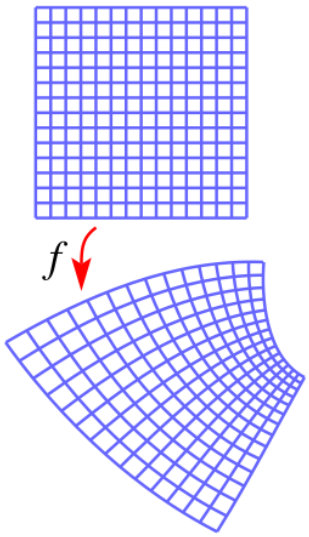
ÉCOLE NORMALE SUPÉRIEURE PARIS-SACLAY

Project Report - Proof of Concept

Contour-Parallel Offset Machining for Trimmed Surfaces

Based on Conformal Mapping with Free Boundary



Riadh ZERKOUK

Supervised by: **Kevin GODINEAU**, **Gregory FARAUT**

1 Chapter 1: Input Data Preparation

1.1 Surface Design

The 3D surface geometry was designed using **SolidWorks**, a professional CAD software. The design represents a typical trimmed surface that would be encountered in CNC machining applications.

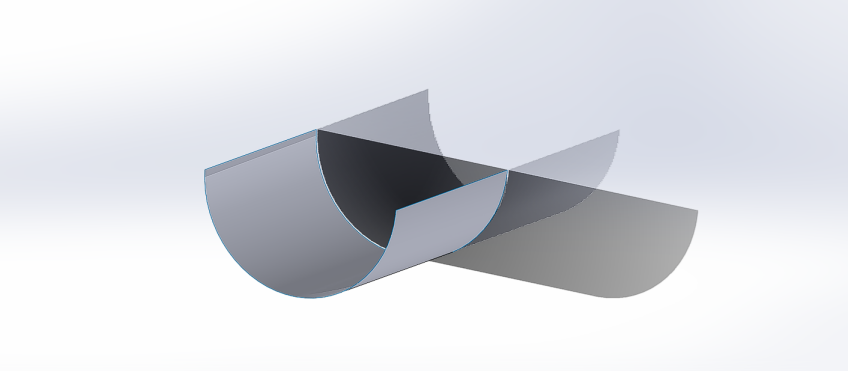


Figure 1: The 3D surface

1.2 Mesh Generation

The CAD model was imported into **ANSYS** for mesh generation. ANSYS generated a high-quality triangular mesh consisting of:

- 240 nodes (vertices)
- 422 triangular elements

The mesh was refined to ensure adequate resolution for toolpath generation while maintaining computational efficiency.

1.3 Data Extraction

ANSYS outputs the mesh data in a combined format. To make it compatible with our MATLAB toolpath generation algorithm, we used a **Python script** to separate the mesh data into two distinct files:

- `nodes.txt` — Contains node coordinates (X, Y, Z positions)
- `elements.txt` — Contains element connectivity (triangle vertex indices)

This preprocessing step ensures clean, organized input data for the MATLAB pipeline.

2 Chapter 2: Implementation Method

2.1 Coding Philosophy

The implementation follows a **modular design philosophy**, where the algorithm is divided into independent, reusable functions. Each function handles a specific task in the toolpath generation pipeline. This approach offers several advantages:

- **Maintainability** — Easy to debug and update individual components
- **Readability** — Clear separation of concerns makes the code easier to understand
- **Testing** — Each module can be tested independently

2.2 Code Structure

The implementation consists of the following MATLAB functions:

1. `main.m` — Orchestrates the entire pipeline
2. `loadMesh.m` — Reads mesh data from text files
3. `findBoundaries.m` — Detects outer boundaries and holes
4. `ConformalMapping.m` — Flattens 3D surface to 2D plane
5. `computeCotangentWeights.m` — Calculates mathematical weights for mapping
6. `generateToolpaths2D.m` — Creates offset toolpaths in 2D
7. `mapToolpathsTo3D.m` — Maps 2D toolpaths back to 3D
8. `visualizeToolpaths3D.m` — Displays final results

Each function is well-documented with input/output specifications and inline comments explaining complex operations.

3 Chapter 3: Results

First geometry

3.1 3D Surface Mesh

The input mesh loaded from ANSYS data:

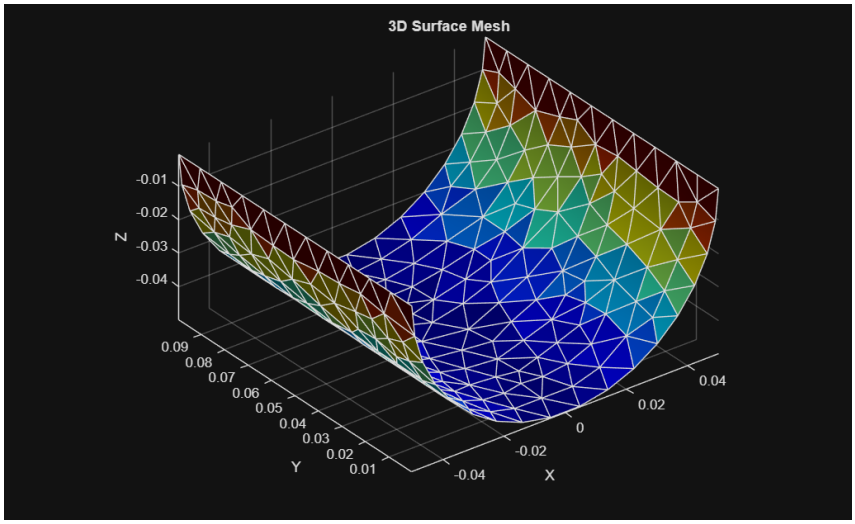


Figure 2: Original 3D surface mesh (240 nodes, 422 elements)

3.2 Boundary Detection

Detected boundary loops (1 outer boundary, 56 nodes):

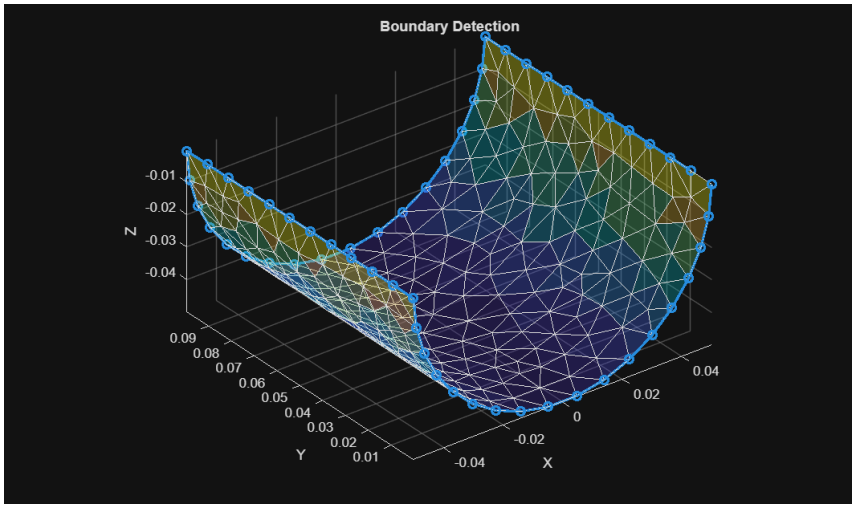


Figure 3: Detected boundary loops highlighted on the mesh

3.3 2D Conformal Mapping

The 3D surface flattened to 2D with 1.69% average distortion:

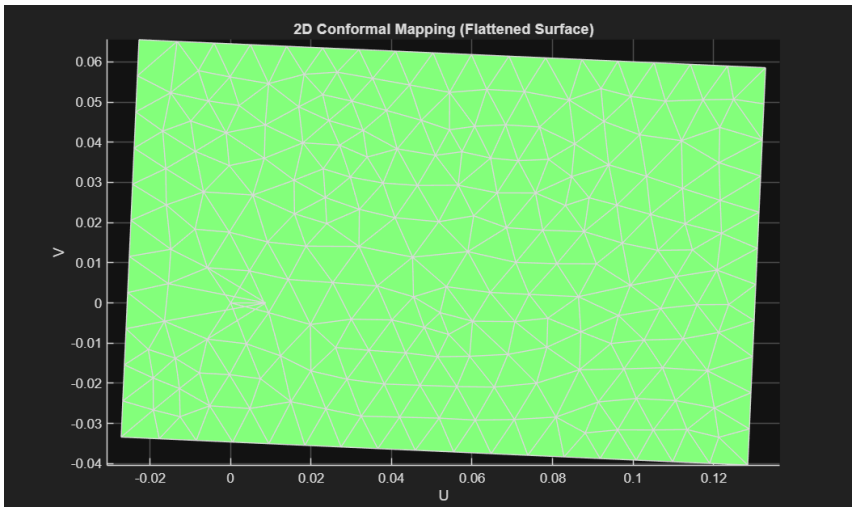


Figure 4: 2D conformal mapping result (UV coordinates)

3.4 Generated 2D Toolpaths

69 contour-parallel toolpaths in the 2D plane:

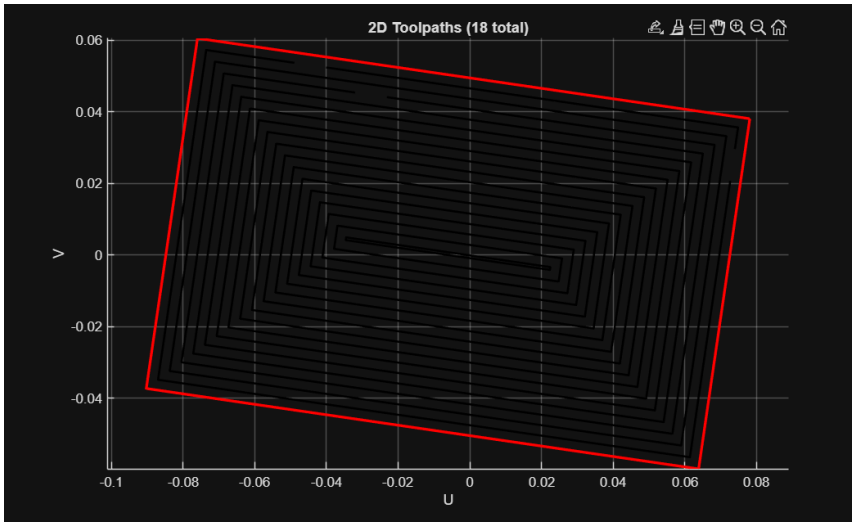


Figure 5: Generated 2D toolpaths with uniform spacing

3.5 Final 3D Toolpaths

Toolpaths mapped back to the 3D surface:

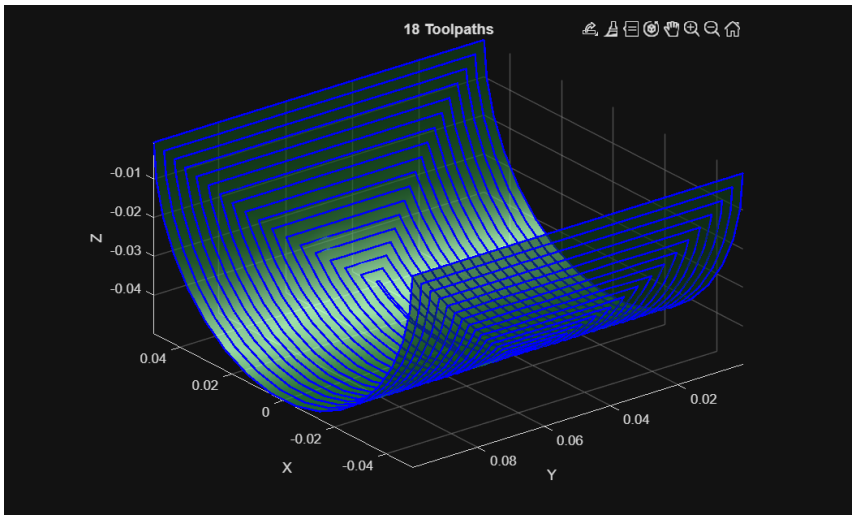


Figure 6: Final 3D toolpaths overlaid on surface mesh

Second geometry

3.6 3D Surface Mesh

The input mesh loaded from ANSYS data:

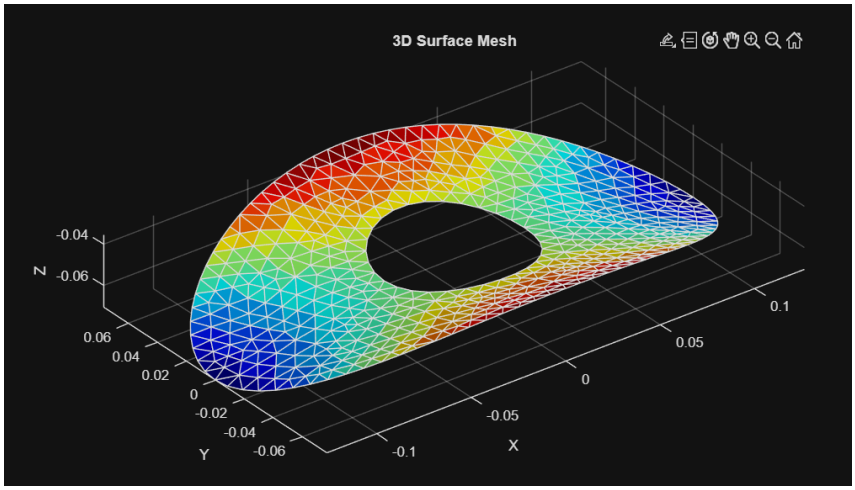


Figure 7: Original 3D surface mesh

3.7 Boundary Detection

Detected boundary loops (1 outer boundary, 56 nodes):

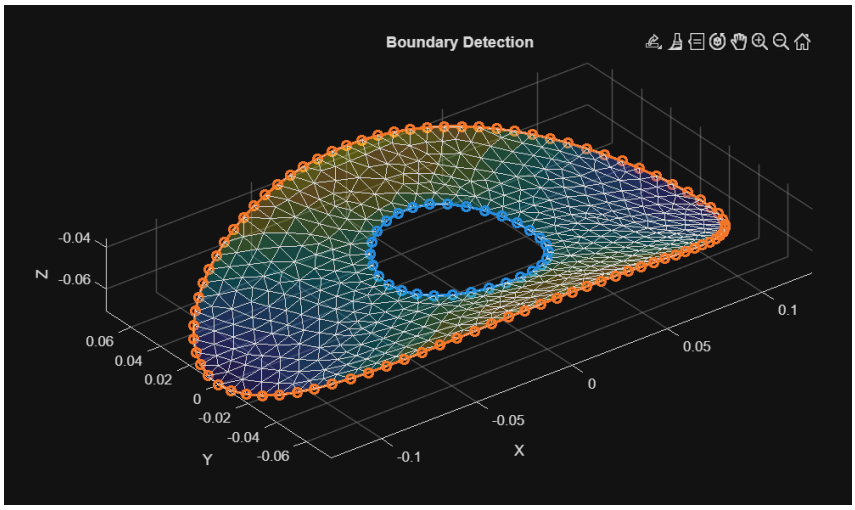


Figure 8: Detected boundary loops highlighted on the mesh

3.8 2D Conformal Mapping

The 3D surface flattened to 2D with 2.1% average distortion:

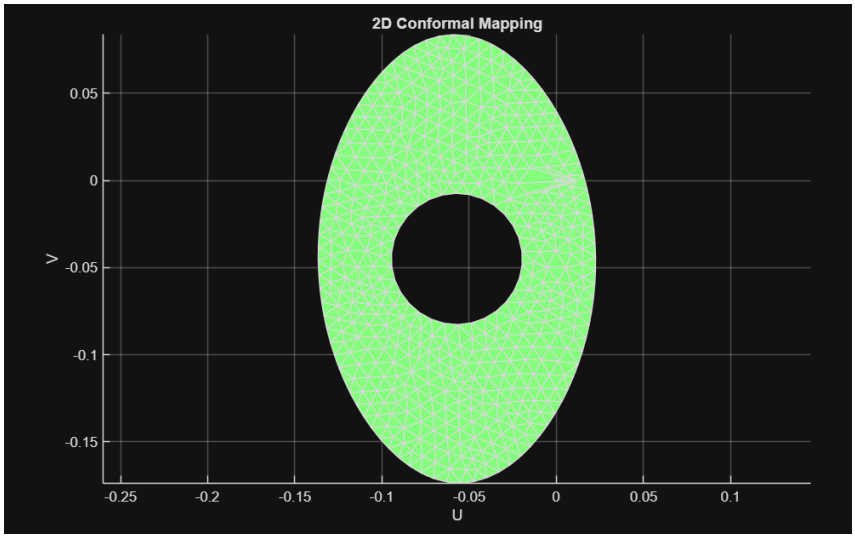


Figure 9: 2D conformal mapping result (UV coordinates)

3.9 Generated 2D Toolpaths

58 contour-parallel toolpaths in the 2D plane:

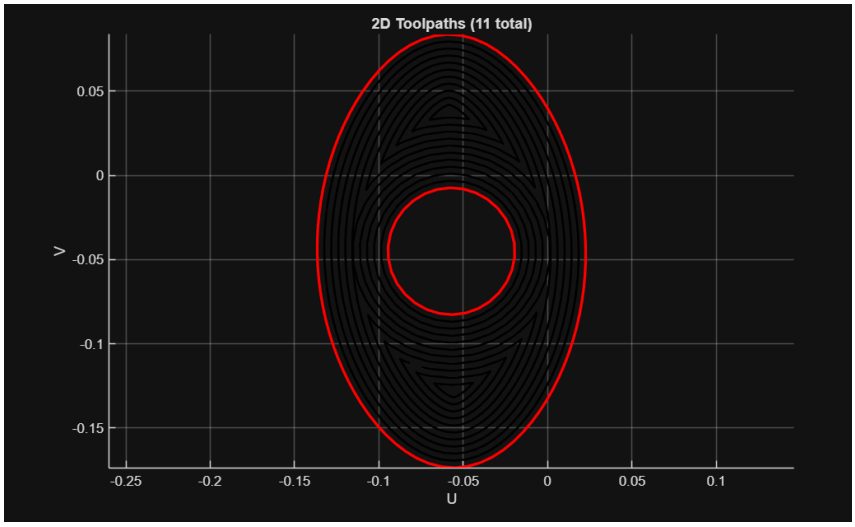


Figure 10: Generated 2D toolpaths with uniform spacing

3.10 Final 3D Toolpaths

Toolpaths mapped back to the 3D surface:

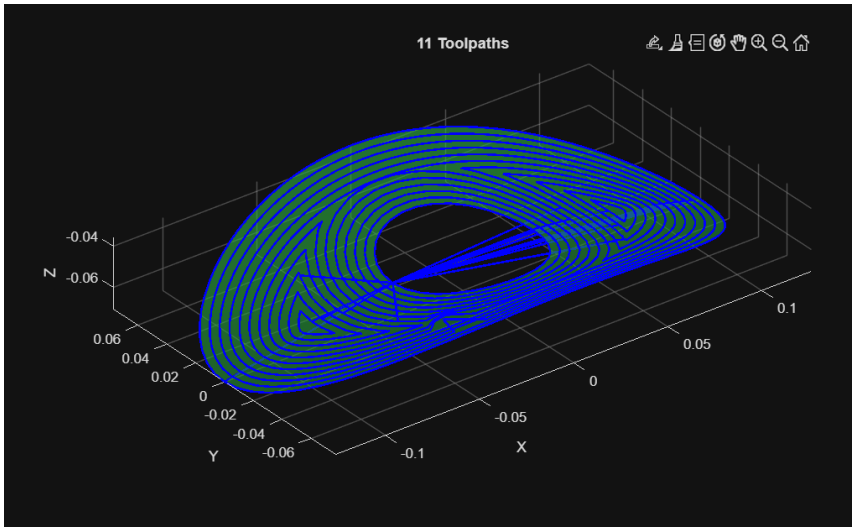


Figure 11: Final 3D toolpaths overlaid on surface mesh

4 Conclusion

4.1 Advantages of Conformal Mapping Approach

- **Simplicity** — Avoids complex 3D interference detection by working in 2D
- **Efficiency** — Fast computation due to simple 2D offsetting operations
- **Robustness** — Automatic handling of self-intersections via `polybuffer`
- **Quality** — Low distortion (1.69%) preserves surface characteristics
- **Boundary conformity** — Toolpaths naturally follow surface boundaries

4.2 Limitations

- **Distortion artifacts** — Fixed anchor points introduce minor distortion in central regions
- **3-axis only** — No tool orientation optimization (not suitable for 5-axis machining)
- **Non-developable surfaces** — Some distortion is unavoidable for complex geometries
- **Computational cost** — Solving large sparse linear systems can be expensive for very large meshes

4.3 Alternative Methods in Literature

Several alternative approaches exist for toolpath generation on 3D surfaces:

- **Iso-parametric methods** — Follow parametric curves, but don't conform well to trimmed boundaries
- **Iso-scallop height methods** — Maintain constant scallop height, better for surface finish but computationally expensive
- **Space-filling curves** — Hilbert or fractal curves for complete coverage, but complex implementation
- **Direct 3D offsetting** — Works directly in 3D space, handles complex geometry but requires robust interference detection
- **Medial axis transform** — Generates centerline-based toolpaths, good for pockets but less suitable for free-form surfaces
- **Machine learning approaches** — Recent methods use neural networks for toolpath optimization, but require extensive training data

The conformal mapping approach offers a good balance between simplicity, efficiency, and quality for trimmed surfaces without interior complexity.

Reference:
Sun, Y., Ren, F., Zhu, X., & Guo, D. (2012). Contour-parallel offset machining for trimmed surfaces based on conformal mapping with free boundary. *The International Journal of Advanced Manufacturing Technology*, 60(1-4), 261-271.