc geometry is shown in Fig. 6. As expected, boundary conditions and the solution are smooth, so the solution benefits from rapid convergence of the SEM.

If neither smoothness condition is satisfied at a boundary corner point, the boundary condition (3) has a jump discontinuity there. In that case, we use a DG formulation, which can account for discontinuous BCs. Unlike the CG approximation used in traditional low order solvers, no ad hoc smoothing of a corner’s BCs is required, leading to a discretization-consistent solution.

As an example of a geometry that does not satisfy the smoothness conditions, we present the polygon geometry shown in Fig. 9, which shows the solution and $\psi$ fields.

We note two consequences of using a DG formulation to handle corners with angles that are not multiples of $\pi/2$. First, jump lines can originate from corners. This fact is important, for it shows that one cannot a priori and ad hoc determine the valence of a boundary corner solely from its angle.
Figure 9: Solution $u$ (top left) and $v$ (top right) fields and computed $\psi$ field (bottom) on a polygon with acute and obtuse corners using a DG discretization. Isocontours of $u = 0$ and $v = 0$ are shown in white and black respectively.
Fig. 10 (left) shows how the jump line originating from the top corner \(A\) in the geometry ends in the sole singularity of the domain. Although we have observed that curvature produces singularities, see Sec. 2.3, this straight-sided example shows that a corner whose angle is not a multiple of \(\pi/2\) can also generate interior singularities, again, see Fig. 10 (left).

![Figure 10: Closer views on the computed \(\psi\) field on the polygon geometry.](image)

Second, observe that discontinuous BCs are naturally enforced in the DG formulation. No \textit{ad hoc} smoothing of the boundary is required, as shown in Fig. 10 (right), because the Dirichlet BCs can be enforced either through fluxes when one triangle element used to compute the guiding field shares both edges of the corner (see lower corner, \(B\)), or exactly for all other corners (see upper corner, \(C\)).

When a DG method is used to solve the Laplace problems for the guiding fields, the fields are discontinuous between elements. This is especially true in domains where discontinuous BCs are present, as is clear in Fig. 10 (right). Enough resolution (either through \(h\)- or \(p\)-refinement) needs to be provided so that the jumps at the element interfaces are small. Too low resolution could result in zero isocontours located within solution jumps between elements. The current implementation would not be able to detect such isocontours, nor could it consequently detect singularities. See Sec. 3.2 for more details.

Fortunately, singularities should naturally be located in regions of smooth
fields where jumps between elements are small. Discontinuities between ele-
ments are large only near boundaries where discontinuous BCs appear. The
field on the inside of the domain is mostly smooth, due to the properties
of the Laplace equation, resulting in very small jumps between elements. If
strong curvature is present, we observe that singularities can appear near
curved boundaries where BCs are continuous and the field is smooth. If no
(or little curvature) is present, singularities are controlled by discontinuous
BCs and are seen far away from the boundaries, on the inside of the domain,
again where the field is smooth. In all cases, singularities are not expected
to naturally occur in areas where large jumps between elements exist due to
nearby discontinuous BCs.

Subsequently, the integration of streamlines is unaffected by the discon-
tinuous nature of the discretization. By using a multi step high order inte-
grator, as explained later in Section 3.3, streamline direction is not strongly
affected by abrupt changes of direction between elements. Streamlines re-
main smooth even when gradients in the guiding field are large, e.g. when
crossing a large jump between elements.

3.2. Singularity Detection and Valence Evaluation

Once the solution to eq. (4) is computed, zeros in the field $\vec{v}$ are found and
their valences are evaluated. Unlike the usual cross field approach, we never,
in fact, generate crosses. As seen in Sec. 2 the analysis can be performed
on the $\vec{v}$ field, and $\psi$ is computed only when necessary, i.e. for computing
the valence of vertices and tracing streamlines via (15) and (16). All other
operations can be accomplished by operating on $\vec{v}$.

Singular points inside the domain are located at $\vec{v} = \vec{0}$. To find those,
we first search all elements and flag each that contains a $\vec{v} = \vec{0}$ point at
any quadrature point in the element. If we find at least one value of each
$u \leq 0$, $u \geq 0$, $v \leq 0$ and $v \geq 0$, a singularity is either inside this or a nearby
element. In each flagged element, we then use Newton’s method to approach
the location of $\vec{v} = \vec{0}$. To do so efficiently, we perform the singularity search
in parametric space. If a singularity is found outside the reference element,
we dismiss it and assume that it will be found through parametric search in a
neighbor element. In Fig. 6, we can clearly see the two symmetrically located
singularities where the black and white contours cross, consistent with what
is seen elsewhere for this geometry, e.g. [10].

After completing the search for all singularities, a valence must be com-
puted for each. Referring to Section 2.3 and equation (13), we notice that
the integral \( I_c \) can be evaluated solely by looking for the presence of jump lines. In a counter-clockwise manner, a positive jump (i.e. from \(-\frac{\pi}{4}\) to \(\frac{\pi}{4}\)) leads to a negative integral \( I_c = -\frac{\pi}{2} \), and vice versa. The sign of the jump can itself be determined solely by values of \( \vec{v} \), without the need to compute \( \psi \) or the cross field.

A jump must satisfy two conditions: \( u \) must be negative and \( v \) must change sign. A positive increase of \( v \) indicates a positive jump, and vice versa. To detect jumps, and therefore compute the valence of a singularity, we step counter-clockwise over a sequence of uniformly distributed quadrature points located on a circle of small radius \( c \), centered on the singularity. Following this logic, we determine that the singularities seen in Fig. 6 both have a valence of three.

The valences of all boundary vertices are computed in a similar fashion, this time from expression (15). The angle at the corner, determined from \( \theta_0 \) and \( \theta_f \), is computed from the geometry. To compute the integral \( I_c \) we compute \( \Delta \psi = \psi_f - \psi_0 \) from the values of \( \psi \) at the boundaries from the boundary conditions. To \( \Delta \psi \), we add a jump contribution, if present, in the same way as for the computation of singularity valences. Because both corners in Fig. 6 have an angle of \( \frac{\pi}{2} \), a valence of one can be trivially determined.

It is important to note that the boundary vertex operation does not require an \textit{a priori} determination of the corner valence based on angle only. The corner valences can be determined through equation (15) which does not only depend on the known BCs but also on the existence of jump lines in the \textit{computed} \( \psi \) field in the neighborhood of the vertex, like that seen in Fig. 10.

The boundary vertex valence gives the number of quads located at the associated corner. Due to the definition, this valence could be zero. Indeed, \( \Delta \theta \) and \( \Delta \psi \) could cancel out if the corner is sufficiently sharp. Physically, this indicates the presence of a degenerate quadrilateral block (i.e. a triangular block) where all streamlines would converge towards the degenerate corner. This topology requires \textit{ad hoc} manipulation where the streamlines are used later to construct the quadrilateral subdivision of the domain. This manipulation will be explained in the next section.

3.3. Streamline Integration and Manipulation

Streamlines are traced throughout the domain after the valences of all singularities and vertices are determined. The first step is to find the initial direction of each streamline. Because \( \vec{v} = 0 \) at singularities, \( \psi \) is undefined.
Therefore, we choose to evaluate one of the streamline angles from \( \vec{v} \) at a small distance \( c \) of the singularity. This distance is currently set empirically based on the size of the elements.

The initial direction is refined iteratively from an initial guess using Algorithm 1 which is inspired by bisection. For the first streamline, we can take any initial guess for the direction. For subsequent streamlines, we take an initial guess at angles multiples of \( \frac{2\pi}{V} \). The size of the tolerance, \( \epsilon \) is not critical; we use \( 10^{-9} \).

**Data:** Direction \( \alpha_0 \) (initial guess), Singularity location \( \vec{p}_0 \)

**Result:** Direction \( \alpha_f \) (converged value)

Initialize \( \alpha \) from \( \alpha_0 \);

while \( |\Delta \alpha| > \epsilon \) do

1. Compute point \( \vec{p}_1 \) at distance \( c \) and direction \( \alpha \) of \( \vec{p}_0 \);
2. Interpolate \( \vec{v} \) and compute \( \psi \) at \( \vec{p}_1 \);
3. Find \( \psi' = \psi + k \frac{\pi}{2} \) where \( k = 0, 1, 2, 3 \) such that \( \Delta \alpha = |\alpha - \psi'| \) is minimized;
4. Update \( \alpha = \psi' \);

end

Return \( \alpha_f = \alpha \);

**Algorithm 1:** Determination of the initial direction of a streamline at a singularity \( \vec{p}_0 \).

The angle search in Alg. 1 can also be applied to boundary vertices, including those where the boundary conditions might be discontinuous and \( \psi \) is ambiguous. An initial guess for all streamlines may be obtained at angles multiples of \( \frac{\Delta \theta}{V} \).

After an initial direction is obtained, streamlines are synchronously advanced throughout the domain. This part of the procedure is the most computationally expensive. Each new streamline point requires the search for the element that contains it. Next, the inverse map \( \vec{\xi} = \mathcal{X}_e^{-1}(\vec{x}) \) is used to transform to parametric coordinates. Finally the high order interpolation of \( \vec{v} \) is computed via eq. (8).

A multi stage integration like Runge-Kutta, where intermediate evaluations are used, would prove too expensive to integrate the streamlines. Instead, we use a 4th order multi step Adams-Bashforth integrator. The current implementation reduces the order of the algorithm when too few steps are available. While advancing the streamlines, it is important to validate the direction \( \alpha \) obtained at the latest point. When a streamline crosses a jump
line, $\psi$ abruptly rotates $\pm \pi/2$ and $\alpha$ must be adjusted accordingly.

As streamlines are advanced throughout the domain, they may meet. To anticipate two streamlines meeting, the front points of each are compared at each step. If the distance between the front points of two streamlines is less than the step size and they are advancing in opposite directions, the streamlines are assumed to meet. When the absolute value of the difference in the latest $\alpha$ values equals $\pi$ when rounded to the nearest $\pi/2$, we consider the streamlines to be advancing in opposite directions. Each of two meeting streamlines are then advanced to the starting point of the other, keeping the same number of points in each. The two streamlines are then merged point-wise. This constitutes expected or normal merging, but the user may also want to conduct aggressive merging to reduce the number of streamlines and therefore simplify the block decomposition.

For aggressive merging, a larger distance threshold can be carefully chosen to make streamlines that would normally just miss each other merge anyway. An example of this is presented in Fig. 11 where the left figure corresponds to normal merging and the right one to aggressive merging. In this case, a distance threshold of 5 times the step size was used for aggressive merging.

A merged streamline is created by weight-averaging pairs of streamline points. A simple linear weight function could be used, but that will change the angles computed with Alg. 1 at the starting points, both singularities

![Figure 11: Example of normal (left) and aggressive (right) merging. A distance threshold of 5 times the step size was used for aggressive merging.](image-url)
and boundary vertices. This is not desirable as the PDE problem has already
optimized these angles to be as equally distributed as possible. Instead we
use trigonometric weight functions of the form \( W_0(x) = \cos^2(x) \) and \( W_1(x) = \sin^2(x) \), where \( x \in [0, 1] \) is the parametrization along the streamline. Because
\( \frac{dW_0}{dx} \) and \( \frac{dW_1}{dx} \) are zero at \( x = 0 \) and \( x = 1 \), angles are preserved at the initial
and final points of the merged streamline. This approach is especially useful
when performing aggressive merging of two reasonably distant streamlines.

Fig. 11 (right) shows the curving of some streamlines to preserve the original
direction at each end.

3.4. Quad Mesh Generation

NekMesh relies on the OpenCASCADE platform [20] as its CAD engine
for mesh generation and for the projection to high order boundary representa-
tions. OpenCASCADE also includes tools for CAD manipulation. We make
use of these capabilities for spline and wire creation and for face splitting.

The streamlines computed from the guiding field are transformed into
OpenCASCADE edges represented by interpolating splines. These edges can
be joined into a set of wires, with each wire consisting of topologically con-
ected edges. The original CAD file is loaded again, consisting of a single two
dimensional face, representing the domain \( \Omega \). This face is then iteratively
split by each wire and a set of quadrilateral faces is obtained. These faces are
topologically connected, meaning that the future mesh will be conforming.
At this point, each face can be trivially meshed with a single quadrilateral
and projected to high order. Fig. 12 (left) illustrates the quadrilateral de-
composition and coarse quadrilateral mesh obtained for the half circle after
computing the singularities and tracing of the streamlines.

We use the midpoint division approach [21] to split a triangular block
that forms when a corner valence is zero. We insert an artificial 3-valence singularity which we connect to each of the three sides of the triangle. One streamline can then be physically integrated away from the degenerate corner and throughout the domain. The other two branches of the 3-valence singularity are defined as straight lines at $\pm \frac{2\pi}{3}$ from the physical streamline. Fig. 13 shows what the midpoint division of a degenerate quadrilateral might look like.

![Figure 13: Detail of a domain showing a midpoint division of a degenerate quadrilateral caused by a sharp angled corner. The full domain is shown in Fig. 16.](image)

If desired, we make use of an isoparametric splitting approach described in [22] to further subdivide the quadrilateral subdomains into smaller elements. This technique, initially developed for boundary layer mesh division, guarantees that subdivided elements have the same quality as the macro element from which they are created. In the current context, it also ensures that angles are preserved. Each row of quads can be individually split using the isoparametric approach. The right-hand side of Fig. 12 shows what the mesh on the left looks like after a conformal splitting of each row of quads.

4. Examples

We present four more examples of meshes generated for geometries taken from, or inspired by, the literature. The meshes shown in Fig. 12 are typical of those already generated by low order approaches [1, 10, 11]. We use the additional examples to illustrate the application of the high resolution approach on multiply connected geometry, one for which low order approaches
have been shown to have difficulty, a polygon with both acute and obtuse angles, and a NACA 0012 airfoil.

The geometry for Example I shown in Fig. 14 is a rectangle with two quarter circle holes. This geometry has also been meshed, for example, in reference [10]. The minimum number of singularities (two 3-valence singularities) is obtained while preserving the symmetry of the geometry without the need to coalesce spurious singularities that can result from low order detection algorithms. Fig. 14 shows the coarse quadrilateral mesh obtained on this geometry.

Example II, shown in Fig. 15, is one that low order methods generate spurious spiral separatrices that end in a limit cycle, therefore creating an invalid block decomposition [11, 12]. To date, no approach using cross fields obtains a valid separatrix graph. The use of high order integration of an accurate guiding field allows NekMesh to generate streamlines that do not form a spiral. In fact, the singularities found by NekMesh on this geometry are identical to those one would obtain for a simple disc. Fig. 15 shows the coarse quadrilateral mesh obtained on this spiral geometry. It is also important to note that Fig. 15 was generated using aggressive merging with a distance threshold of value 5 times the step size. If we were to use normal merging, the block decomposition would have looked more complicated, as
shown in Fig. 11 (left), reducing the flexibility for users to split the mesh to their liking.

Figure 15: Coarse quadrilateral mesh obtained on Geometry II, a spiral.

Example III consists of a polygon whose corners include acute and obtuse angles. We use this example to demonstrate the use of the DG discretization for the solution of the guiding field. The rest of the meshing procedure remains the same and is in fact unaffected by the type of discretization. Fig. 9 has already shown the solution \( \vec{v} \) and computed \( \psi \) field for this geometry, with \( u = 0 \) and \( v = 0 \) isocontours shown in white and black, respectively. As mentioned earlier in Sec. 3.3, the only consideration when using the DG formulation is that the approximate solutions are accurate enough to not affect the streamline integration.

The Example III geometry features a sharp corner (far right, Fig. 16) whose valence is evaluated as zero. That creates a degenerate quadrilateral that must be split into three valid quads as explained in Section 3.3. Fig. 16 shows the coarse quadrilateral mesh obtained on the polygon geometry. It contains only two irregular points, one created \textit{ad hoc} for the degenerate corner and the other detected as a singular point in the guiding field.

The coarse meshes can be split further according to the user’s preference using the isoparametric splitting described in Section 3.4. We show in Fig. 17 examples of two split meshes for the quadrilateral decompositions obtained above.
Finally, Example IV represents a geometry more relevant for CFD applications, that of a NACA 0012 profile in a rectangular domain. Since the trailing edge angle is not a multiple of \( \pi/2 \), a DG discretization was used to compute the guiding field. With this example, we demonstrate the flexibility of the isoparametric splitting module in NekMesh to define different distributions of elements. This allows the user to easily obtain elements of the required size, e.g. high aspect ratio elements in the boundary layer of the airfoil. Fig. [18] shows the coarse and the split quadrilateral meshes obtained on this NACA 0012 geometry.

5. Summary and Conclusions

We have developed a field guided method to generate quadrilateral meshes for general two dimensional domains. Inspired by cross field mesh generation methods, the procedure consists of four steps. The first is to compute the guiding field on an existing triangular mesh. For that, we use a spectral element method to compute a high resolution approximation to the two Laplace problems for the guiding field. We use either a continuous or discontinuous Galerkin method depending on the angles formed along the physical boundaries. If the angles are not multiples of \( \pi/2 \), the DG formulation can handle the discontinuous boundary conditions in a discretization consistent manner,
Figure 17: Split meshes obtained on Geometries I and III.
Figure 18: Coarse (top) and split (bottom) quadrilateral meshes obtained on Geometry IV, a NACA 0012 profile. High order interior degrees of freedom have been hidden for clearer visualization.
without the need for ad hoc smoothing required by the CG formulation. Singular points in the field and their valences are then found by exploiting the local high order polynomial approximations within each element. Streamlines are computed in the field, again, using the high order solutions rather than crosses at element corners. Finally, the domain is cut into elements of the desired size. The procedure is implemented in the open source program NekMesh, and uses the companion open source Nektar++ to compute the guiding fields.

An advantage of the approach is that it can generate meshes with naturally curved quadrilateral elements that do not need to be curved a posteriori to eliminate invalid elements. It avoids low order errors that make it difficult to locate singular points and accurately trace streamlines.

Numerous refinements to the algorithm can be incorporated. The guiding field can be adjusted, and hence the location of the irregular points moved, by solving a Poisson rather than a Laplace problem for the guiding fields. The use of an adaptive streamline integrator can efficiently integrate streamlines when the geometric scales have large variations over the domain, thus making the approach more flexible for very complex geometries. Nonetheless, the procedure is applicable for the generation of curved quadrilateral meshes over general two dimensional domains.

Acknowledgements

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675008. This work was supported by a grant from the Simons Foundation (#426393, David Kopriva). The authors would also like to thank Prof. Gustaaf Jacobs of the San Diego State University for his hospitality.

References

[1] R. Viertel, B. Osting, An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg-Landau theory arXiv:1708.02316.

[2] J. Kozdon, E. Dunham, J. Nordström, Simulation of dynamic earthquake ruptures in complex geometries using high-order finite difference methods, Journal of Scientific Computing 55 (1) (2013) 92–124. doi:10.1007/s10915-012-9624-5
[3] G. E. Karniadakis, S. J. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics (Second Edition), Oxford University Press, 2005.

[4] D. A. Kopriva, Implementing Spectral Methods for Partial Differential Equations, Scientific Computation, Springer, 2009.

[5] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, D. Zorin, Quad-mesh generation and processing: A survey, Computer Graphics Forum 32 (6) (2013) 51–76.

[6] F. Hindenlang, Mesh curving techniques for high order parallel simulations on unstructured meshes, Ph.D. thesis, University of Stuttgart (2014).

[7] B. H. V. Topping, J. Muylle, P. Ivanyi, R. Putanowicz, B. Cheng, Finite Element Mesh Generation, Saxe-Coburg Publications, 2004.

[8] X. Liang, M. S. Ebeida, Y. Zhang, Guaranteed quality all-quadrilateral mesh generation with feature preservation, Computational Methods in Applied Mechanics and Engineering 199 (2010) 2072–2083.

[9] R. Schneiders, Algorithms for quadrilateral and hexahedral mesh generation, Tech. rep., MAGMA Gießereitechnologie GmbH. URL \texttt{www.robertschneiders.de/papers/vki.pdf}

[10] N. Kowalski, F. Ledoux, P. Frey, A PDE based approach to multidomain partitioning and quadrilateral meshing, in: 21th Int. Meshing Roundtable, 2012.

[11] R. Viertel, M. Staten, B. Osting, Toward a paver replacement, in: 26th International Meshing Roundtable, IMR26, 2017.

[12] G. Bunin, A Continuum Theory for Unstructured Mesh Generation in Two Dimensions, Tech. rep. (sep 2006). \texttt{arXiv:0609078}. URL \texttt{http://arxiv.org/abs/cs/0609078}

[13] G. Bunin, Towards Unstructured Mesh Generation Using the Inverse Poisson Problem, ArXiv e-prints: \texttt{arXiv:0802.2399}
[14] H. J. Fogg, C. G. Armstrong, T. T. Robinson, Multi-block decompo-
sition using cross-fields, in: J. P. M. de Almeida, P. Díez, C. Tiago,
N. Pares (Eds.), VI International Conference on Adaptive Modeling and
Simulation, 2013.

[15] J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Béchet,
E. Marchandise, C. Geuzaine, T. Mouton, A frontal Delaunay quad
mesh generator using the $l^\infty$ norm, International Journal For Numerical
Methods In Engineering 94 (5) (2013) 494–512. doi:10.1002/nme.4458

[16] C. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo,
D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu,
Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. Kirby,
S. Sherwin, Nektar++: An open-source spectral/hp element framework,
Computer Physics Communications 192 (2015) 205–219. doi:10.1016/
j.cpc.2015.02.008

[17] X. Tricoche, G. Scheuermann, H. Hagen, Continuous topology simpli-
fication of planar vector fields, in: Proceedings of the Conference on
Visualization '01, VIS '01, IEEE Computer Society, Washington, DC,
USA, 2001, pp. 159–166. doi:10.1109/VISUAL.2001.964507

[18] S. Sherwin, J. Peiró, Mesh generation in curvilinear domains using high-
order elements, International Journal for Numerical Methods in Engi-
neering 53 (1) (2002) 207–223. doi:10.1002/nme.397

[19] M. Turner, J. Peiró, D. Moxey, Curvilinear mesh generation using a vari-
ational framework, Computer-Aided Design doi:10.1016/j.cad.2017.
10.004

[20] Open Cascade SAS, Open Cascade (2018).

[21] T. Li, R. McKeag, C. Armstrong, Hexahedral meshing using midpoint
subdivision and integer programming, Computer Methods in Applied
Mechanics and Engineering 124 (1-2) (1995) 171–193. doi:10.1016/
0045-7825(94)00758-F

[22] D. Moxey, M. Green, S. Sherwin, J. Peiró, An isoparametric approach
to high-order curvilinear boundary-layer meshing, Computer Methods
in Applied Mechanics and Engineering 283 (2015) 636–650. doi:10.
1016/j.cma.2014.09.019

31