

# 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

|          |                                         |           |
|----------|-----------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                     | <b>3</b>  |
| <b>2</b> | <b>Methodology</b>                      | <b>3</b>  |
| 2.1      | Mixed-Integer Solver . . . . .          | 3         |
| 2.1.1    | Problem Formulation . . . . .           | 3         |
| 2.1.2    | Common Approaches . . . . .             | 4         |
| 2.1.3    | Greedy Mixed-Integer Solver . . . . .   | 4         |
| 2.2      | Frame Field Generation . . . . .        | 5         |
| 2.2.1    | Mathematical Formulation . . . . .      | 5         |
| 2.3      | Quadrilateral Mesh Extraction . . . . . | 7         |
| <b>3</b> | <b>Results</b>                          | <b>8</b>  |
| <b>4</b> | <b>Discussion</b>                       | <b>9</b>  |
| <b>5</b> | <b>Conclusion</b>                       | <b>10</b> |
| <b>A</b> | <b>Julia Code Listings</b>              | <b>11</b> |
| <b>B</b> | <b>Reference Paper</b>                  | <b>12</b> |

# 1 Introduction

In Computational 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 Bommès et al. (Bommès 2009) and evaluate its performance on various geometries.

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

- **Individual 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 minimization 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 Bommès et al..

### 2.1 Mixed-Integer Solver

#### 2.1.1 Problem Formulation

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

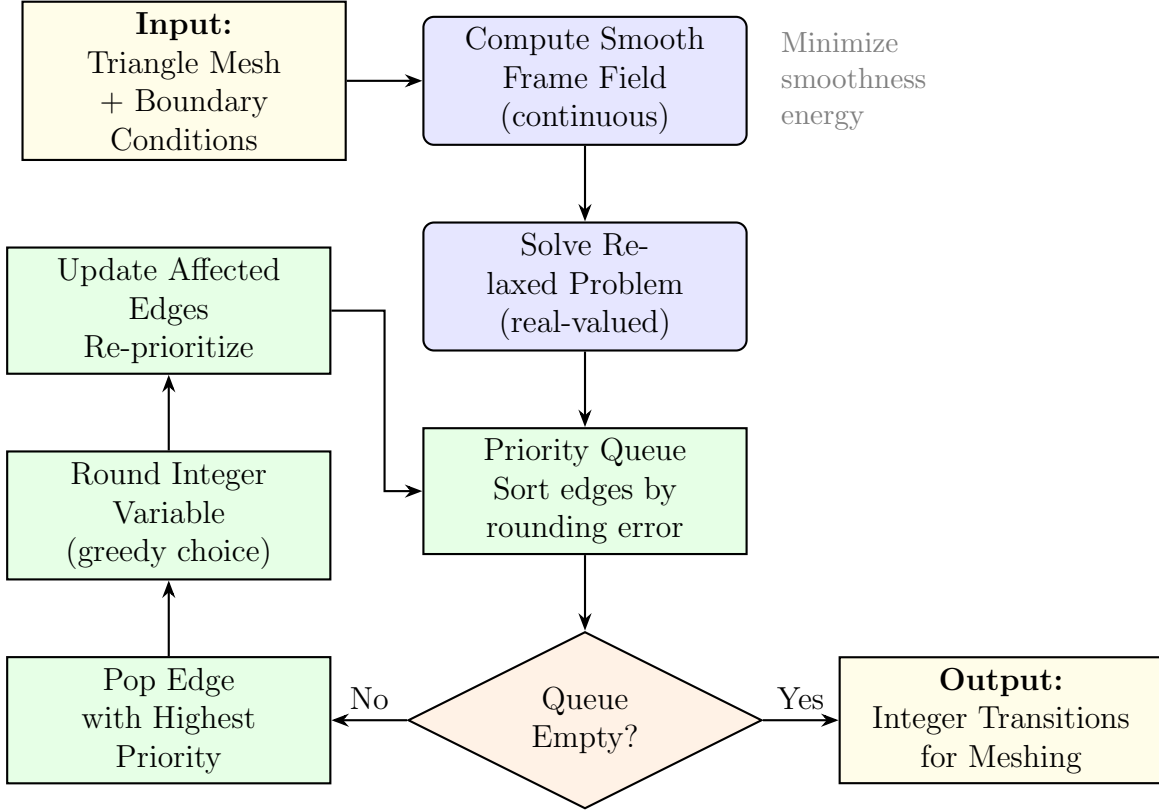


Figure 1: High-level workflow of the greedy mixed-integer solver.

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 simply requires the solution of the linear system  $\{\partial E / \partial x_i = 0\}_{i=1}^n$ . Then the first  $k$  variables of the solution 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 significantly deviate from the optimal solution. This motivates the authors 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 approach rounds one integer variable at a time. This is followed by an immediate update of the continuous 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 each step that variable which causes the least round-off error until we have an integer for all  $x_1, \dots, x_k$ .

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  $x_i$  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

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

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  $\theta_f$  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  $k = 0, 1, 2, 3$ . 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  $\theta_{f_2} + p_e \cdot \pi/2$ .

---

**Algorithm 1:** Greedy Mixed-Integer Solver

---

**Input:** System matrix  $A$ , right-hand side  $b$ , number of integer variables  $k$ , threshold  $\tau$ **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  $< \text{max\_iterations}$ ) **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**  $\vec{x}$ ;

---

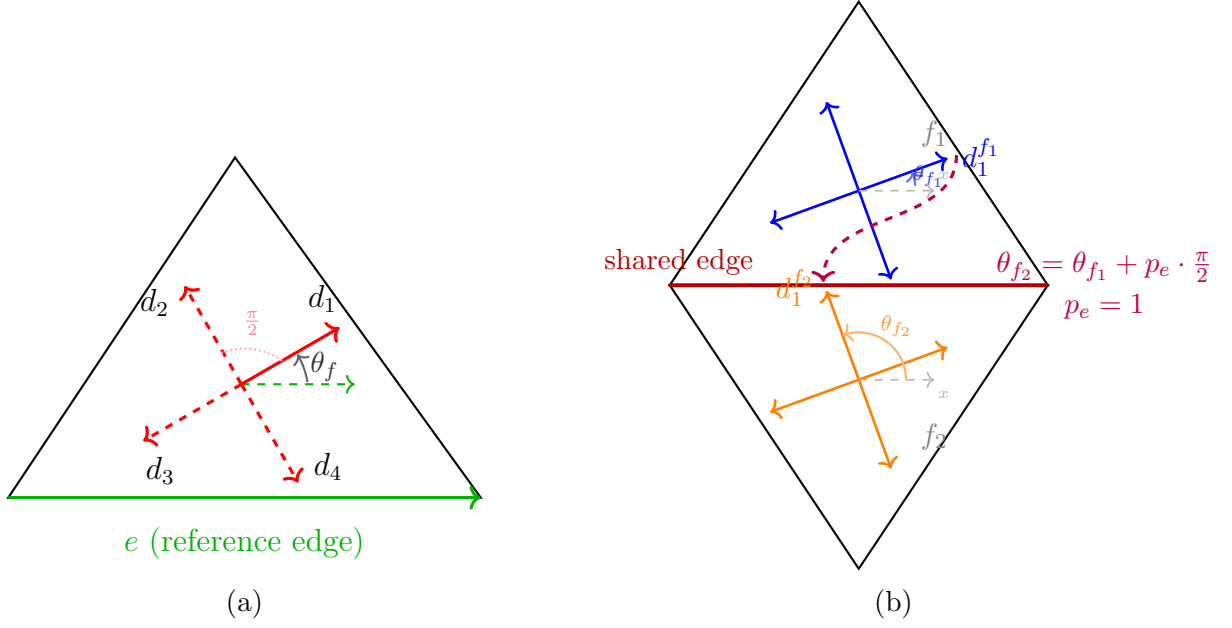


Figure 2: Frame field visualization. (a) The reference edge  $e$  is aligned with the  $x$ -axis. The frame at face  $f$  consists of four directions obtained by rotating the  $x$ -axis by angles  $\theta_f, \theta_f + \frac{\pi}{2}, \theta_f + \pi, \theta_f + \frac{3\pi}{2}$ . (b) The frame field in 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  $90^\circ$  rotation.

### 2.3 Quadrilateral Mesh Extraction

### 3 Results



## 4 Discussion

## 5 Conclusion

## A Julia Code Listings

## **B Reference Paper**

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