Automatic Structured Quadrilateral Grid Generation via Frame Fields

Marvyn Bailly

December 11, 2025

Abstract

This paper presents an investigation into automatic structured quadrilateral grid generation using frame fields. The research explores the theoretical foundations and practical implementation of frame field-based methods for generating high-quality structured meshes suitable for finite volume methods. We discuss the mathematical framework, implementation details, and demonstrate the effectiveness of the approach through various test cases.

Contents

1 Introduction

In Computional Fluid Dynamics (CFD), the generation of high-quality meshes is crucial for accurate simulations which use finite volume methods. Structured quadrilateral grids are particularly advantageous due to their regular connectivity and alignment with flow features. This paper explores the use of frame fields for automatic generation of structured quadrilateral grids, providing a systematic approach to mesh generation that can adapt to complex geometries. In this study, we implement the Mixed-Integer Quadrangulation (MIQ) method in Julia as described by Bommes et al. (Bommes 2009) and evaluate its performance on various geometries.

We aim to generate structured quadrilateral meshes that maintain the following qualities:

- **Indiviual Cell Quality:** Each quad should be as close to a square as possible.
- **Orientation:** The grid should align with the geometry's features and flow directions.
- **Alignment:** The grid should resolve sharp features and boundaries accurately.
- **Global Structure:** Points in the mesh known as singularities are required to composite for the Gaussian curvature of a general surface. singularities correspond to irregular vertices in the quad mesh where the number of incident edges differs from four. These should be minimized and strategically placed.

In the following sections, we will detail the methodology used to implement the MIQ algorithm, present the results of our mesh generation experiments, discuss the implications of our findings, and conclude with insights in this area.

2 Methodology

We break the methodology into two main steps: (1) Frame Field Generation and (2) Quadrilateral Mesh Extraction. The core of both steps relies on solving a minimzation problem where some of the variables are integers while the others are real-valued. This is known as a Mixed-Integer Program (MIP) which are generally NP-hard to solve. To address this, we implement an approximation algorithm known as a greedy mixed-integer solver as described by Bommes et al..

2.1 Mixed-Integer Solver

2.1.1 Problem Formulation

The minimzation of a quadratic energy function $E(x_1, \ldots, x_n)$ is an integer problem if $x \in \mathbb{Z}^n$. However, in this paper, we encounter a more general problem were some of the unknown variables $x_1, \ldots, x_k$ are integers while the remaining variables $x_{k+1}, \ldots, x_n$ are real-valued.

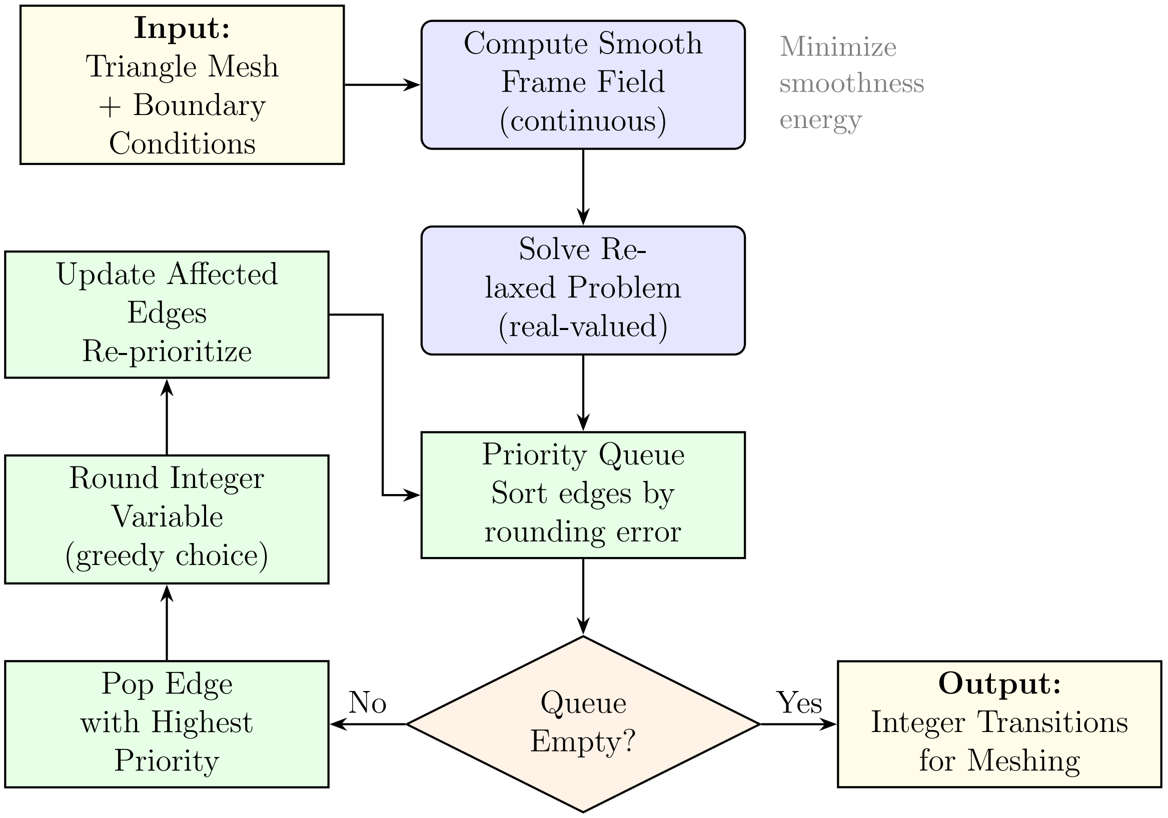

Figure 1: High-level workflow of the greedy mixed-integer solver.

This is known as a Mixed-Integer Program (MIP) which are generally NP-hard to solve and so we hope to avoid solving them directly. We note that the continuous relaxation of a MIP, obtained by allowing all variables to be real-valued, provides a lower bound on the optimal solution of the original MIP. Thus this provides a reasonable starting point for approximation algorithms.

2.1.2 Common Approaches

A common approach to approaching a MIP is to first compute the continuous minimizer, which simlpiy requires the solution of the linear system $\{\partial E/\partial x_i = 0\}_{i=1}^{n}$. Then the first k variables of the oslution are rounded to the nearest integer and treated as constant while resolving the linear system for the remaining real-valued variables. This is known as direct rounding but the solution can signficailty deviate from the optimal solution. This motivates the authers to propose a greedy mixed-integer solver which is explained next.

2.1.3 Greedy Mixed-Integer Solver

Rather than rounding the first k real valued variables to integers, the greedy rounding approch rounds one integer variable at a time. This is followed by an immediate update of the continous part treating the integers as constants.

Let $\vec{x}^0$ be the continuous solution to the linear system and let x_i, $i \leq k$ be the variable which causes the smallest absolute error if we round it to the nearest integer. Then we set x_i to this integer value. We solve the remaining variables $\vec{x}^1$ and continue to eliminate in eahc step that variable which cauess the least round-off error until we have an integer for all $x_1, \ldots, x_k$.

To reduce the computational cost, we use a local Gauss-Seidel update rather than solving the full linear system after each rounding step. That means after after x_i is rounded, we add all variables whose Gauss-Seidel update depends on x_i to a priority queue. These elements correspond to the nonzero elements in row A_i. Now on each iteration, we pop the highest priority element from the queue, compute the local residuum

$$r_k = b_k - \sum_{j \neq n} A_{kj} x_j. \tag{1}$$

If $|r_k| \geq \tau$, a small threshold, we update $x_k \leftarrow x_k - r_k/A_{kk}$ and push all variables which depend on x_k back into the priority queue. This continues until the queue is empty. If the method does not converge in a reasonable number of iterations, we fall back to solving the full linear system. Figure 1 illustrates the high-level workflow of this algorithm while Algorithm 1 provides the detailed pseudocode.

2.2 Frame Field Generation

In this section we introduce the mathematical formulation and implementation details for generating a smooth frame field over a given triangular mesh.

2.2.1 Mathematical Formulation

A frame field is a N-symmetry direction field (Ray et al. 2008b) where at each point in the domain, N directions are defined that are symmetric under rotation by $2\pi/N$ around the point of interest. For quadrilateral meshing, we are interested in 4-symmetry frame fields, or cross fields, where $N = 4$. The frame field is represented by an angle-field $\theta : F \mapsto \mathbb{R}$ which assigns an angle θ_f to each face $f \in F$ of the triangular mesh and a period-jump field $p : E \mapsto \mathbb{Z}$ which assigns an integer period-jump p_e to each edge $e \in E$ of the mesh.

The idea is that for each face f, we can define a local frame of reference by rotating a reference edge e along the x-axis. Then the four directions of the frame at face f are given by rotating this reference edge by angles $\theta_f + k \cdot \pi/2$ for $k = 0, 1, 2, 3$. This is visualized in Figure 2 a. The period-jump field p_e determines which direction in face f_1 corresponds to which direction in the adjacent face f_2 across edge e. Specifically, the direction in face f_2 that aligns with the reference edge is given by $\theta_{f_2} + p_e \cdot \pi/2$. This relationship is illustrated in Figure 2 b.

Algorithm 1: Greedy Mixed-Integer Solver

Input: System matrix A, right-hand side b, number of integer variables k, threshold τ
Output: Solution vector $\vec{x}$ with integer values for $x_1, \dots, x_k$

```
// Solve continuous relaxation
Solve linear system $A\vec{x}^0 = b$ for all variables;
rounded $\leftarrow \emptyset$;

// Round integer variables greedily
for $i = 1$ to $k$ do
    // Find variable with minimum rounding error
    $j \leftarrow \arg\min_{j \in \{1,\dots,k\} \setminus \text{rounded}} |x_j^{i-1} - \text{round}(x_j^{i-1})|$;
    // Round and fix variable
    $x_j \leftarrow \text{round}(x_j^{i-1})$;
    rounded $\leftarrow$ rounded $\cup \{j\}$;

    // Update affected continuous variables using Gauss-Seidel
    // Q contains indices $m$ where $A_{mj} \neq 0$ (row $j$ has nonzero entry)
    $Q \leftarrow \{m : A_{mj} \neq 0\}$;
    while ($Q$ is not empty) and (iteration count < max_iterations) do
        $m \leftarrow Q.\text{pop}()$;
        // Compute local residual
        $r_m \leftarrow b_m - \sum_n A_{mn} x_n$;
        if $|r_m| \geq \tau$ then
            // Update variable
            $x_m \leftarrow x_m - r_m / A_{mm}$;
            // Add dependent variables to queue
            for each $n$ where $A_{nm} \neq 0$ do
                $Q.\text{push}(n)$;

return $\vec{x}$;
```

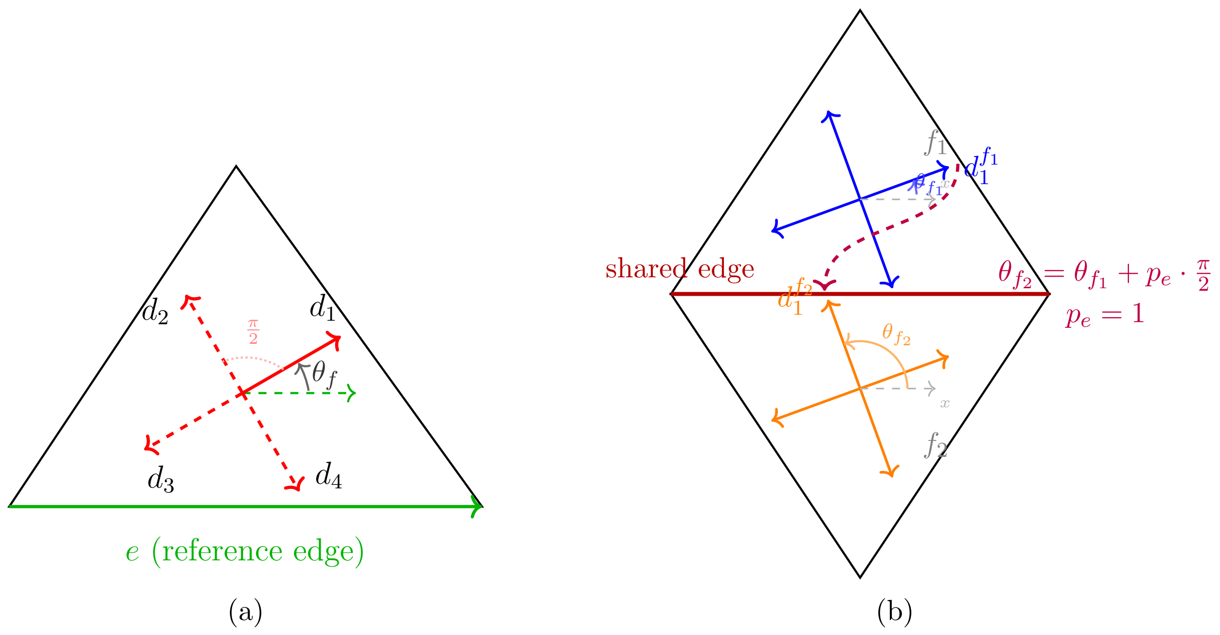

Figure 2: Frame field visualization. (a) The reference edge e is aligned with the x-axis. The frame at face f consists of four directions obtained by rotating the x-axis by angles $\theta_f, \theta_f + \frac{\pi}{2}, \theta_f + \pi, \theta_f + \frac{3\pi}{2}$. (b) The frame field in face f_2 is related to face f_1 by $\theta_{f_2} = \theta_{f_1} + p_e \cdot \frac{\pi}{2}$, where $p_e = 1$ causes a 90° rotation.

2.2.2 Constructing a Smooth Frame Field

The smoothness of a unit frame field can be measured by the angular difference between faces:

$$E_{\text{smooth}} = \sum_{e_{ij} \in E} \left(\theta_i + \kappa_{ij} + \frac{2\pi}{N} p_{ij} - \theta_j \right)^2, \tag{2}$$

where θ_i, and θ_j are the angles of trangle i and j to it's respective reference edge, $\kappa_{ij} \in (-\pi, \pi]$ is the angle between the reference edges of triangles i and j, p_{ij} is the integer period-jump across edge e_{ij}, and $N = 4$ for quadrilateral meshing.

The cross field index of a vertex v is defined as:

$$I(v) = I_0(v) + \sum_{e_{ij} \in \text{star}(v)} \frac{p_{ij}}{4}, \tag{3}$$

where $I_0(v)$ is the initial index determined by the geometry of the mesh

$$I_0(v) = \frac{1}{2\pi} \left(A_d(v) + \sum_{e_{ij} \in \text{star}(v)} \kappa_{ij} \right), \tag{4}$$

where $A_d(v)$ is the angle defect at vertex v and star(v) is the set of edges incident to vertex v. Ray et. al. (2008b) show that vertices with non-zero index correspond to singularities in the frame field which will later correspond to irregular vertices in the quadrilateral mesh. The index will be an multiple of 1/4 and determine the valence of the irregular vertex in the quad mesh e.g., an index of $+1/4$ corresponds to a vertex with valence 3 while an index of $-1/4$ corresponds to a vertex with valence 5.

2.2.3 Finding a Smooth Frame Field

The problem we wish to solve is given a triangular mesh M and a subset of faces $F_c \subset F$ with constrained direction $\theta_i = \hat{\theta}_i$ for $i \in F_c$, we search for a smooth frame field by minimizing the energy given by equation 2. That is, we wish to solve for θ_i and p_{ij}, which minimize E_{smooth} subject to the constraints $\theta_i = \hat{\theta}_i$ for $i \in F_c$. This is a mixed-integer problem since θ_i are real-valued while p_{ij} are integers.

To reduce the problem space, we can take advantage of the invaraiance under rotation of the frame field. Following the work of Bommes et al. (2009), the minimization problem can be mind unique by fixing one period jump variable to zero per triangle. In order to not change the value of the energy, we take care not to fix edges whose dual paths connect two constrained faces, or closes loops. To generate a valid set of edges whose period jumps may be fixed, we construct a Dijkstra tree over the dual graph of the triangular mesh, starting from the constrained faces.

2.2.4 Implementation Details

To implement the frame field generation, we first construct the Dijkstra tree to identify edges whose period jumps can be fixed. We then set up the minimization problem by taking the derivative of the energy function with respect to each variable, leading to a system of linear equations for the real-valued variables θ and integer constraints for the period jumps p. The resulting equations are as follows:

$$\begin{aligned} \frac{\partial E_{\text{smooth}}}{\partial \theta_k} &= \sum_{e_{kj} \in \text{star}(v)} 2\left(\theta_k + \kappa_{kj} + \frac{\pi}{2} p_{kj} - \theta_j\right) = 0, \\ \frac{\partial E_{\text{smooth}}}{\partial p_{ij}} &= \pi\left(\theta_i + \kappa_{ij} + \frac{\pi}{2} p_{ij} - \theta_j\right) = 0. \end{aligned} \tag{5}$$

For all variables that are not fixed, we assemble a row in a single matrix that we pass to the greedy mixed-integer solver described earlier. The solver returns the optimal θ and p values which define the smooth frame field over the triangular mesh adhering to the user-defined constraints on F_c.

2.3 Quadrilateral Mesh Extraction

3 Results

4 Discussion

5 Conclusion

A Julia Code Listings

All code used in this project is available at the following GitHub repository: `https://github.com/MarvynBailly/Structured-Grid-Generation-via-Frame-Fields`.

B Reference Paper

The following pages contain the full text of Bommes et al.