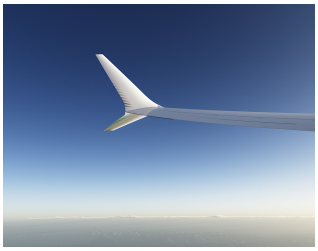
CHAPTER 1

EXTENSION OF THE RSVS TO THREE DIMENSIONS

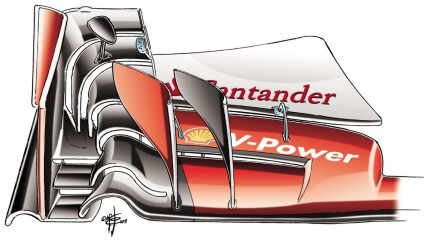
While there exists a wide range of parameterisation methods for aerodynamics in two dimensions, in three dimensions the parameterisation offering is dominated by free-form deformation (FFD) methods. Thanks to their deformative formulation, an existing discrete geometry can be used as the starting point, maintaining its properties. By preserving surface characteristics, deformative methods allow a high level of accuracy provided the initial geometry is of high quality; however this precludes topological design of aerodynamic bodies. Generating the initial geometries and discretisations is a significant challenge in its own right, with the water-tightness of surfaces not generally guaranteed by the CAD tools used to define them.

These challenges mean that aerodynamic topological optimisation of an entire aircraft or wing is unlikely to be a reality in the near or medium term. However, there is scope for the aerodynamic topological design of local features; topological optimisation of wing tips would allow feathered or split winglets more complex than that on the Boeing 737-MAX to be explored (Figure 1.1a and Figure 1.1c). An effective topological aerodynamic optimisation framework also offers the possibility of radically new designs in applications to: Formula 1 (Figure 1.1b), unmanned aerial vehicles, commercial strut-braced wing design, and, internal engine design. No current optimisation framework for external aerodynamics supports the exploration of 3-dimensional topological changes, in large part because of the dominance of deformative parameterisation methods.

The area constrained length minimisation formulation of the restricted snakes volume of solid (RSVS) offers a natural extension to 3D as the minimisation of the surface area of a geometry under volume constraints. During design and testing of the two dimensional



(a) Winglet of the Boeing 737-MAX, from Boeing [4].



(b) Front wing of the SF-15 Ferrari F1 car, from Giorgio Piola [30].

(c) Eagle in flight.

Figure 1.1: Examples of complex topology in aerodynamic applications.

RSVS; features which would extend to 3D were prioritised. The flexibility and generality of the RSVS formulation, useful features in two dimension, becomes necessary for efficient exploration of 3 dimensional design spaces where isotropic tessellation of the geometric space would be prohibitively expensive.

A new ‘restricted surface (r-surface)’ tool is developed, generalising the restricted snake (r-snake) to higher dimensions, this new method allows the robust evolution and containment of objects of arbitrary topology. This r-surface is then integrated with a marching procedure minimising the area of the surface under localised volume fraction constraints to form the restricted surface volume of solid (3D-RSVS) parameterisation. Properties of this new parameterisation are discussed, highlighting similarities with the generation of constant mean curvature (CMC) surfaces and minimal surfaces. Design of volume of solid (VOS) layouts is treated in Section 1.4.2, and finally the 3D-RSVS is integrated into an optimisation framework.

1.1 Restricted Surfaces Volume of Solid for 3-Dimensional Aerodynamic Parameterisation

This section presents how the 3-dimensional restricted surface volume of solid (3D-RSVS) parameterisation translates sets of volume fraction design variables specified on a fixed grid into closed surfaces of varying topology. For optimisation frameworks to exploit the 3D-RSVS efficiently, this process must reliably produce smooth features at a resolution below the grid on which VOS values are defined. To achieve this level of smooth control, the 3D-RSVS profile is defined as: the closed surface of minimum area that will match the volumes of the design variables. It is built using a restricted surface (r-surface). The r-surface is a method developed in this thesis for “vertex marching" which allows efficient

topology handling and is tolerant of any layout of VOS design variables. The 3D-RSVS is implemented in C++, the code is made available by the author on GitHub[1] under the LGPL-3.0 license[2].

1.1.1 Governing Equations of the 3D-RSVS

By using a formulation which is very closely related to the two dimensional parameterisation, the 3D-RSVS maintains many of the benefits of that parameterisation, notably: intuitiveness, homogeneity, smoothness, compactness and flexibility. The parameterisation process is shown in Figure 1.2 with slices through the domain indicating the volume fractions.

The 3-dimensional RSVS geometries are defined as the surface with the smallest area matching the VOS in every cell. The mathematical formulation of this problem is given in Equation 1.1. This system is analogous to the effect of a tensile force “shrink-wrapping" the required VOS in each cell. The general form of the 3-dimensional RSVS problem is developed for a closed surface S which is constrained in m design cells (C_j) to have a specified volume fraction V_j. These variables are represented graphically in Figure 1.2b for a 2 dimensional grid.

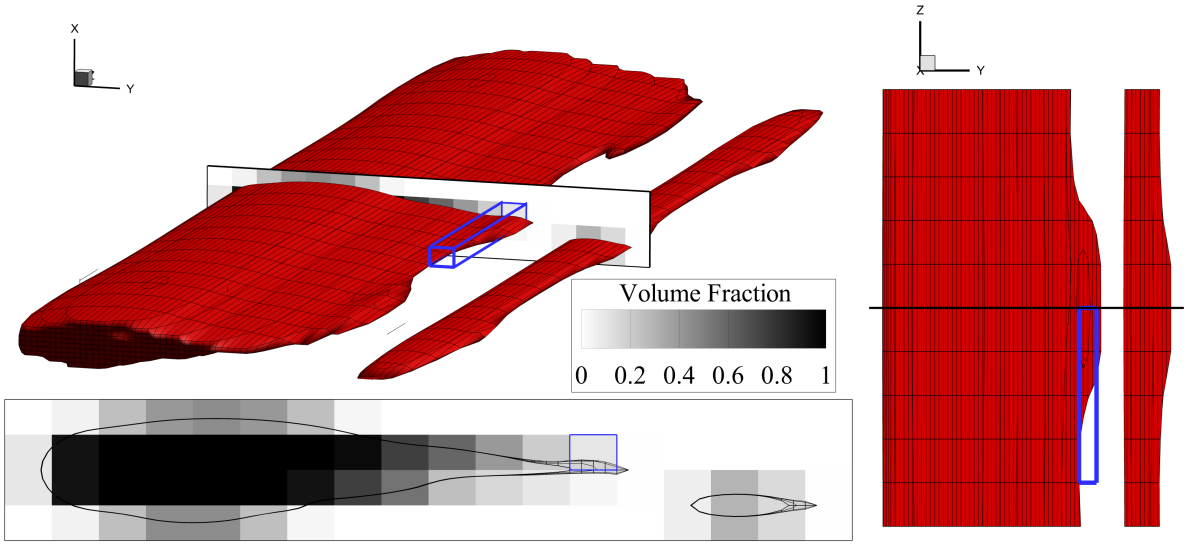
$$\begin{aligned} \min \quad & \iint_{S\to\mathbf{x}(t,u)} \left\| \frac{\partial \mathbf{x}}{\partial t} \times \frac{\partial \mathbf{x}}{\partial u} \right\| dt\,du \\ \text{s.t.} \quad & \iiint_{(S\cap C_j)} dx\,dy\,dz = V_j \qquad \forall j \in \{0,\cdots,m\} \end{aligned} \tag{1.1}$$

The rules above are the natural extension to 3-dimensions of the 2D-RSVS: the length minimisation has become a surface minimisation and the area constraints become volume constraints. The design variables that control the surface are volume fractions specified in each cell of a *design grid* which remains unchanged during an aerodynamic topology optimisation procedure. The next sections detail how the mathematical program is solved using restricted surfaces to produce an effective shape and topology parameterisation method.

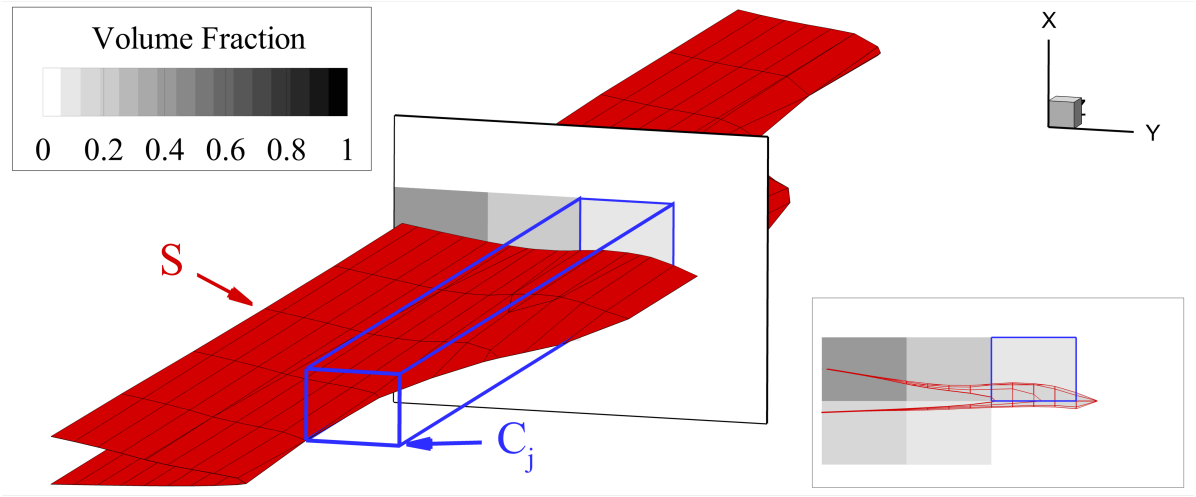
Unlike the two dimensional case; the 3D-RSVS does not support exact analytical solutions. The 3D-RSVS surfaces are part of a class of problems known as constant mean curvature (CMC), a super-class of “minimal surfaces”, for which explicit analytical

[1] `https://github.com/payoto/rsvs3d` and `https://github.com/farg-bristol/rsvs3d`

[2] Available at: `https://opensource.org/licenses/lgpl-3.0.html` accessed on 05/06/2019.



(a) VOS design variables as grey-scale and 3D-RSVS surface; 1 corresponds to a completely full cell and 0 an empty cell.



(b) VOS definitions for Equation 1.1

Figure 1.2: Example RSVS profile and design grid with label definitions for the governing equation (Equation 1.1).

solutions are available for specific boundary conditions [28]. In the RSVS process the boundary conditions, as understood in the study of minimal surfaces, are not known explicitly as they are the result of the marching of the snake or surface. This additional level of complexity makes general analytical solutions to the 3D-RSVS unlikely; however comparison of specific parameterised layouts to existing analytical solutions is performed in Section 1.2. A more substantial discussion of minimal surfaces and CMC surfaces is done in Section 1.4.1.

1.1.2 The Restricted Surface Method for 3D Topology Evolution

Development of a 3-dimensional, volume of solid based, topologically flexible parameterisation requires an efficient method for evolving topologically complex geometries. The 2D-RSVS used restricted snakes, a type of parametric active contour developed by Bischoff et al. [3, 19]. Previous work in the extension of parametric active contour methods to 3 dimensions have been successful, notably the development of topologically flexible T-surfaces by McInerney et al. [22] for medical image segmentation. This subsection outlines the extension of the r-snake to evolve as a surface on 3-dimensional grids, and how this may be used to solve the 3D-RSVS governing equation. A complete description of the development of these r-surfaces is provided is Section 1.3.

1.1.2.1 Topology evolution of polyhedra

To build the 3D-RSVS parameterisation method the restricted surface must be evolved until it solves the governing equation. The restricted surface is a vertex marching procedure where the control points (called snaxels) are constrained to move on a predefined grid, as a consequence properties of the snaking grid controls the number of snaxels and the resolution of the geometry. By marching the snake on a grid finer than the VOS grid, smooth features below the resolution of the volume design variables can be recovered. This allows a high degree of geometric flexibility with few design variables.

In order to maintain the water-tightness of the surface the connectivity elements between snaxels, are restricted. The original rules developed by Bischoff and Kobbelt [19] for contours have been generalised to surfaces in 3D space into the following:

- No 2 snaxels connected by a r-surface edge are on the same *snaking grid* edge;
- No 2 r-surface edges connected by a snaxel are in the same *snaking grid* face;
- No 2 r-surface faces connected by r-surface edge are in the same *snaking grid* cell.

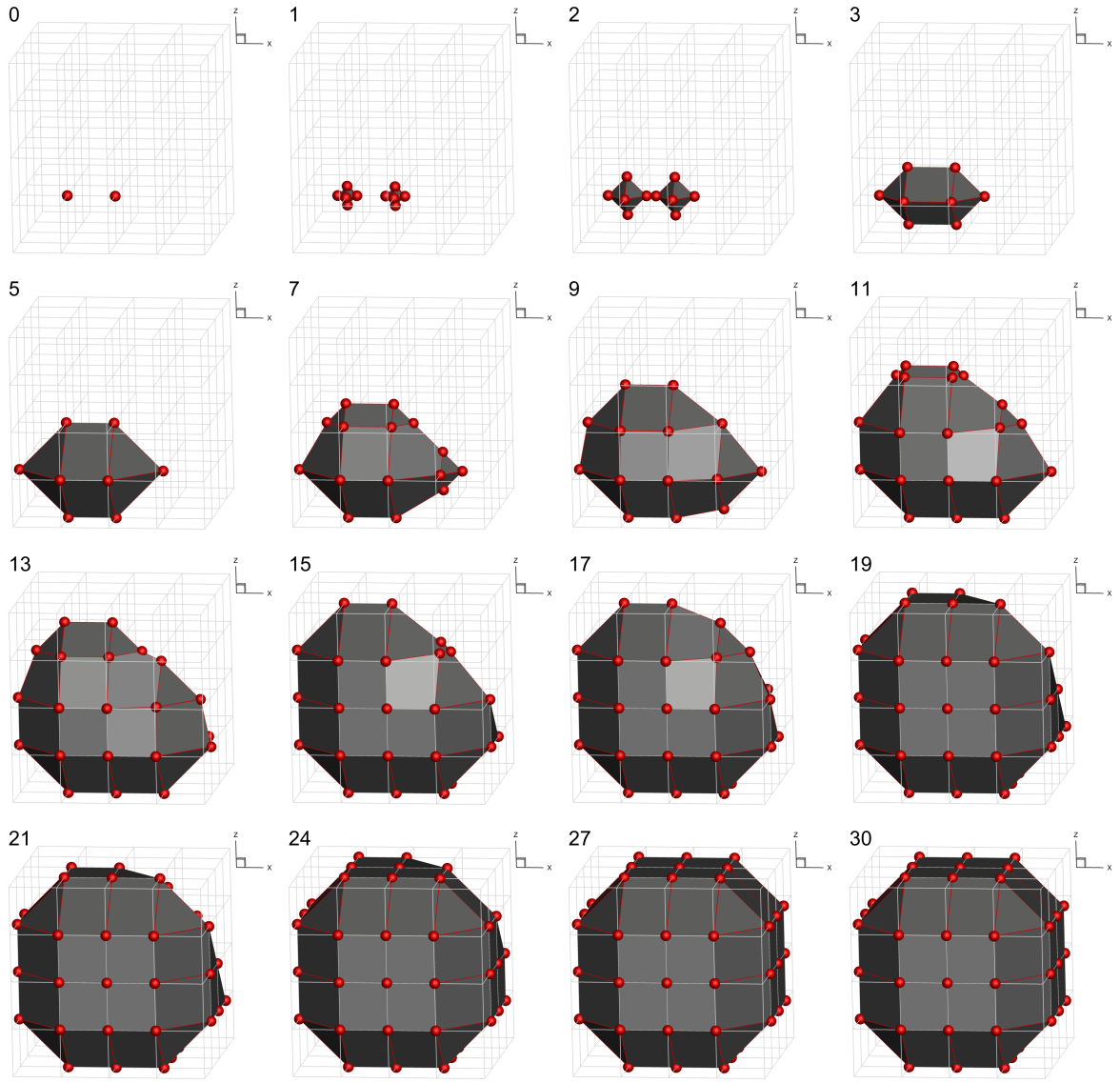


Figure 1.3: Evolution of a restricted surface in a 4^3 snaking grid spawned in two locations under a unit velocity field.

These connectivity rules, can be used to maintain a meaningful surface when two surfaces collide or when a surface crosses through a vertex of the snaking grid. Figure 1.3 shows the evolution of restricted surfaces initialised from two vertices in a 4^3 snaking grid. The two surfaces collide on the third step, and the connectivity is adjusted using the rules specified above.

While the rules for building the r-surface guarantee the formation of water-tight surfaces, it does not guarantee that faces will be flat. This is because the r-surface is controlled by the positioning of its vertices with the rest of the geometry derived from

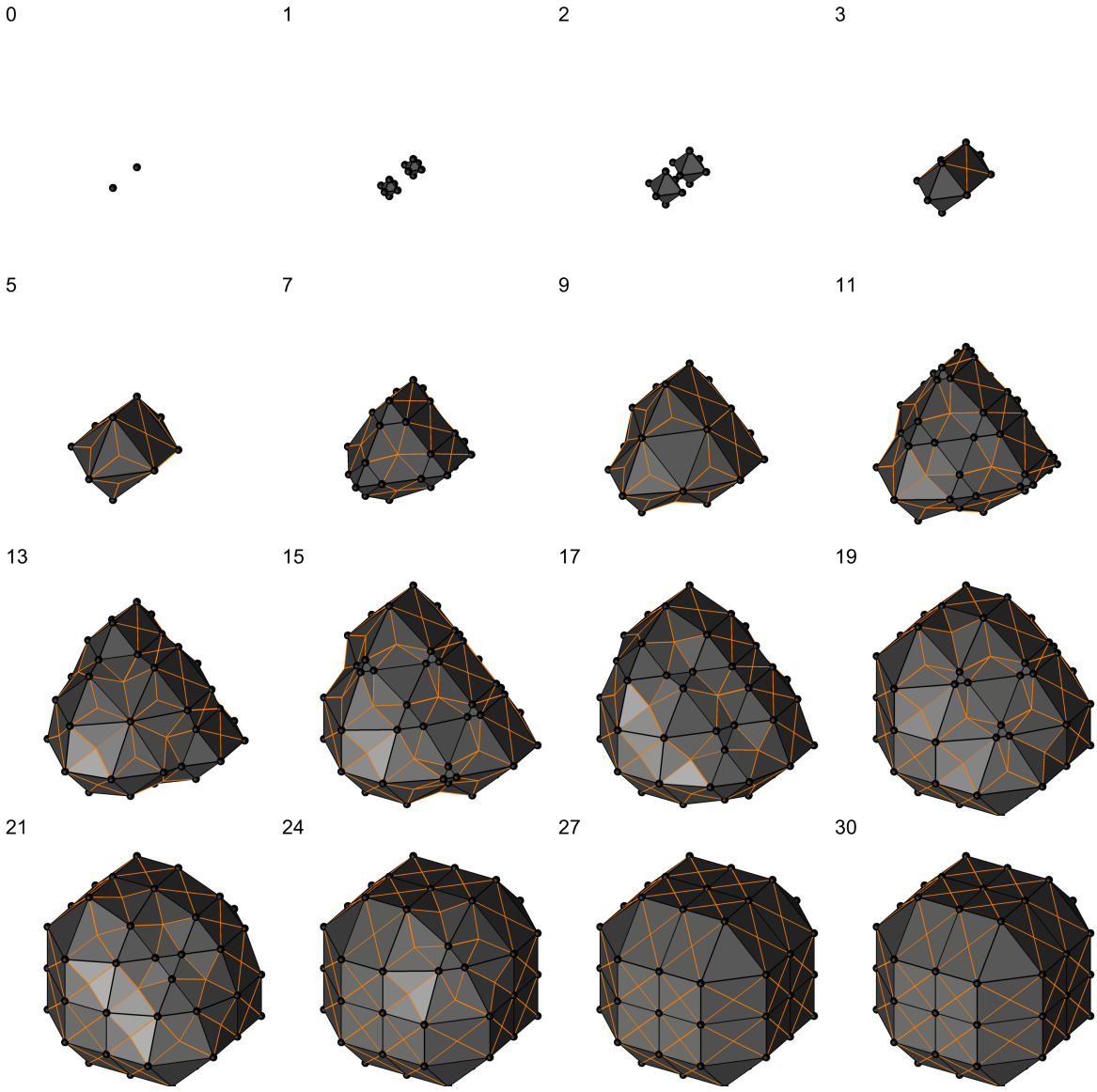
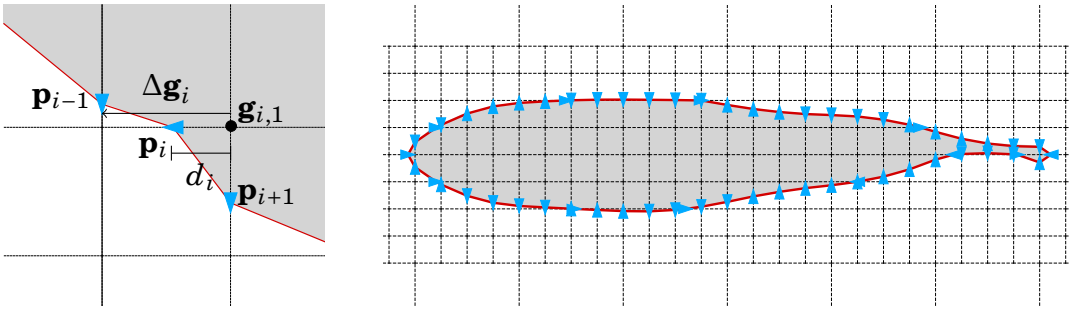


Figure 1.4: Evolution of the triangulation (edges in orange) of a restricted surface (in black) using the centroid defined in Equation 1.3. Step numbers are in the top left corner.

the connectivity information forced by the snaking grid. Flat surfaces are required for the reliable calculation of volume and area of the polyhedron, faces with more than 3 edges need to be triangulated. Consistency and smoothness of the triangulation through changes of connectivity is achieved by triangulating faces through point $\bar{\mathbf{c}}$; this point is the mean position of face vertices normalised by edge length. Figure 1.4 shows the evolution of the chosen triangulation process through the evolution of the restricted surface.



(a) Close-up view of snake and snaxel

(b) Full view of a closed r-snake

Figure 1.5: R-snake contour (in red) with snaxels (in blue) evolving on the snaking grid (dashed line) [Repetition of Figure **??**].

1.1.2.2 Integration of the r-surface with the area minimisation problem

To drive the position of the restricted surface the original continuous area minimisation problem (Equation 1.1) is discretised in terms of the r-surface and snaxel variables, becoming the mathematical program of Equation 1.2. The discretisation process expresses the integrals in terms of the triangulated r-surface using six properties of the r-surface geometry and the snaxel positions. The first three of these properties are part of the movement algorithm; the last three properties of the snaxels are derived from connectivity and grid information. These properties are: the snaxel index (i), used to reference it in all operations; the normalised position along an edge ($d_i \in [0,1]$); the scalar velocity along that edge ($v_i \in \mathbb{R}$); the snaxel position in Cartesian coordinates ($\mathbf{p}_i$); and; the direction of travel of the snaxel ($\Delta\mathbf{g}_i$) and the vertex of origin ($\mathbf{g_{i,1}}$). These properties are represented graphically for a restricted snake in Figure 1.5a. They are used to calculate and differentiate $A_{S,k}$ and $V_{S,k}$, respectively the area and volume contributed by each triangle forming the polyhedron. These definitions are integrated into Equation 1.1 to form the discrete mathematical program used to drive the r-surface in Equation 1.2.

$$\begin{aligned} &\min_{\mathbf{d}} \sum_{k=1}^{q} A_{S,k}\left(\mathbf{p}_{0,k},\mathbf{p}_{1,k},\mathbf{p}_{2,k}\right) \quad \text{with } \mathbf{p}_{i,k}(\mathbf{d}) \\ &\text{s.t.} \sum_{k=j_S(1)}^{j_S(q_{j_S})} V_{S,k}\left(\mathbf{p}_{0,k},\mathbf{p}_{1,k},\mathbf{p}_{2,k}\right) + \sum_{k=j_C(1)}^{j_C(q_{j_C})} V_{C,k}\left(\mathbf{g}_{0,k},\mathbf{g}_{1,k},\mathbf{g}_{2,k}\right) = V_j \end{aligned} \tag{1.2}$$

Building an RSVS surface requires the positions $\mathbf{d}$ of the r-surface snaxels satisfying Equation 1.2 to be found. As is the case in 2-dimensions the objective function and the

constraint are readily differentiable. This is critical to solving the area minimisation governing equation as it allows the use of efficient gradient based optimisation method. While the area and volume could be differentiated by hand, the task would be tedious and error prone. The differentiation of $A_{S,k}$, and $V_{S,k}$ with regard to $\mathbf{p}_{i,k}$ was carried out for triangles using the MATLAB symbolic toolbox. This allows C code to be directly generated for the mathematical functions, ensuring that no mistake is made when calculating Jacobian and Hessian. The same process is followed for the derivatives of $\mathbf{p}_{i,k}$ and $\bar{\mathbf{c}}$ with regard to $\mathbf{d}$.

This formulation has the benefit of being very general, it can be tackled on an arbitrary volume grid with any underlying snaking grid with any optimisation method. This generality guarantees a high degree of flexibility in the range of shapes that can be represented. Later sections will show how the r-surface is implemented and how the 3D-RSVS parameterisation can be constructed using a Newton step sequential quadratic programming (SQP) procedure. The next section shows parameterisation results using the 3D-RSVS on Cartesian grids and an empirical study of the behaviour of the parameterisation.

1.2 3 Dimensional Parameterisation Results

This section presents profiles generated using the Restricted Surface method driven by SQPsolving the area minimisation, volume constrained, governing equation. The VOS values are manually specified; these results serve to validate the implementation of the 3D-RSVS, and to highlight the topological flexibility of 3D-RSVS on small layouts of Volume of Solid cells.

1.2.1 Practical Surface Generation

The RSVS rules only specify how to evolve a surface but provide no guidance regarding the initialisation. For aerodynamic applications and more generally the design of external boundaries it is effective to start at the faces which touch a void and a non-empty volume cell. This provides fast convergence and intuitive behaviour to a designer. Internal voids can then be created if the restricted surface has failed to explore non-full volume cells.

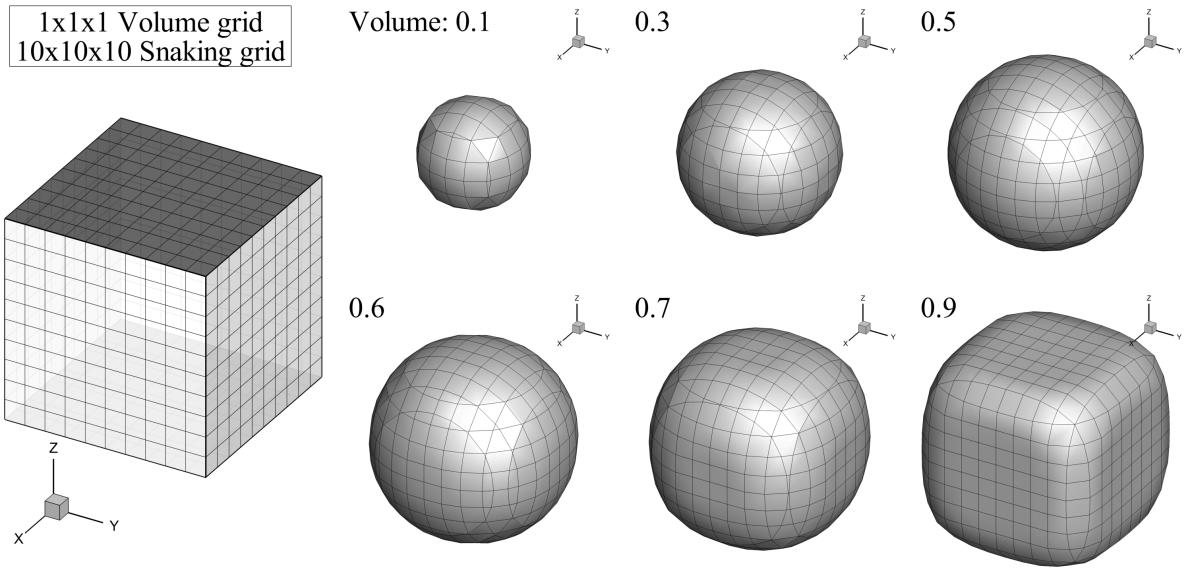


Figure 1.6: Geometries defined by a single volume cell with VOS values from 0.1 to 0.9.

1.2.2 Validation of the RSVS Parameterisation

While an analytical study of the 3D-RSVS problem has not yet been performed it is natural to expect its behaviour to be similar to the two dimensional parameterisation: for shapes defined by few design variables the geometry is hypothesised to be spherical patches. To validate the implementation of the RSVS shapes designed with a single VOS cell were generated and are shown in Figure 1.6. This single volume cell is refined into a 10^3 snaking grid. These shapes clearly show that the RSVS converges to spherical profiles up to volume fractions of 0.5. Beyond that volume fraction, the surface comes in contact with the VOS cell boundaries and starts to form a cuboid with round edges.

To supplement the qualitative observations from Figure 1.6, the volume and area of 3D-RSVS bodies is compared to spheres of equivalent volumes in Table 1.1. This table shows that for low values of requested volume fraction (up to 0.5) volume convergence is good ($\lessapprox 10^{-5}$). As the required VOS approaches 0.5 the area approaches that of a sphere, the area error dropping as low as 0.34% for a sphere of volume 0.5. This is expected: the discretisation of the sphere produced by the 3D-RSVS depends on the number of intersections the geometry has with the background snaking mesh. As the object gets smaller, the number of intersections reduces and the discretisation becomes worse. This observation is confirmed by generating a sphere of volume 0.5 on a finer snaking grid with 24^3 cells; on this snaking grid the area match was even closer at 0.09% (Table 1.1).

For objects coming in contact with the edges of the design space volume convergence

Table 1.1: Numerical comparison of the areas and volumes of 3D-RSVS geometries and spheres of the same target volume.

Fig.	Expected Sphere Properties			RSVS geometry		Error		Observation
	V	Diameter	Area	Volume	Area	Volume	Area	
1.6	0.1	0.576	1.042	0.100	1.054	3.46E-13	-1.16%	
	0.2	0.726	1.654	0.200	1.672	3.23E-06	-1.08%	
1.6	0.3	0.831	2.167	0.300	2.176	-2.03E-07	-0.40%	
	0.4	0.914	2.625	0.400	2.638	2.62E-05	-0.48%	
1.6	0.5	0.985	3.046	0.500	3.057	-2.45E-07	-0.34%	
1.6	0.6	1.046	3.440	0.599	3.456	1.02E-03	-0.44%	at border
1.6	0.7	1.102	3.813	0.700	3.871	3.52E-04	-1.54%	at border
	0.8	1.152	4.168	0.798	4.362	2.03E-03	-4.66%	at border
1.6	0.9	1.198	4.508	0.893	4.840	8.28E-03	-7.36%	at border
1.10	0.75	0.895	5.030	0.750	5.052	2.70E-04	-0.45%	3DVs 2 spheres
	0.5	0.985	3.046	0.500	3.049	-1.04E-06	-0.09%	24^3 snaking grid

is due to the different treatment of snaxels at the edge of the design space. Indeed these cannot be treated as normal design variables for the area minimisation process as they cannot move further outwards but still must be free to move back inwards. A change to the solver of the quadratic program might be needed to support inequality constraints for those snaxels which can only move in one direction. Approaches similar to QPOPT (the internal quadratic solver of SNOPT) [15] are being investigated to improve convergence speed.

1.2.3 Generation of Shapes of Aerodynamic Interest

With the implementation of the 3D-RSVS equations validated, manually specified aerodynamic surfaces were generated as a test for the smoothness of the geometries produced by multiple design variables. The surfaces chosen were the Sears-Haack body, the truncated Sears-Haack body and a wing with aerofoil cross-sections.

Figure 1.7 shows a Sears-Haack body, and the truncated Sears-Haack body is presented in Figure 1.8. These surfaces use a [10, 2, 2] layout of VOS cells and 4^3 snaking refinement. As was the case in 2 dimensions the RSVS produces mostly smooth profiles but can be forced to produce a sharp corner or a sharp edge by using small volume fractions, providing accurate positioning of the leading and trailing edges. In Figures 1.7 and 1.8, the final volume error is displayed on the background volume mesh; showing that the 3D-RSVS process can very precisely match the volume fractions ($e_v \in [10^{-13},\ 10^{-6}]$).

Drag minimisations of wings are common cases within the aerodynamic shape opti-

Table 1.2: Numerical comparison of the areas and volumes of 3D-RSVS geometries to spheres of the same target volume.

Figure	Design space size	Design variables	VOS layout	Volume fractions	Expected Sphere Properties			RSVS geometry		Error	
					V	Diameter	Area	Volume	Area	Volume	Area
fig. 1.6 up-le	[1, 1, 1]	1	[1, 1, 1]	0.1	0.1	0.576	1.042	0.100	1.054	3.46E-13	-1.16%
	[1, 1, 1]	1	[1, 1, 1]	0.2	0.2	0.726	1.654	0.200	1.672	3.23E-06	-1.08%
fig. 1.6 up-mid	[1, 1, 1]	1	[1, 1, 1]	0.3	0.3	0.831	2.167	0.300	2.176	-2.03E-07	-0.40%
	[1, 1, 1]	1	[1, 1, 1]	0.4	0.4	0.914	2.625	0.400	2.638	2.62E-05	-0.49%
fig. 1.6 up-r	[1, 1, 1]	1	[1, 1, 1]	0.5	0.5	0.985	3.046	0.500	3.057	-2.45E-07	-0.35%
fig. 1.6 lo-le	[1, 1, 1]	1	[1, 1, 1]	0.6	0.6	1.046	3.440	0.599	3.456	1.02E-03	-0.45%
fig. 1.6 lo-mid	[1, 1, 1]	1	[1, 1, 1]	0.7	0.7	1.102	3.813	0.700	3.871	3.52E-04	-1.54%
	[1, 1, 1]	1	[1, 1, 1]	0.8	0.8	1.152	4.168	0.798	4.362	2.03E-03	-4.66%
fig. 1.6 lo-r	[1, 1, 1]	1	[1, 1, 1]	0.9	0.9	1.198	4.508	0.893	4.840	8.28E-03	-7.36%
	[1, 1, 1]	2	[2, 1, 1]	0.05	0.05	0.457	0.656	0.050	0.670	2.88E-04	-2.12%
	[1, 1, 1]	2	[2, 1, 1]	0.1	0.1	0.576	1.042	0.100	1.058	-7.64E-07	-1.54%
	[1, 1, 1]	2	[2, 1, 1]	0.15	0.15	0.659	1.365	0.150	1.374	2.77E-03	-0.67%
	[1, 1, 1]	2	[2, 1, 1]	0.2	0.2	0.726	1.654	0.200	1.669	-1.52E-09	-0.90%
	[1, 1, 1]	2	[2, 1, 1]	0.25	0.25	0.782	1.919	0.250	1.930	-7.25E-09	-0.55%
	[1, 1, 1]	2	[2, 1, 1]	0.3	0.3	0.831	2.167	0.300	2.180	-1.54E-06	-0.61%
	[1, 1, 1]	2	[2, 1, 1]	0.35	0.35	0.874	2.402	0.350	2.415	-2.25E-09	-0.54%
	[1, 1, 1]	2	[2, 1, 1]	0.4	0.4	0.914	2.625	0.400	2.668	5.95E-04	-1.61%
	[1, 1, 1]	2	[2, 1, 1]	0.45	0.45	0.951	2.840	0.450	2.853	6.95E-07	-0.48%
	[1, 1, 1]	2	[2, 1, 1]	0.5	0.5	0.985	3.046	0.500	3.063	4.08E-07	-0.53%
	[2, 1, 1]	2	[2, 1, 1]	0.05	0.1	0.576	1.042	0.100	1.054	4.78E-04	-1.13%
	[2, 1, 1]	2	[2, 1, 1]	0.1	0.2	0.726	1.654	0.200	1.667	1.69E-03	-0.78%
	[2, 1, 1]	2	[2, 1, 1]	0.15	0.3	0.831	2.167	0.300	2.178	6.99E-04	-0.48%
	[2, 1, 1]	2	[2, 1, 1]	0.2	0.4	0.914	2.625	0.399	2.660	2.35E-03	-1.33%
	[2, 1, 1]	2	[2, 1, 1]	0.25	0.5	0.985	3.046	0.500	3.060	1.21E-05	-0.44%
	[2, 1, 1]	2	[2, 1, 1]	0.3	0.6	1.046	3.440	0.600	3.455	2.69E-04	-0.43%
	[2, 1, 1]	2	[2, 1, 1]	0.35	0.7	1.102	3.813	0.699	3.841	1.19E-03	-0.76%
	[2, 1, 1]	2	[2, 1, 1]	0.4	0.8	1.152	4.168	0.799	4.220	1.73E-03	-1.25%
	[2, 1, 1]	2	[2, 1, 1]	0.45	0.9	1.198	4.508	0.899	4.602	1.57E-03	-2.08%
	[2, 1, 1]	2	[2, 1, 1]	0.5	1	1.241	4.836	0.998	4.980	1.69E-03	-2.98%
fig. 1.10	[3, 1, 1]	3	[3, 1, 1]	[0.3 0.1 0.3]	0.75	0.895	5.030	0.750	5.050	2.70E-04	-0.46%

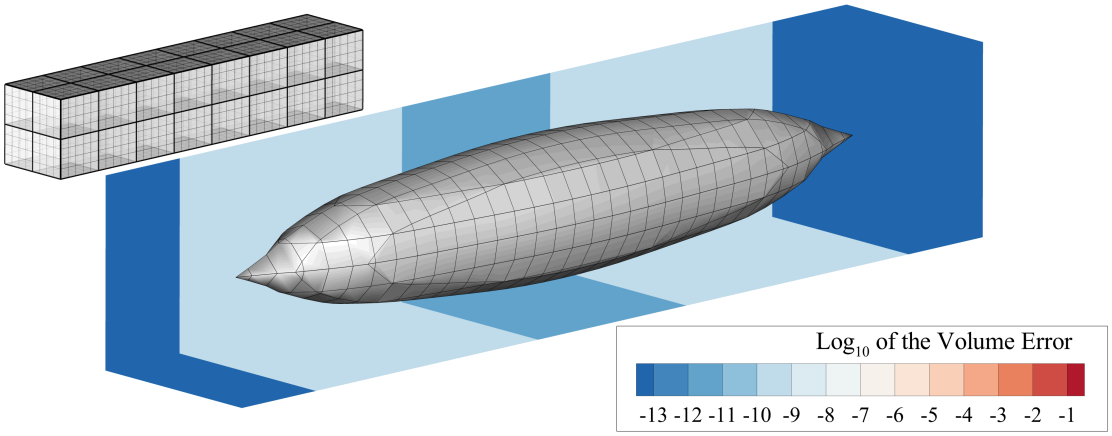


Figure 1.7: Sears-Haack body represented using 40 VOS cells in [10, 2, 2] layout. The colour in the colours in the background present the level of convergence of the r-surface on the correct volume.

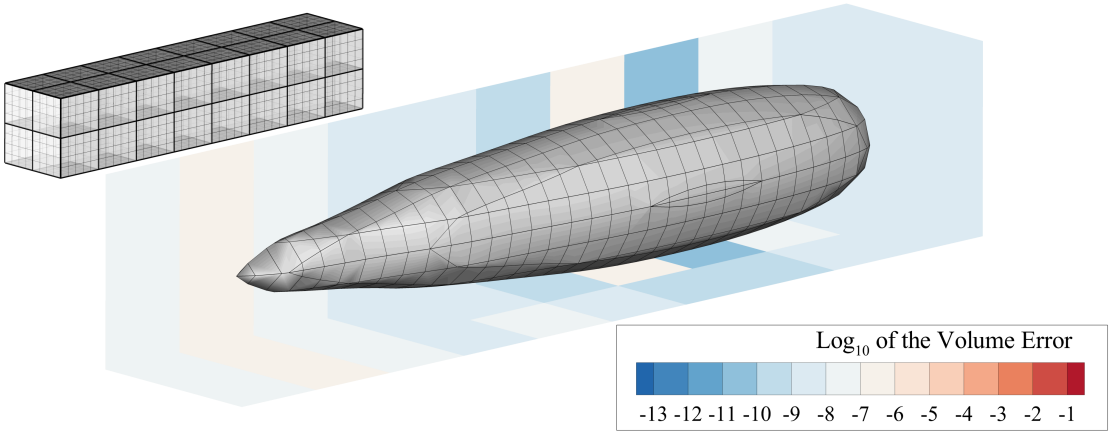


Figure 1.8: Truncated Sears-Haack body represented using 40 VOS cells in [10, 2, 2] layout. The colour in the colours in the background present the level of convergence of the r-surface on the correct volume.

misation community [8, 18]. Figure 1.9 presents a coarse representation of a wing using a [2,5,6] layout of design variables. This provides 10 volume fraction values to design the cross-section of the wing at six span locations. One of the side effects of building surfaces of minimum area is that long and slender profiles are not initially possible. To allow elongated bodies, the longer dimension of the surface needs to be de-weighted in terms of area. This can be achieved either inside the shape generation by multiplying the coordinates by individual weights or by externally altering the aspect ratio of the design grid.

Three dimensional optimisation usually relies on deformative methods starting

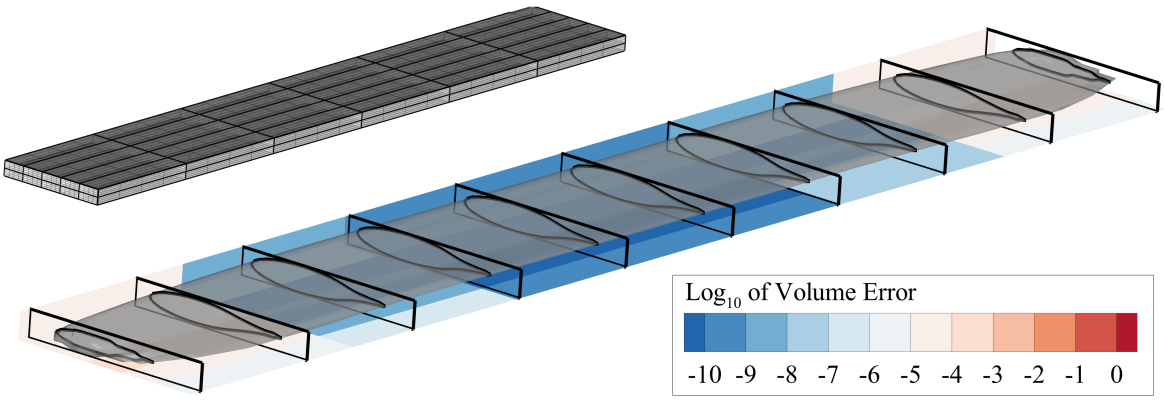


Figure 1.9: Coarse wing represented using 60 VOS cells in [2, 5, 6] layout. The colour in the colours in the background present the level of convergence of the r-surface on the correct volume.

from a high quality discretisation. This presents a challenge in evaluating the RSVS: as a constructive parameterisation method it cannot easily benefit from an existing geometry. Three approaches are envisaged to resolve this issue: progressive design space refinement; the integrated parameterisation approach which was pioneered in Section **??**; or, partial design space representations. Of these, hierarchical design variables have been shown to be effective in three dimensions to optimise aerodynamic features at a range of geometric scales concurrently [20, 21]. Despite these possibilities, design of an entire wing is not the primary use case of the RSVS: the RSVS will be targeted at cases where its topological flexibility, is an asset not a drawback.

1.2.4 Topological Flexibility of the 3 Dimensional RSVS

The minimal case to show the topological behaviour of the 3D-RSVS requires 3 VOS cells. Figure 1.10 presents the geometries generated by varying the value of the central VOS cell. Between values of 0.3 and 0.1 the topology of the geometry changes from a single body to 2 spherical bodies. The case generating two spheres is added to Table 1.1 and shows a similar geometric convergence on spheres as the cases discussed in the previous Section (1.2.2).

Figure 1.11 shows 4 different surfaces generated by the 3 dimensional RSVS. On the left the volume grid on which the volume fractions are specified (thick lines) and the snaking grid (thin lines) on which the restricted surface evolves. These surfaces illustrate some of the more complex topologies that can be achieved with a small set of design variables. While these topologies may not be of interest for external aerodynamic

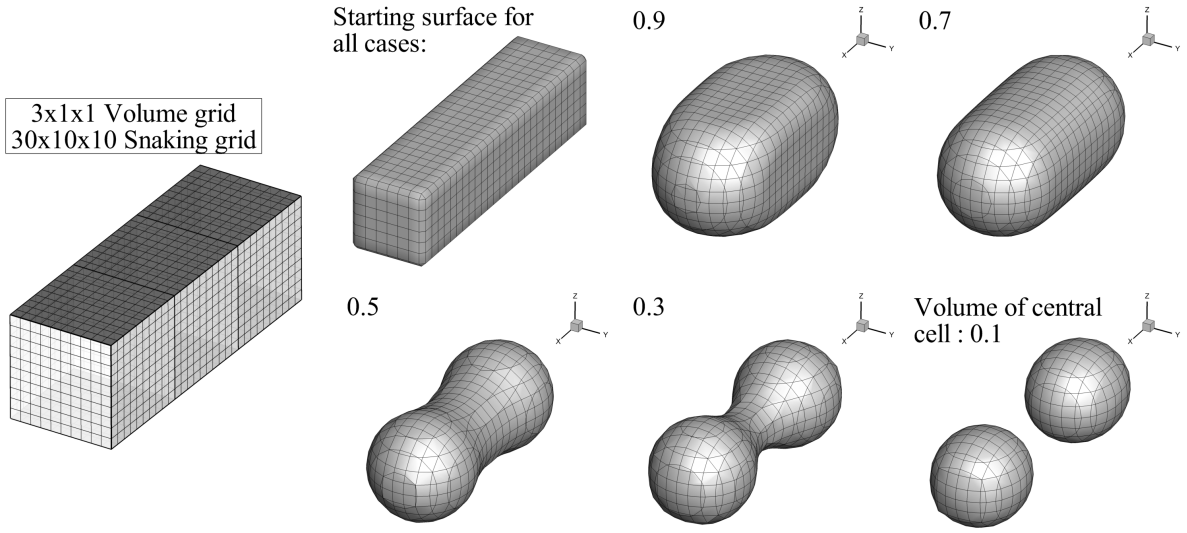


Figure 1.10: 5 different final geometries defined by 3 volume cells. The VOS at each end are kept constant while the volume fraction of the central cell is varied from 0.1 to 0.9.

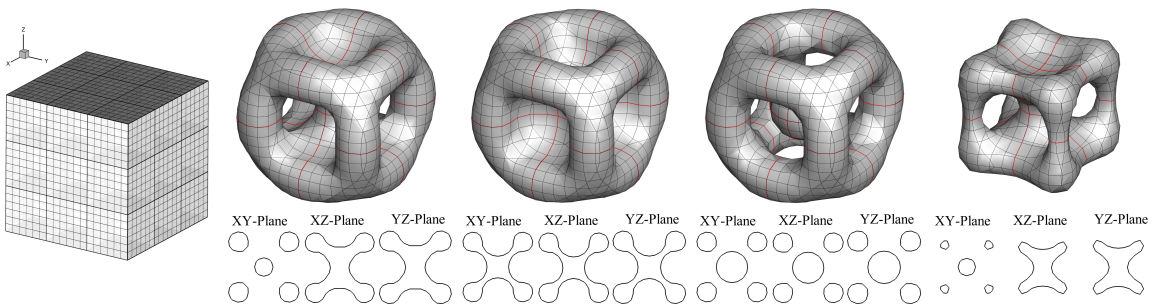


Figure 1.11: Four different final topologies defined by 27 VOS cells in a 3^3 layout (on the left). These cases illustrate the topological and smooth shape control 3D-RSVS provide with few design variables. To aid understanding of the topologies being represented slices through the centre of each dimension are provided.

optimisation, these could have application in the design of pipes or structures.

1.3 Implementation of the 3D-RSVS

This section delves into some of the detail of the 3D implementation motivating choices and exploring properties of the 3D-RSVS system. In particular the novel r-surface method is described and its implementation is outlined; finally the integration with the area minimisation defining the 3D-RSVS is presented. The implementation of the 3D-RSVS is done in C++ and is made available on GitHub [27] for contribution and download under the GNU Lesser-GPL license.

1.3.1 Rules for the Evolution of Water-Tight Surfaces

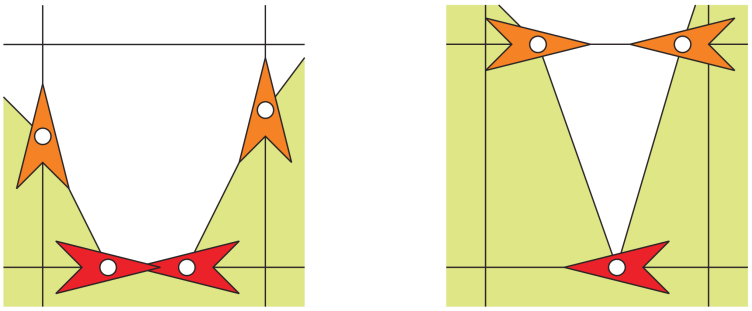
At the core of a three dimensional shape and topology parameterisation method must be an efficient method for topology evolution, compatible with the smooth and compact support requirements of aerodynamic parameterisation. The explicit evaluation of intersections between discrete or analytical geometries is a difficult and expensive problem. The r-surface simplifies those calculations by constraining control vertices to a grid, forcing intersections to happen point-to-point along edges. This property is very desirable in three dimensions as it reduces the cost of computing intersections to a search through a hashed map and, if necessary, a floating point comparison.

The extension of the restricted snake to surface objects relies on reformulating the two dimensional connectivity rules into a generalisable form. Recall that these rules, as described in [3], are:

1. No two connected snaxels can be on the same edge;
2. Snaxels must travel out of the profile.

In terms of connectivity the second rule manifests itself as: a snaxel cannot be connected to two edges which are part of the same face of the snaking grid. Examples of invalid connections are shown in Figure 1.12.

The initial connectivity rules can be formalised in terms of the relationship between r-snake elements (vertices and edges) and the underlying snaking grid elements (vertices, edges and faces). This observation enables the systematic extension of the connectivity rules to three dimensions. If two connected r-snake edges cannot be part of the same face of the snaking grid, it follows that two faces of a restricted surface cannot be part of the same volume cell of the underlying snaking grid.



(a) The snaxels in red are connected and on the same edge.

(b) The snaxel in red travels tangentially to the profile.

Figure 1.12: Invalid snaxel connections, image from [19].

The rules for this restricted surface (r-surface) are formalised into the following:

1. No two snaxels connected (by an r-surface edge) can be on the same *snaking grid* edge;
2. No two r-surface edges connected (by a snaxel) can be in the same *snaking grid* face;
3. No two r-surface faces connected (by r-surface edge) can be in the same *snaking grid* cell.

The 3-dimensional rules stated above can be further generalised to handle the marching of a N-dimensional restricted-polytope, including support for topology change. In all dimensions, there is a single special case for vertices which are 0-dimensional objects and all other rules are the same relative to the dimensionality of the object being handled. The N-dimensional rules are the following:

1. No two restricted-polytope 0-dimensional object connected (by a restricted-polytope 1-dimensional object) can be on the same *snaking grid* 1-dimensional object;
2. No two restricted-polytope z-dimensional object connected (by a restricted-polytope (z-1)-dimensional object) can be in the same *snaking grid* (z+1)-dimensional object for $z \in \{1, \cdots, N\}$.

The 3-dimensional rules were shown to work robustly and efficiently for arbitrarily complex geometries evolved on tetrahedral and hexahedral snaking-grids; the implementation is expected to work for all convex snaking grids (no internal angle above π). Algorithms, data structures and pseudo code for the current implementation of the r-surface are available in Section 1.3.2. Figures 1.13, 1.14 and 1.15 show the evolution of

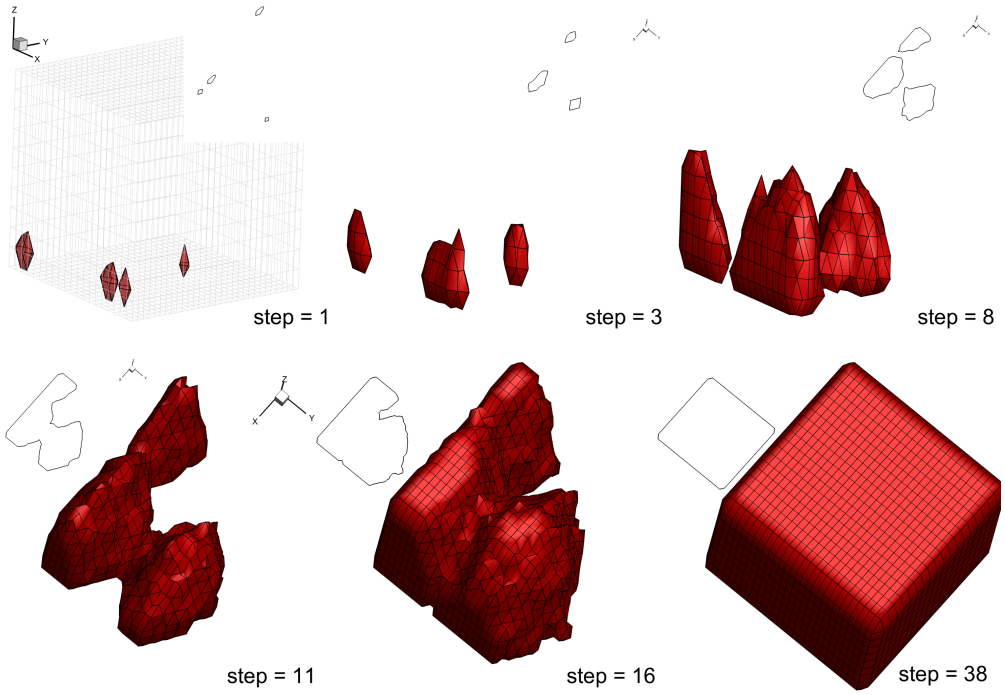


Figure 1.13: A restricted surface in a Cartesian snaking grid is evolved under a uniform velocity field. The surface is in red, the black outlines are a section through the surface at $z = 0.15$.

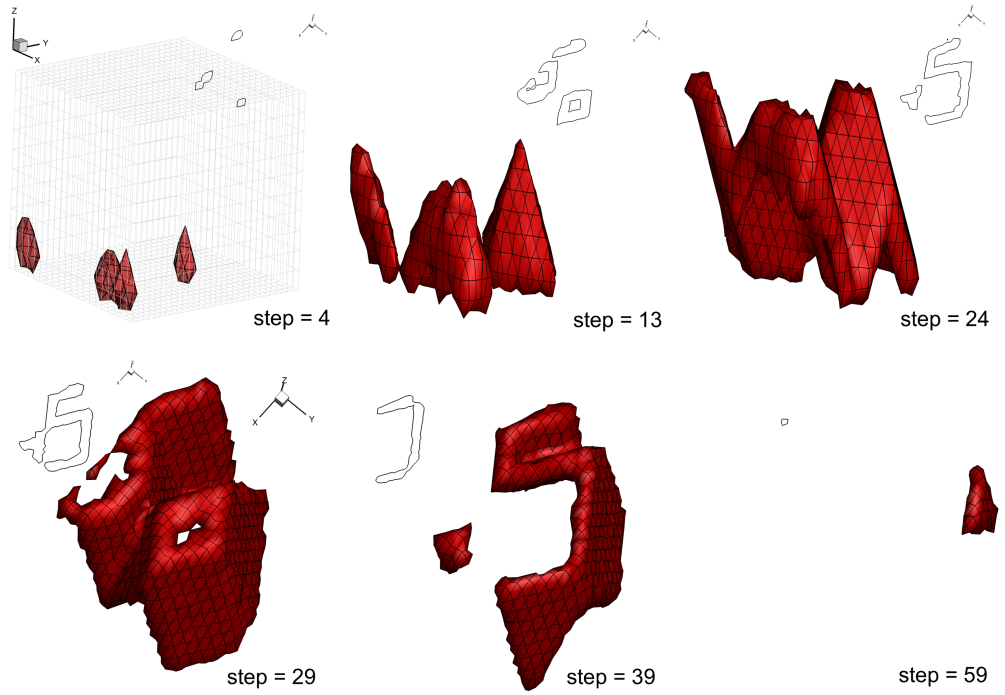


Figure 1.14: A restricted surface in a Cartesian snaking grid is evolved under a uniform velocity field with reflections at the design space boundary. The surface is in red, the black outlines are a section through the surface at $z = 0.15$.

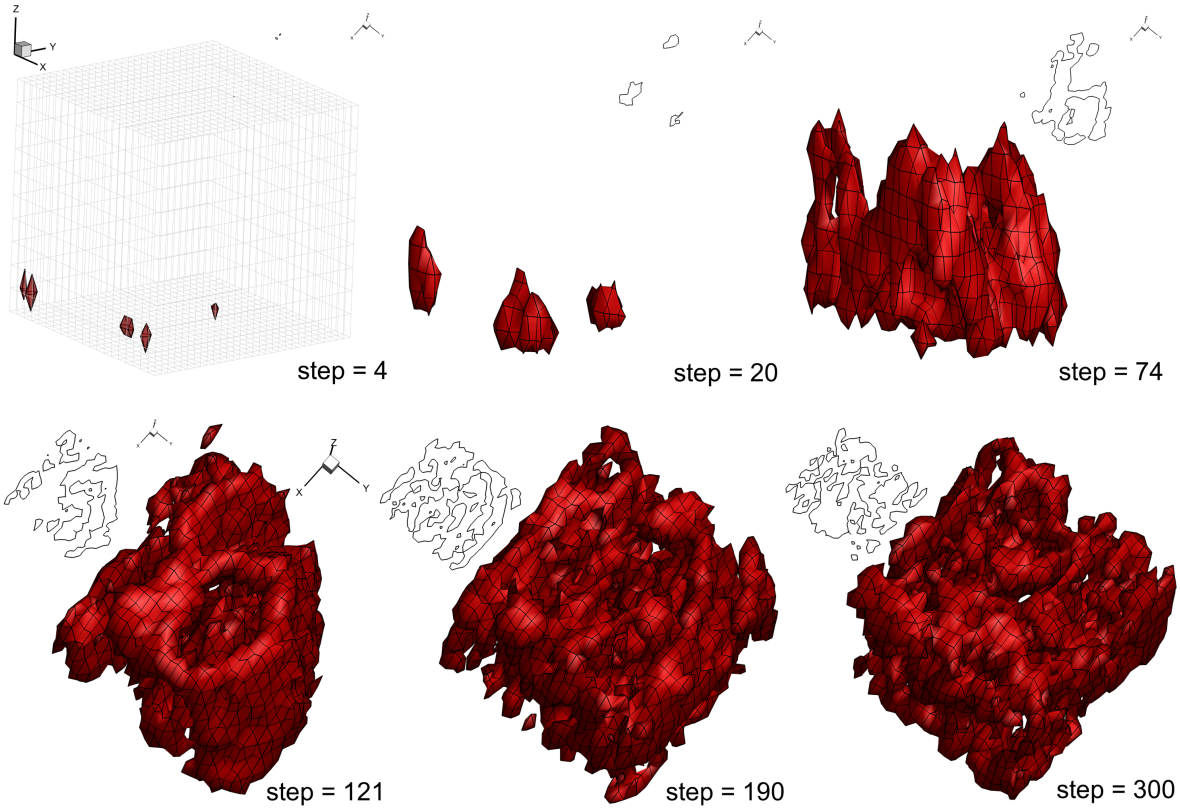


Figure 1.15: Test of the restricted surface process and rules: a restricted surface in a Cartesian snaking grid is evolved under a velocity field with random variations. The surface is in red, the black outlines are a section through the surface. The proposed restricted surface algorithm is robust and fast for arbitrary complex topologies.

r-surfaces spawned from 6 distinct vertices. Between these figures, only the algorithm for snaxel velocity update differs, respectively: unit velocity, unit velocity with reflection at the boundary and random velocity with reflection. This was performed as a test of the topological flexibility of the r-surface process, validating its use to evolve the 3D-RSVS surface.

The r-surface relies exclusively on connectivity information to detect collisions avoiding the need for expensive, floating point, intersection calculations. This process is efficient, robust and scalable: in Figure 1.15 the evolution features up to a thousand topological changes per step, processed in around a second per step.

1.3.2 Algorithms for the Implementation of the R-Surface

While the rules defining a valid r-surface have been stated in Section 1.3 they must now be implemented into a method for surface evolution. The restricted surface method relies on the data structures presented in Figure 1.16 and three core algorithms:

- **clean** the r-surface connectivity to have only valid elements (Algorithm 1);
- **cut and merge** geometries to evolve topology (Algorithm 3);
- **spawn** at snaking grid vertices (Algorithm 5).

The removal of invalid connections in the restricted surface relies on the combination of grid elements, and, the removal of orphaned elements. These two steps applied in the correct order for vertices, edges, faces and volume cells allows invalid connections to be pruned; Algorithm 5 presents the steps necessary to clean up those connections. In addition to these core processes two composited algorithms are used to complete the r-surface method:

- **snake initialisation** to generate the starting restricted surface (Algorithm 6);
- **grid vertex crossing** to enable the progression of the r-surface through the snaking grid (Algorithm 4).

Algorithm 1 Restricted surface algorithm for spawning at a snaking grid vertex.

To spawn at a vertex vx_i
for Each snaking grid edge e_j in vx_i`.edgeind` **do**
    Add a new snaxel on edge e_i
end for
for Each snaking grid face f_j in vx_i`.edgeind[].surfind` **do**
    Add a new edge in face f_j
    Connect this new edge to snaxels on edges f_j`.edgeind`
end for
for Each snaking grid cell c_j in ve_i`.edgeind[].surfind[].voluind` **do**
    Add a new face in volume element c_j
    Connect this new face to edges in faces c_j`.surface`
end for
Connect all faces with a new volume element.

`int`	Index			
`double`	VOS	Target VOS	VOS error	Volume
`vector<int>`	Surface indices (`surfind`)			

(a) Data structure of a restricted surface volume element ($d = 3$).

`int`	Index	Snaking grid volume index (`parentind`)		
`double` (for 2D)	VOS	Target VOS	VOS error	Area
`vector<int>`	Edge indices (`edgeind`)		Volume indices (`voluind[2]`)	

(b) Data structure of a restricted surface face element ($d = 2$).

`int`	Index	Snaking grid face index (`parentind`)	
`double`	Length		
`vector<int>`	Vertex indices (`vertind[2]`)	Surface indices (`surfind`)	

(c) Data structure of a restricted surface edge ($d = 1$).

`int`	Index	Snaking grid edge index (`parentind`)
	Origin vertex (`fromvert`)	Destination vertex (`tovert`)
	Frozen status (`isfreeze`)	
`double`	Normalised distance (`d`)	Normalised speed (`v`)
`vector<double>`	Length `l[3]`	
`vector<int>`	Edge indices (`edgeind`)	

(d) R-surface vertex (snaxel) element data structure ($d = 0$).

Vector of volume objects	Vector of face objects
Vector of edge objects	Vector of snaxel objects
`vector<bool>` are snaking grid vertices inside the r-surface?	

(e) R-surface data structure.

Figure 1.16: Definitions of the data structure used in the restricted surface process.

Algorithm 2 3D-RSVS element combination algorithm

if Elements (e_1 and e_2) of dimension n are identified for merging **then**
  Replace e_2 by e_1 in all connectivity lists.
  Mark e_2 for deletion.
end if

Algorithm 3 Restricted surface algorithm for cutting/merging of bodies.

for All snaxels s_i and s_j carried by the same grid edge **do**
  if s_i and s_j are in the same location **and** are not moving apart **then**
    Mark snaxels s_i and s_j for combination.
  end if
end for
Apply the snaxel combination. This creates invalid connections.
Run the connectivity clean-up of Algorithm 5.

Algorithm 4 Restricted surface algorithm for crossing through snaking grid vertices.

for All snaxels s_i **do**
  if s_i.d== 1 **and** s_i.v> 0 **then**
    Mark snaking grid vertex s_i.`tovert` for spawning.
  else if s_i.d== 0 **and** s_i.v< 0 **then**
    Mark snaking grid vertex s_i.`fromvert` for spawning.
  end if
end for
for Each snaking grid vertex vx_i marked for spawning because of snaxel(s) $s_{i1,\cdots,im}$. **do**
  Apply the spawning algorithm (Algorithm 1) at vertex vx_i
end for
Apply the algorithm for topology cutting/merging (Algorithm 3) to all the new snaxels.
Run the connectivity clean-up of Algorithm 5.

Algorithm 5 Restricted surface connectivity clean-up process (notation defined in Figure 1.16).

while Changes of connectivity are detected **do**
  for All snaxels s_i **do**
    if Snaxel s_i is disconnected from the r-surface **then**
      Mark snaxel s_i for deletion
    end if
  end for
  Remove snaxels marked for deletion.
  for All r-surface edge e_i **do**
    if Edge is connected to a single vertex: e_i.`vertind`[1] == e_i.`vertind`[2] **then**
      Mark edge e_i for deletion.
    else if e_i.`vertind`[1].`parentind`== e_i.`vertind`[2].`parentind` **then**
      Mark snaxels e_i.`vertind` 1 and 2 for combination.
      Mark edge e_i for deletion.
    end if
  end for
  Apply snaxel combination operations (Algorithm 2).
  Remove edges marked for deletion.
  for All r-surface edge e_i **do**
    if e_i.`vertind`[l] == e_j.`vertind`[m] **and** e_i.`parentind`== e_j.`parentind` **then**
      Mark edges e_i and e_j for combination.
    end if
  end for
  Apply snaxel combination operations.
  for All r-surface face f_i **do**
    if f_i.`edgeind`[l] == f_j.`edgeind`[m] **and** f_i.`parentind`== f_j.`parentind` **then**
      Mark faces f_i and f_j for combination.
    end if
  end for
  Apply face combination operations.
end while

Algorithm 6 Restricted surface algorithm for initialisation at the boundary of the void domain.

Note: identify vertices around which lie between region of empty VOS and non-empty VOS.
for All faces f_i in the snaking grid **and** the VOS grid **do**
  if f_i`.voluind[0].target==` 0 **and** f_i`.voluind[1].target>` 0 **then**
    Mark all vertices connected to this face for spawning f_i`.edgeind[].vertind`
  end if
end for
for Each snaking grid vertex vx_i marked for spawning because of snaxel(s) $s_{i1,\cdots,im}$. **do**
  Apply the spawning algorithm (Algorithm 1) at vertex vx_i
end for
Take a step of length $d = 0.5$ for all snaxels.
Run the topology cutting/merging Algorithm 3.
Note: At this stage two surfaces exist for each block of faces: one just outside the boundary and one just inside.
remove the outside surface, identifying it using a flood fill on vertices lying on edges which are in cells with a VOS target of 0.
for Each face f_i of the r-surface **do**
  if f_i`.parentind.target(VOS)==` 0 **then**
    Mark f_i for removal.
    Use flood fill to mark all faces, edges and vertices of the r-surface connected to f_i for deletion.
  end if
end for
Process requested element removals.
Reverse surface direction by flipping snaxels (`s.fromvert`↔`s.tovert` and `s.d=` 1−`s.d`)
Identify snaking grid vertices inside the r-surface using flooding from snaxels origin vertex.
Build r-surface volumes by identifying blocks of internal grid vertices and connected surfaces.

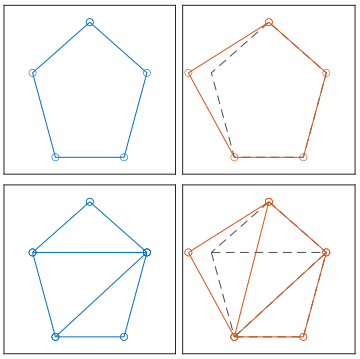
1.3.3 Triangulation of Restricted Surfaces into Polyhedra

In order to compute the area and volume of a geometry and drive the 3D-RSVS process an expression for the shape of those faces is required. While the rules for building the r-surface guarantee the formation of water-tight surfaces, it does not guarantee that the surface will be a polyhedron: some of the faces may not be flat. This is because the r-surface is controlled by the positioning of its vertices with the rest of the geometry derived from the connectivity information forced by the snaking grid. Since an analytical solution to the 3D-RSVS formulation has not been found, another approach to define the position of the boundary is needed.

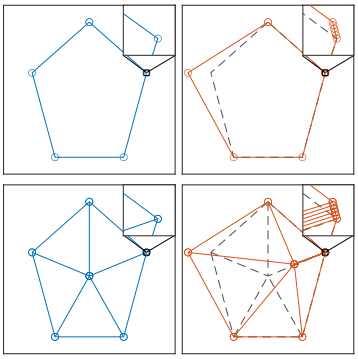
The most natural approach to arrive at an explicit boundary is to triangulate the restricted surface. A triangulation guarantees flat faces, meaning that the output of the process is always a valid polyhedron; in turn this allows reliable calculation of the volume and area enclosed by the surface. By triangulating all faces with more than 3 edges, regardless of whether they were flat or not, simplifies the implementation of the volume and area calculations and differentiations as only triangular faces will need to be processed. In order to be effective in the r-surface context the triangulation needs to guarantee a smooth response of the area and the volume of the polyhedron; the following observations are made:

1. the triangulation connectivity must not change when the snaxels move, or in other words, triangulation connectivity must depend only on r-surface connectivity, this rule is breached by Figure 1.17a;
2. the triangulation properties (area, volume) must only depend on the geometric properties of the surface, not on its connectivity, this rule is breached by Figure 1.17b.

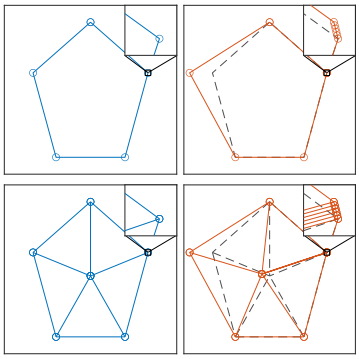
In light of these observations, three triangulations are considered in Figure 1.17: a Delaunay triangulation (Figure 1.17a), a triangulation built by linking every vertex of the polygon to the average point of those vertices (Figure 1.17b); and, one built by linking every vertex to the centroid of the curve defining the polygon (Figure 1.17c). The triangulation relying on the mean position of the vertices fails when a r-surface crosses through a snaking grid vertex: at that point, multiple vertices will lie very close to each other, unduly impacting the triangulation. To achieve consistent connectivity and the smooth response through changes of r-surface connectivity the triangulation of faces is built around point $\bar{\mathbf{c}}$ which is the mean position of face vertices weighted by edge length. Equation 1.3 formalises this process for a closed face with $n+1$ vertices and the last vertex repeated.



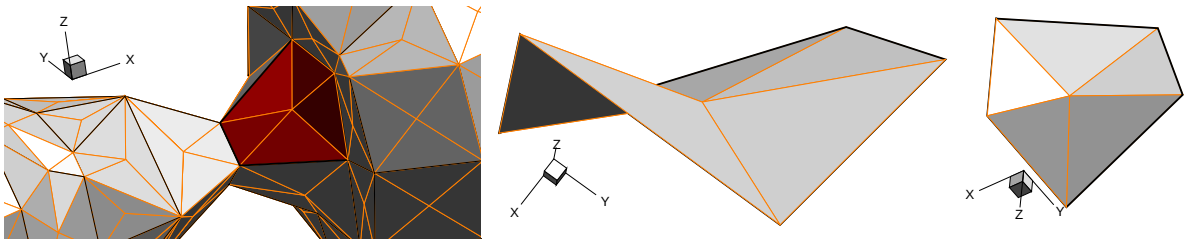
(a) Delaunay triangulation



(b) Triangulation through the mean of the control points



(c) Triangulation through the centroid of the curve (Equation 1.3)



(d) Triangulation through the centroid for a non-flat face.

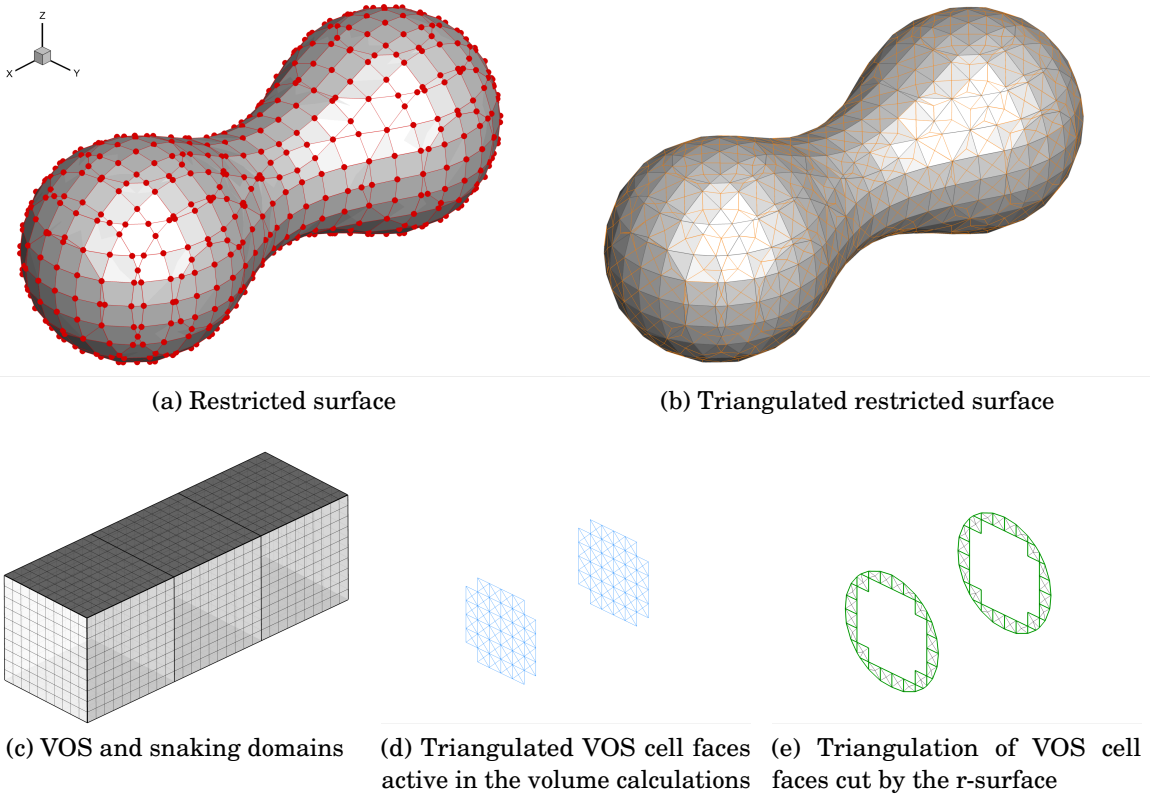
Figure 1.17: Triangulation of a pentagon and a pentagon with one moved vertex, and one repeated vertex with three possible triangulations. Only the contour length weighted centroid (Figure 1.17c), fulfils the stability requirements of the 3D-RSVS parameterisation.

$$\bar{\mathbf{c}} = \frac{\sum_{i=1}^{n} \|\mathbf{p}_{i+1} - \mathbf{p}_i\| (\mathbf{p}_{i+1} + \mathbf{p}_i)}{2 \sum_{i=1}^{n} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|} \qquad \text{with} : \mathbf{p}_{n+1} = \mathbf{p}_1 \tag{1.3}$$

The r-surface face is triangulated by linking this pseudo-centroid of the face to each of the vertices. This formulation is used as it prevents changes in connectivity, due to movements of the restricted surface, to cause jumps in the position of $\bar{\mathbf{c}}$; it is not affected by duplicate points. Figure 1.17 shows the stability of the centroid calculation with changes of connectivity, allowing smooth evolution of the surface necessary for the r-surface method to converge on a solution of the 3D-RSVS governing equation.

1.3.4 Evaluation of the 3D-RSVS Equations on the R-Surface

To drive the position of the restricted surface the original continuous area minimisation problem (Equation 1.1) needs to be discretised in terms of the r-surface snaxel variables.



(a) Restricted surface

(b) Triangulated restricted surface

(c) VOS and snaking domains

(d) Triangulated VOS cell faces active in the volume calculations

(e) Triangulation of VOS cell faces cut by the r-surface

Figure 1.18: Triangulations necessary for the calculation of the objective function and constraints of the 3D-RSVS problem.

This process is achieved by expressing the area and volume integrals on the triangulated surface; a discrete form of the governing mathematical program can then be defined.

The discrete variables used for the 3D-RSVS are very similar to the two dimensional implementation: the same six properties from the r-surface geometry and the snaxel positions are needed. These properties are:

- the snaxel index (i), used to reference it in all operations;
- the normalised position along an edge ($d_i \in [0,1]$);
- the scalar velocity along that edge ($v_i \in \mathbb{R}$);
- the snaxel position in Cartesian coordinates ($\mathbf{p}_i$);
- the direction of travel of the snaxel ($\Delta\mathbf{g}_i$) and the vertex of origin ($\mathbf{g_{i,1}}$);
- the normal vectors to the preceding and following edges ($\mathbf{n}_i$ and $\mathbf{n}_{i+1}$).

These properties were represented graphically in Figure 1.5.

The continuous expressions presented in Section 1.1.1 are easily computed for polyhedra with triangular faces. The following notation is adopted: S refers to the r-surface;

C to cells of the VOS grid. The area of each triangular face (A_{S_k}) is computed using Equation 1.4, only the position of each of the corner vertices is required ($\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}$). This approach is also applicable to $V_{S,k}$, the contribution of face k, to the volume of the polyhedron. The volume contributions from the underlying grid ($V_{C_j,k}$) are also taken into account. Vertices represented by symbol $\mathbf{p}_{i,k}$ are *active* vertices (snaxels or pseudo-centroids) which move with the surface being designed, *static* vertices which are part of underlying grids are represented by symbol $\mathbf{g}_{i,k}$.

$$A_{S,k} = \frac{1}{2} \left\| \left(\mathbf{p}_{1,k} - \mathbf{p}_{0,k}\right) \times \left(\mathbf{p}_{2,k} - \mathbf{p}_{0,k}\right) \right\| \tag{1.4}$$

$$V_{S,k} = \frac{1}{6} \left(\mathbf{p}_{0,k} \cdot \left(\left(\mathbf{p}_{1,k} - \mathbf{p}_{0,k}\right) \times \left(\mathbf{p}_{2,k} - \mathbf{p}_{0,k}\right) \right) \right) \tag{1.5}$$

$$V_{C_j,k} = \frac{1}{6} \left(\mathbf{g}_{0,k} \cdot \left(\left(\mathbf{g}_{1,k} - \mathbf{g}_{0,k}\right) \times \left(\mathbf{g}_{2,k} - \mathbf{g}_{0,k}\right) \right) \right) \tag{1.6}$$

The equations presented above can be assembled to calculate the volume of the polyhedron formed by the intersection of the r-surface and the faces of the *design grid*, this quantity is represented by value $V_{S\cap C,j}$. Components of this polyhedron are represented in Figure 1.18.

$$V_{S\cap C,j} = \frac{1}{6} \sum_{k=j_S(1)}^{j_S(q_{jS})} \mathbf{p}_{0,k} \cdot \left(\left(\mathbf{p}_{1,k} - \mathbf{p}_{0,k}\right) \times \left(\mathbf{p}_{2,k} - \mathbf{p}_{0,k}\right) \right) + \frac{1}{6} \sum_{k=j_C(1)}^{j_C(q_{jC})} \mathbf{g}_{0,k} \cdot \left(\left(\mathbf{g}_{1,k} - \mathbf{g}_{0,k}\right) \times \left(\mathbf{g}_{2,k} - \mathbf{g}_{0,k}\right) \right) \tag{1.7}$$

$j_S(\{1, \cdots, q_{jS}\})$ and $j_C(\{1, \cdots, q_{jC}\})$ are indexing functions specified for each design cell selecting the correct vertices respectively from the triangulated r-surface and the volume grid. The equations for volume and area of the polyhedra formed by the intersection of the r-surface and the design grid are derived and are now substituted into the mathematical program which defines the RSVS problem in 3-dimensions (Equation 1.1). This process leads to Equation 1.2 (repeated below) for the 3D-RSVS discretised by a triangulated r-surfaces.

$$\min_{\mathbf{d}} \sum_{k=1}^{q} A_{S,k}\left(\mathbf{p}_{0,k}, \mathbf{p}_{1,k}, \mathbf{p}_{2,k}\right) \quad \text{with } \mathbf{p}_{i,k}(\mathbf{d})$$

$$\text{s.t.} \sum_{k=j_S(1)}^{j_S(q_{jS})} V_{S,k}\left(\mathbf{p}_{0,k}, \mathbf{p}_{1,k}, \mathbf{p}_{2,k}\right) + \sum_{k=j_C(1)}^{j_C(q_{jC})} V_{C,k}\left(\mathbf{g}_{0,k}, \mathbf{g}_{1,k}, \mathbf{g}_{2,k}\right) = V_j \quad \forall j \in \{1, \cdots, m\}$$

Building a 3D-RSVS surface consists in finding the positions $\mathbf{d}$ of the r-surface snaxels which solve the mathematical program of Equation 1.2. This formulation has the benefit of being very general, it can be tackled on an arbitrary volume grid with any underlying snaking grid with any optimisation method.

1.3.5 Restricted-Surface Marching

As was the case for the 2D-RSVS the 3D implementation relies on SQP to solve the minimisation problem defining the 3D-RSVS surfaces. SQP provides convergence in a limited number of iterations for problems for which derivatives are available, which is the case for the discrete form of the 3D-RSVS problem presented in Equation 1.2.

The Newton step SQP equations presented below are derived in Boggs and Tolle [5]; only Equation 1.8 for the update of the snaxel velocities is shown here. The evaluation of the SQP (Equation 1.8) requires the following derivatives to be calculated: the Jacobian of the constraints ($\nabla_{\mathbf{d}}\mathbf{h}$); the gradient of the objective ($\nabla_{\mathbf{d}} f$) and the Hessian of the objective ($\mathbf{H}_{\mathbf{d}} f$).

$$\begin{aligned}
\boldsymbol{\lambda}^{k+1} &= \left((\nabla_{\mathbf{d}}\mathbf{h})^T (\mathbf{H}_{\mathbf{d}} f)^{-1} (\nabla_{\mathbf{d}} f)\right)^{-1} \left(\mathbf{h} - (\nabla_{\mathbf{d}}\mathbf{h})^T (\mathbf{H}_{\mathbf{d}} f)^{-1} (\nabla_{\mathbf{d}} f)\right) \\
\Delta_{\mathbf{d}}^{k+1} &= \mathbf{d}^{k+1} - \mathbf{d}^k = -(\mathbf{H}_{\mathbf{d}} f)^{-1} \left((\nabla_{\mathbf{d}} f) + (\nabla_{\mathbf{d}}\mathbf{h})\,\boldsymbol{\lambda}^{k+1}\right) \\
&\text{with}: f = A(\mathbf{d}) \quad \text{and } \mathbf{h} = \mathbf{V}(\mathbf{d})
\end{aligned} \tag{1.8}$$

Differentiation of f and $\mathbf{h}$ was done using the MATLAB Symbolic Toolbox, which allows analytical differentiation of the terms and the generation of the corresponding C code. This process greatly reduces the difficulty of differentiating the equations presented in Equations 1.4 and 1.7 and the risk of a mistake in the implementation. Because the differentiation is algorithmically developed from the area, volume and centroid functions; validation of the functions also validates the implementation of the derivatives. The C++ implementation also allows both analytical and finite difference gradients to be used; comparison showed them to be consistent.

This algorithmic process was applied to the differentiation of:

- $A_{S,k}$ with regard to $\mathbf{p}_{\mathbf{i},\mathbf{k}}$;
- $V_{C\cap S,k}$ with regard to $\mathbf{p}_{\mathbf{i},\mathbf{k}}$.
- $\bar{\mathbf{c}}$ with regard to $\mathbf{p}_{\mathbf{i},\mathbf{k}}$.

These differentiations are then expressed in terms of $\mathbf{d}$ using the matrix equation chain rule.

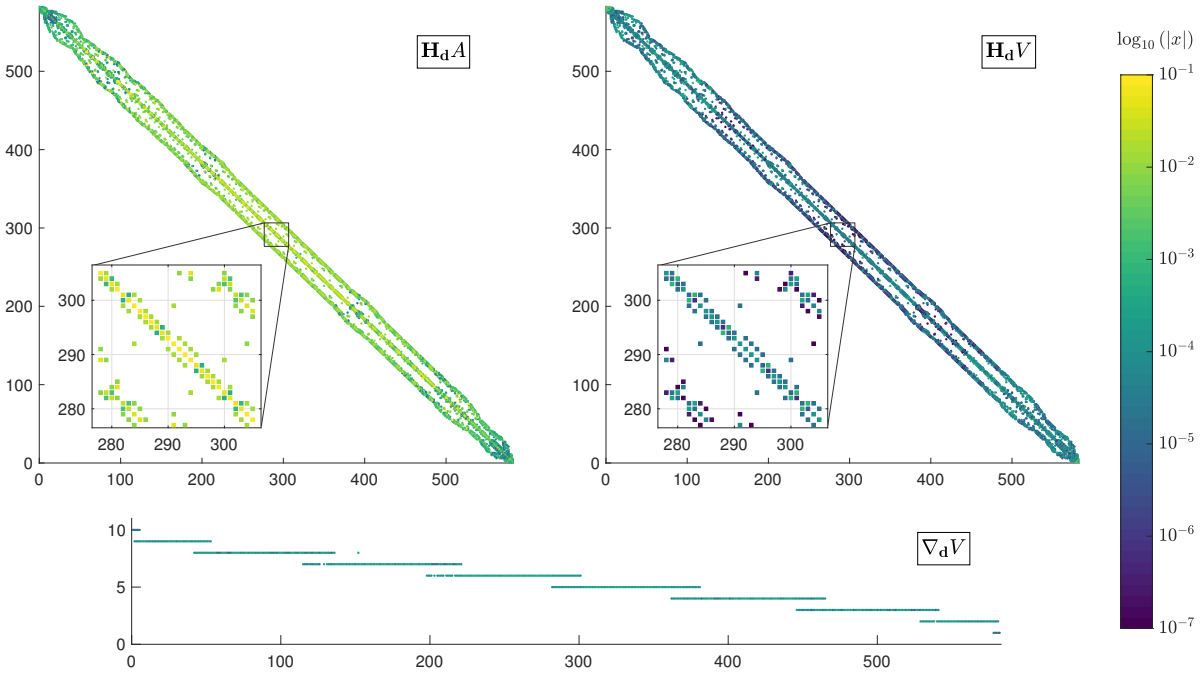


Figure 1.19: Sparsity patterns of the Hessian of the objective function and the Hessian and Jacobian of the constraints. These values are for the geometry of Figure 1.7.

While most properties of the RSVS carry over from the two to the three dimensional implementation, one notable difference is the expected size of the system. Whereas most 2D geometries can be accurately represented with a few hundreds to a few thousand control points, representation of complex topologies in 3D will usually take thousands to tens of thousands of snaxels. With large systems solving the SQP rapidly becomes expensive and methods are required to reduce the cost of the operations. For very large systems it may be necessary to switch to optimisation methods better suited to very large numbers of design variables, like Sequential Linear Quadratic Programming (SLQP), Sequential Linear Programming (SLP), or interior point methods.

The system that must be solved to compute the 3D-RSVS is sparse; this property can be exploited to speed up calculation for systems of intermediate sizes (thousands and low tens of thousands of points). Figure 1.19 shows the Hessian and Jacobian of the constraint and of the objective, for the profile in Figure 1.7. Sparsity is critical in reducing the memory footprint of the SQP calculation, letting it scale linearly instead of quadratically with the number of snaxels. The current implementation exploits both dense [11] and sparse [13] solvers using the Eigen library [16] for matrix mathematics in C++.

1.4 Properties of the 3D-RSVS Design Space

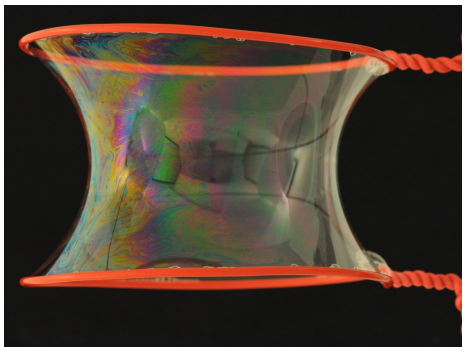
In this section, a comparison to minimal surfaces and an equivalence with surfaces of mean curvature are established. In addition, the wide range of possible design spaces enabled by the formulation of the 3D-RSVS is explored.

1.4.1 Discrete Differential Geometry as a Foundation of the 3D-RSVS

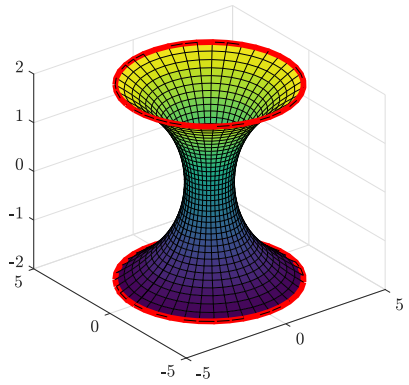
The study of minimal surfaces concerns itself with the discovery of surfaces of minimal area for a set of boundary conditions; these surfaces have a physical analogy: (open) soap films on wire frames. Provided that the air pressure is the same on both faces of the soap film, these naturally tend to minimise area. To develop solutions for a given boundary condition, the minimal surface problem can be formulated in a number of ways, one of which is calculus of variations. The first minimal surface to be described explicitly was the catenoid by Euler in 1741 (Figure 1.20) followed by the helicoid by Meusnier in 1776 (Figure 1.21) [28]. While steady progress had been made in the intervening time, the advent of computers in the 1980s along with the development of advanced mathematical machinery has allowed an explosion in the number and scope of discoveries in the field [28].

The relationship between the 3D-RSVS and minimal surfaces is obvious: both aim to minimise the area functional; however the 3D-RSVS differs in two ways: it is a free standing closed body, and it has a “pressure differential” between its faces in the form of the area constraint. In fact, minimal surfaces are part of a broader family of bodies knowns as surfaces of constant mean curvature (CMC); where minimal surfaces are a special case of CMCs for which the mean curvature is 0. The mean curvature is simply the sum of the principal curvatures. Continuing the “soap-film” analogy, surfaces of constant mean curvature are the result of minimising the area of a boundary between domains at different pressures: where minimal surfaces are films, CMCs allow bubbles. In general, the definition of a CMC surface is made in terms of a constrained volume [26, 35, 37], making the 3D-RSVS an obvious relative of these surfaces. Like minimal surfaces, a number of analytically described constant mean curvature surfaces exist: spheres, Wente tori [36], and the periodic P-CMC family [35]; however general explicit solutions for arbitrary boundary conditions do not exist.

Minimal surfaces and CMCs are found in nature as they allow the minimisation of internal stresses (films, bubbles and polymer mixtures), and optimise the assignment of



(a) Soap film forming a catenoid from soapbubble.dk [33].



(b) Catenoid (boundary in red)

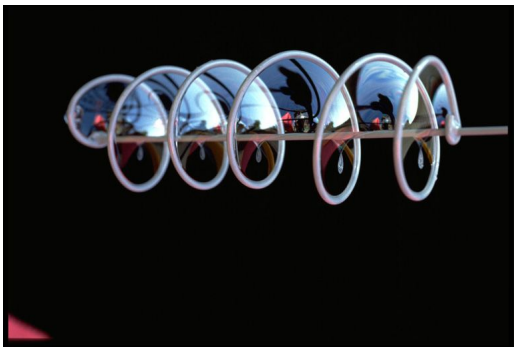
$$
\begin{aligned}
x &= c\cosh\left(\frac{v}{c}\right)\cos u \\
y &= c\cosh\left(\frac{v}{c}\right)\sin u \\
z &= v
\end{aligned}
\qquad
\begin{aligned}
&\text{where } u \in [-\pi, \pi) \\
&\text{and } v \in \mathbb{R} \\
&\text{and } c \text{ a constant}
\end{aligned}
$$

(c) Catenoid equation

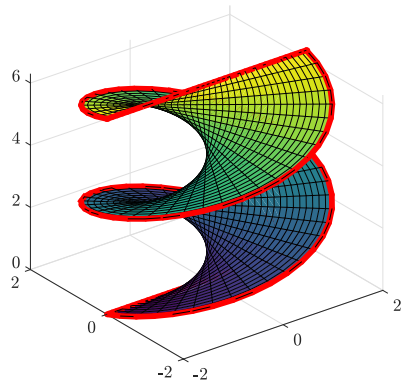
Figure 1.20: Soap films, equivalent minimal surfaces and explicit definitions for the catenoid.

resources [28] (leaves of holly). These properties have motivated their use in architectural design and polymer micro structures. While analytical results for the 3D-RSVS may be out of reach at the moment, the progress in discrete differential geometry offers means of developing and classifying the results of the parameterisation. Of interest to the development of the 3D-RSVS are the numerical solvers for CMCs; notably the "Surface Evolver" by Brakke [6] has had a large impact on the field. This tool allows the evolution of a large number of surfaces, under a variety volume, boundary (wire-frames) and vertex constraints; with constant mesh topology. Later developments of these methods have led to mesh optimizing versions of the Surface Evolver which allowed topological cuts of the mesh, upon the collapse of faces [26].

From the previous discussion it appears that the 3D-RSVS has a very similar definition to CMC surfaces, with the difference that the RSVS resolves multiple, local, volume constraints rather than a single global one for CMCs. The geometry parameterised by the 3D-RSVS is in fact the solution to a set of simultaneous CMC problems, with the boundary conditions separating those CMC patches also unknown. Based on the



(a) Helicoidal soap bubble, from Exploratorium Teacher Institute [17]



(b) Helicoid (boundary in red)

$$\begin{aligned} x &= u\cos(cv) \\ y &= u\sin(cv) \\ z &= v \end{aligned} \qquad \text{where } u,\ v \in \mathbb{R} \text{ and } c \text{ a constant}$$

(c) Helicoid equation

Figure 1.21: Soap films, equivalent minimal surfaces and explicit definitions for the helicoid.

similar formulations between the 3D-RSVS, and CMCs and the derivation of the two dimensional RSVS equivalence to non-uniform rational B-Splines (NURBS); it seems a reasonable hypothesis, that the continuous limit curve of the 3D-RSVS process are G_1 continuous CMCs patches.

This hypothesis is supported by Table 1.2 which showed the tendency of the 3D-RSVS to produce spheres for simple layouts of design variables. While an empirical or analytical proof of this hypothesis is not provided in this thesis, a path to one is suggested.

1. Solve the 3D-RSVS problem for a set of VOS design variables;
2. for each VOS cells extract the boundaries generated by the intersection with the r-surface;
3. for each boundary generate a CMC patch using “Surface Evolver” [6] or the PVT-CMC of Pan et al. [26];
4. compare the surfaces to the RSVS, if the hypothesis is true, the solutions should converge as the discretisations converge.

It was not followed as this potential relationship was only discovered late in the redaction

of this document.

The similarity between the 3D-RSVS problem and CMC surfaces has broader implications for the development of the implementation. The CMC and RSVS systems have similar properties; this suggests that methods used in CMC solvers to guarantee convergence [7] and smooth meshes [26] may be used to alleviate the current limitations of the 3D-RSVS. Another implementation of CMC surfaces is available as a plug-in [29] to the computational aided design (CAD) package *Rhinoceros 3D* [23]. These tools for the calculation of discrete CMCs are all interactive, meaning that the current cost of the 3D-RSVS is not intrinsic to the method.

1.4.2 Tailoring of Volume of Solid Grids

The integration of topology optimisation tools and CMC surface engines into CAD distributions reveals that the scope of the 3D-RSVS may be broader than shape and topology parameterisation for aerodynamics. In fact, the 3D-RSVS is a generic design tool with close links to structural topology optimisation (STO) density methods, level set methods, NURBS, and, CMC surfaces. These relationships suggest that the RSVS could be used interactively as a prototyping tool which natively supports compact and smooth topology optimisation. While the need for compactness in many optimisation applications has been reduced by the ubiquity of adjoint solvers, it is necessary that design modes be intuitive to designers.

While Cartesian VOS layouts are an effective test of the initial implementation of the parameterisation, these grids do not exploit the somewhat unique flexibility of the 3D-RSVS to generate smooth shapes out of arbitrary design spaces. The use of non-Cartesian volume spaces, has two benefits: the number of unused design variables for a given resolution can be significantly reduced, and, modal responses can be tailored to a geometry. The following four grid modifications are envisaged:

1. **global grid deformations:** rotations, shears and stretches applied to the grid to replicate usual engineering design modes (e.g. in aerodynamics span, sweep, thickness);
2. **local grid deformations:** would likely behave in a similar way to an FFD control cage on the design;
3. **voxel based refinement:** extension of the 2D RSVS refinement process;
4. **Voronoi design cells:** user defined points form a design space by using their Voronoi diagram as the VOS cells of the 3D-RSVS, this process allows extreme flexibility and relatively easy user interface.

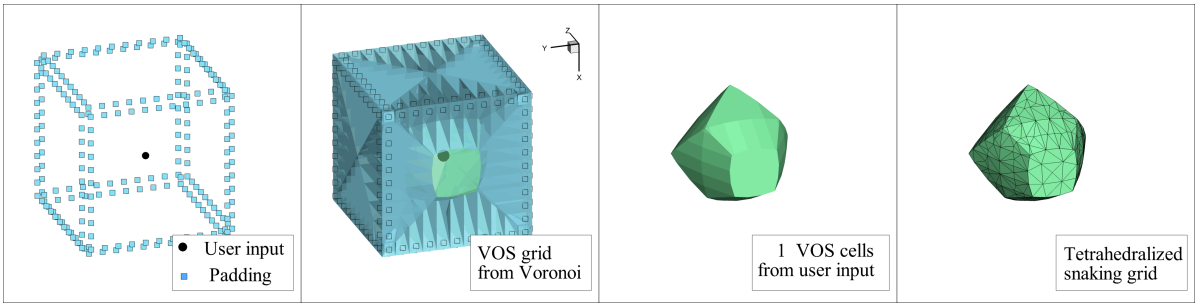
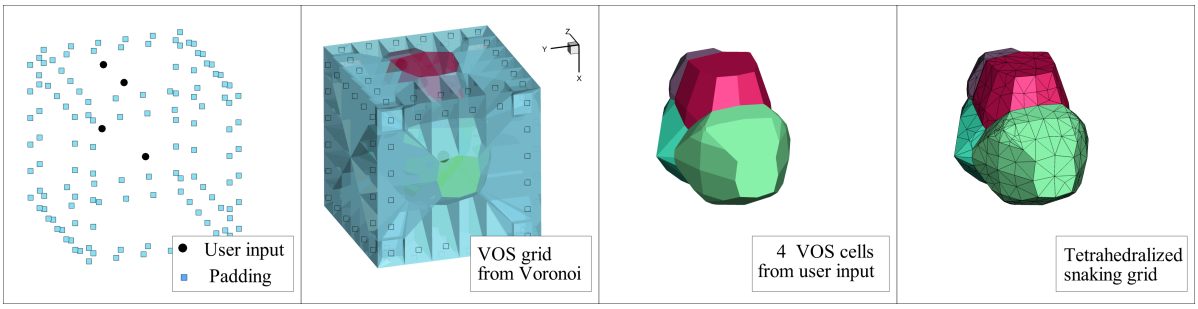
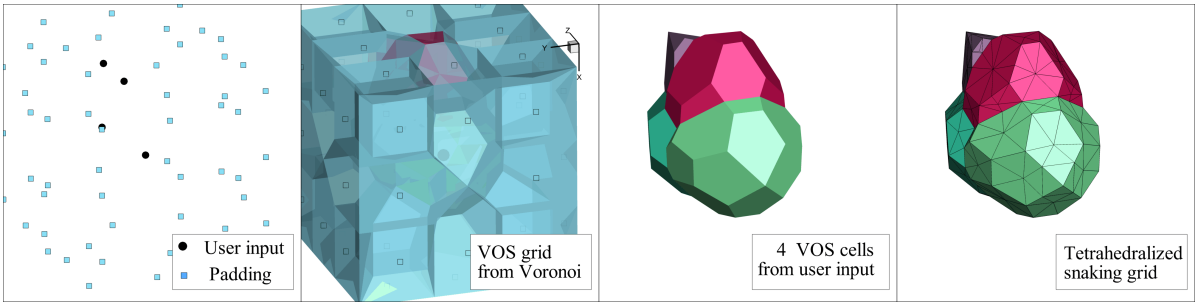


Figure 1.22: VOS mesh generation from a single input point and a padding distance of 0.05.



(a) 4 inputs and padding distance of 0.1.



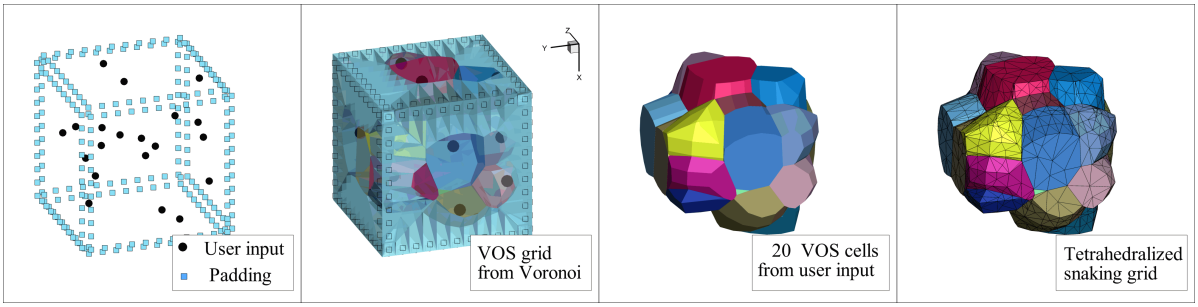
(b) 4 inputs and padding distance of 0.3.

Figure 1.23: Generation of 4 Voronoi cells for various padding distances, the padding distances ensures the cells are closed and the edge of the VOS mesh is convex.

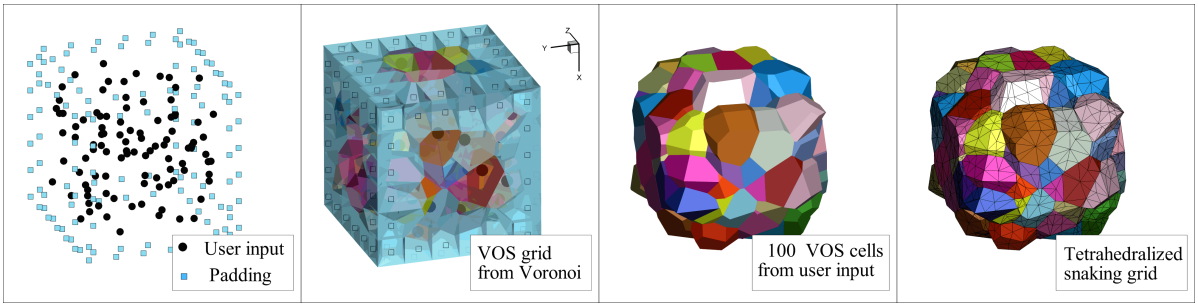
Of those four only the global grid deformations and the Voronoi design cells have been implemented. Examples of the global deformations can be seen in Section 1.2, notably in Figures 1.7 and 1.8 where the x-wise dimension is stretched to achieve an elongated body; and in Figure 1.9 where the span is stretched to achieve a wing-like aspect ratio.

The generation of Voronoi VOS meshes is done in 5 steps which are:

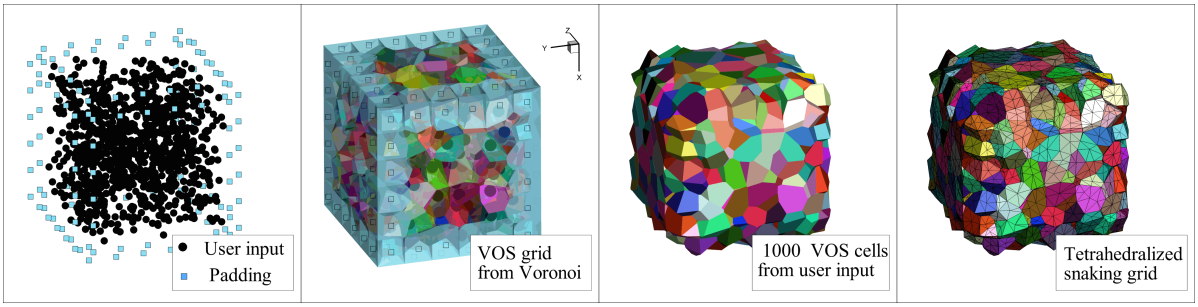
1. **load user input points** the mesh will have a 1 to 1 mapping of point to VOS cells, with the cell occupying the region surrounding its corresponding point;
2. **add padding points** these make sure the VOS mesh can be closed and convex at



(a) 20 inputs and padding distance of 0.05.



(b) 100 inputs and padding distance of 0.1.



(c) 1000 inputs and padding distance of 0.1.

Figure 1.24: Generation of Voronoi design spaces with 20, 100 and 1000 design variables.

the edge of the domain;

3. **generate Voronoi diagram using TetGen [31]** which is used as the VOS mesh;
4. **generate a tetrahedralisation** as the snaking grid on which the r-surface will evolve.

The steps are represented in Figure 1.22 for a single input point (and therefore a single active VOS cell), steps 1 and 2 are combined inside the first image on the left, steps then go from left to right. The impact of changing the padding distance is shown in Figure 1.23 for 4 input points. Examples of Voronoi grids generated with more input points are shown in Figure 1.24.

1.5 Integration into Optimisation Frameworks

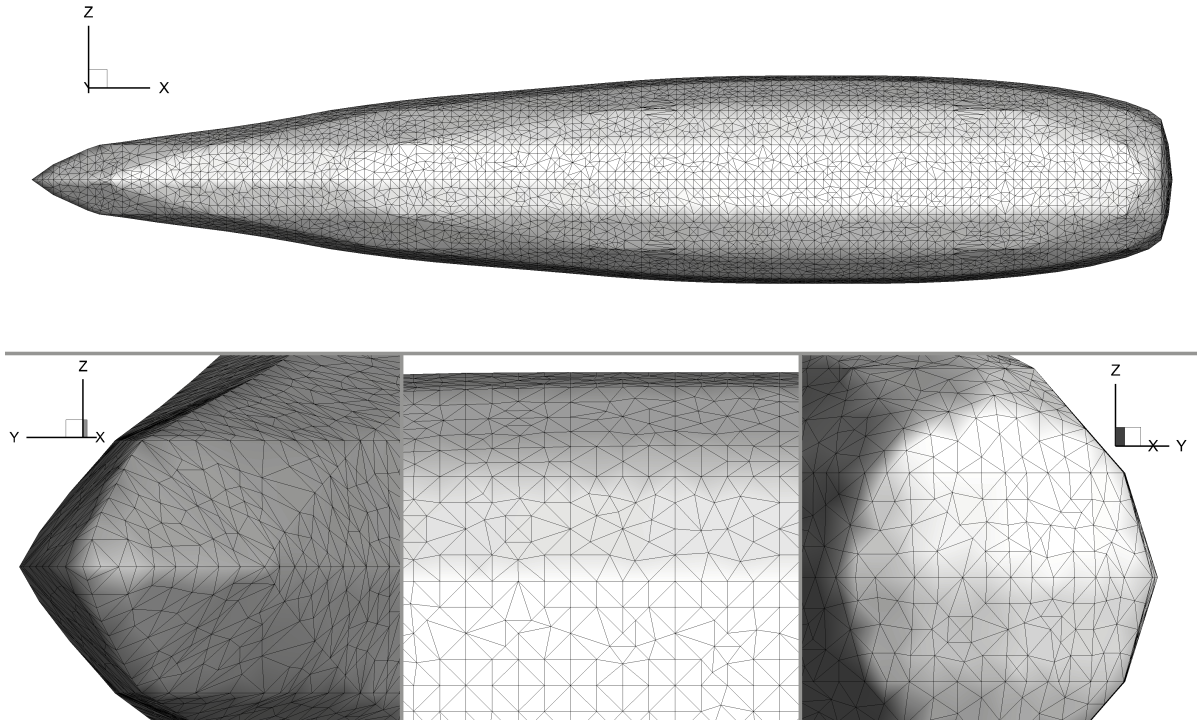
In order to use the 3D-RSVS for aerodynamic optimisation it must be integrated into a framework which supports its topological flexibility. The main bottle neck with current tools are is surface and volume meshing for arbitrary topology; automatic tools are not widely available and traditional finite volume flow solvers are sensitive to mesh quality. Nevertheless, the 3D-RSVS is integrated with SU2 [25] an unstructured flow solver which has been used extensively to tackle aerodynamic shape optimisation (ASO) problems. Recognising that parameterisation methods for three dimensional aerodynamics are usually deformative, a method for the exploiting existing high quality geometries and volume meshes while still exploiting the topological flexibility of the 3D-RSVS is proposed in Section 1.5.3.

1.5.1 Flow Solving for Optimisation of 3D Topology

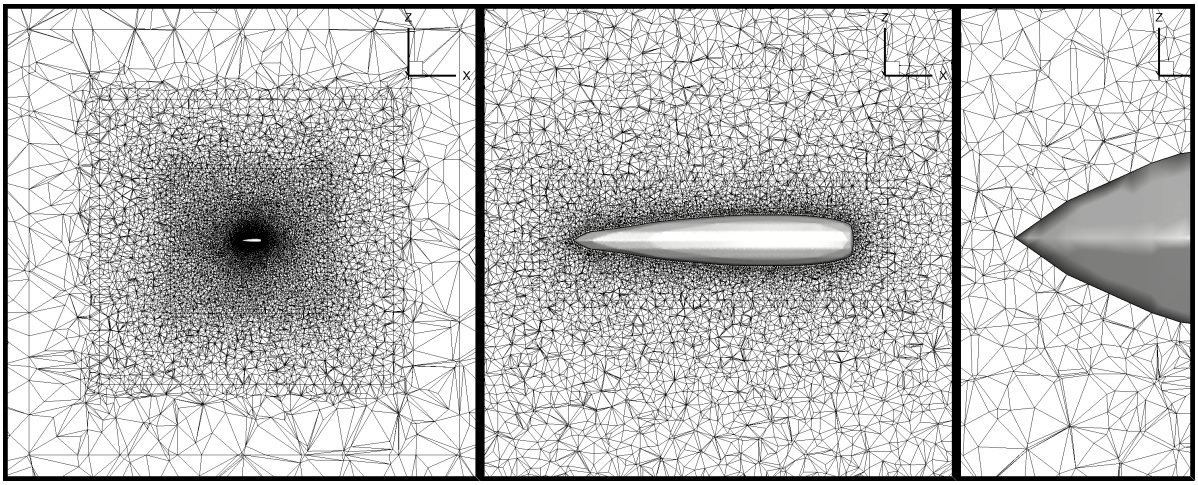
In order to perform optimisation using the 3D-RSVS parameterisation a framework must be capable of running a flow analysis from the geometry without any user input. Meshing of the geometry is done using TetGen [31], the three dimensional version of the Delaunay triangulation tool used in Chapter **??**. TetGen generates tetrahedral meshes which can be tailored to a range of applications. In this work TetGen is integrated with the 3D-RSVS to generate volume meshes from the triangulated geometries. Mesh density is controlled in a cut-cell-like fashion: element volume constraints are applied on the mesh through TetGen, with the constraints relaxing away from the body. Figure 1.25 shows the surface and volume meshes for the truncated ogive which was displayed in Figure 1.8.

This approach to mesh generation allows the tuning of element density at the surface to achieve the required resolution of flow properties. The mesh generated using TetGen is suitable for the solution of inviscid compressible flow conditions. An example flow solution using SU2 is shown in Figure 1.26.

Currently these meshes are not suitable to compressible and viscous flows: the Reynolds-Averaged Navier-Stokes (RANS) equations are very sensitive to mesh quality at the surface, and require a smooth prismatic discretisation of the volume in the boundary layer. While tools exist to grow boundary layer meshes around complex geometries, they are often designed for industrial scale applications and require a human in the loop; meshing for RANS is still one of the bottle necks to aerodynamic design [32] and beyond the scope of this thesis. The scope of the challenges is the reason for adopting a modular

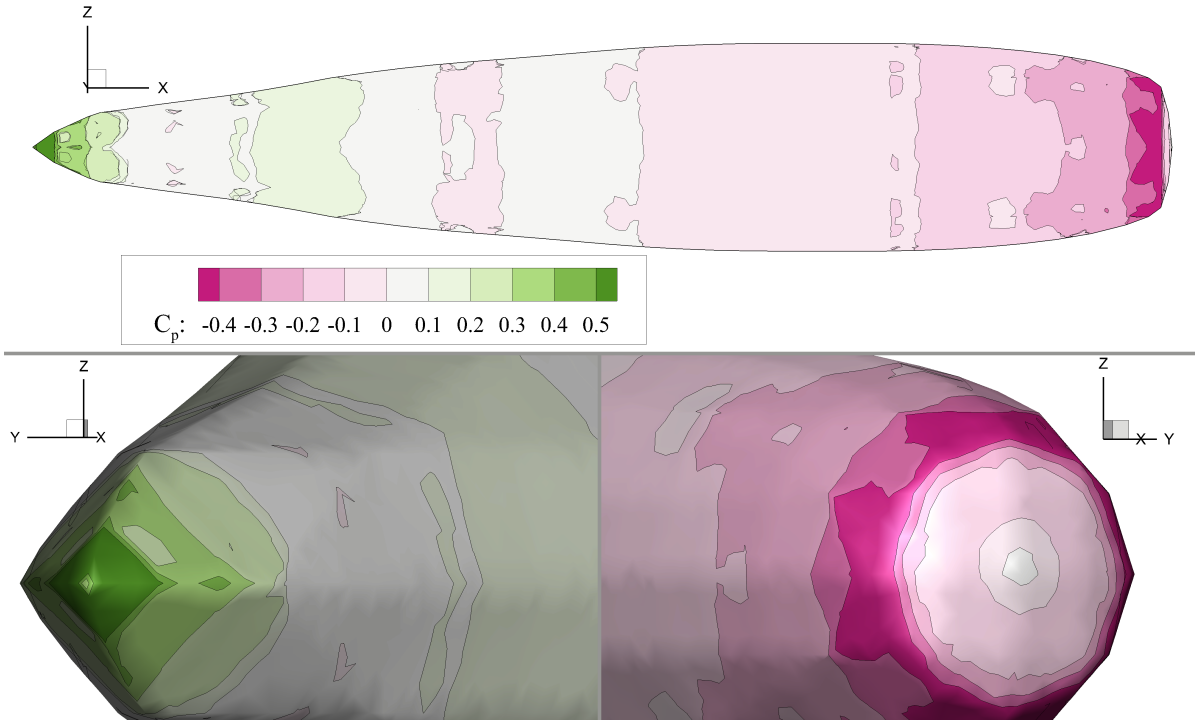


(a) Surface mesh generated by TetGen from a 3D-RSVS geometry.

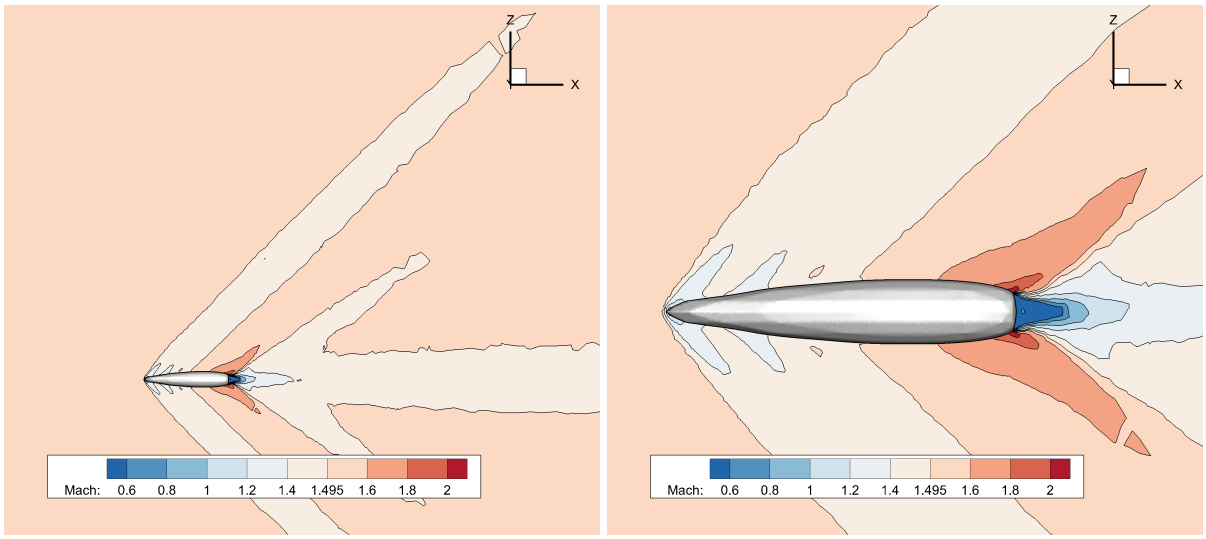


(b) Volume mesh generated by TetGen.

Figure 1.25: Surface and volume meshes for flow analysis of a 3D-RSVS geometry generated by TetGen.



(a) Surface C_p plots from SU2.



(b) Volume Mach number plots from SU2.

Figure 1.26: Surface pressure coefficients and volume Mach number plots on a 3D-RSVS geometry for a free stream Mach number of 1.5 in Euler flow using SU2 ($C_D = 0.00885$)

approach in the development of framework components.

An alternative to the combination of TetGen and SU2 developed here exists commercially in the form of Cart3D [2]. Originally developed by Aftosmis et al. [1], Cart3D implements an efficient three dimensional cut-cell meshing with adjoint based grid adaption to achieve high quality flow results with limited user input. Originally developed for inviscid flows, an interactive boundary layer method has been integrated to provide evaluation of viscous trends in the Cart3D framework [34].

While this system would be suitable for meshing and flow solving, adjoint design sensitivities for the geometry were not commercially available (although available to US federal agencies), ultimately motivating the assembly of TetGen and SU2 to solve compressible flows around complex geometries. Indeed, in three dimensions, the number of design variables and the cost of evaluating a flow solution are such that gradient based optimisation with sensitivities calculated through adjoint flow solvers are the norm. These avoid the excessive computation times that would be the result of agent based optimisation or finite difference gradients.

1.5.2 Integration of the 3D-RSVS with Gradient Based Optimisation

In order for the 3D-RSVS to be effective inside an adjoint gradient based flow solver, the sensitivities of the design variables (the VOS values $\mathbf{V}$) to the aerodynamic forces (C_D, C_L, C_M) need to be computed. To get the sensitivity of the flow to the design variables ($\nabla_{\mathbf{v}}\mathscr{F}$), the chain rule can be used to separate the contributions of the flow solver and the parameterisation process (Equation 1.9). For a flow solution $\mathscr{F}$, calculated on a mesh with vertices at positions $\mathbf{p}$; an adjoint flow solver takes care of calculating the derivative of the aerodynamic coefficients to the position of surface vertices of the aerodynamic mesh ($\nabla_{\mathbf{p}}\mathscr{F}$).

$$\mathscr{F}(\mathbf{p}(\mathbf{V})) = \begin{bmatrix} C_L \\ C_D \\ \vdots \end{bmatrix} \qquad \nabla_{\mathbf{v}}\mathscr{F} = \nabla_{\mathbf{p}}\mathscr{F} \times \nabla_{\mathbf{v}}\mathbf{p} \tag{1.9}$$

The remaining contribution $\nabla_{\mathbf{v}}\mathbf{p}$, is the sensitivity of the surface points to changes of volume fractions used to design the 3D-RSVS geometry. Because the parametrised contour is the result of an SQP process where the volume fraction is a constraint on the design (see Equation 1.2), the calculation of derivatives benefits from a wealth of previous

research into sensitivity analyses for SQP algorithms [9, 14]. Equations 1.10 and 1.11 show how results for the sensitivity analysis of non-linear programs from Buskens and Mauer [9] can be applied directly to the 3D-RSVS to calculate the change in the r-surface due to small changes in VOS.

$$\begin{pmatrix} \nabla_{\mathbf{v}}\mathbf{d} \\ \nabla_{\mathbf{v}}\boldsymbol{\lambda} \end{pmatrix} = -\begin{pmatrix} \mathbf{H}_{\mathbf{d}}\mathscr{L} & \nabla_{\mathbf{d}}\mathbf{h}^T \\ \nabla_{\mathbf{d}}\mathbf{h} & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} \left(\nabla_{\mathbf{d}}^T\nabla_{\mathbf{v}}\right)\mathscr{L} \\ \nabla_{\mathbf{v}}\mathbf{h} \end{pmatrix} \tag{1.10}$$

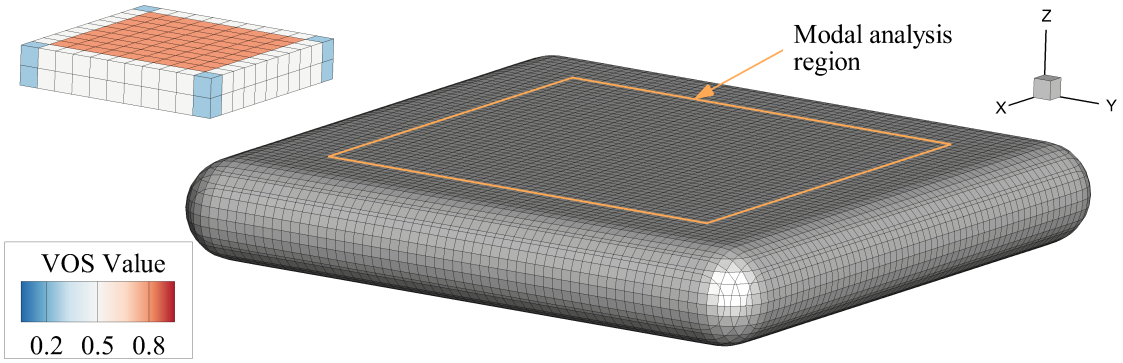
$$\nabla_{\mathbf{v}}\mathbf{p}_i = \nabla_{\mathbf{v}}\left(d_i\Delta\mathbf{g}_i + \mathbf{g}_{i,1}\right) = \Delta\mathbf{g}_i\,\nabla_{\mathbf{v}}d_i \tag{1.11}$$

As was the case for the two dimensional RSVS simply extracting the responses, is likely to be insufficient to ensure the smooth integration with gradient based optimisation. The empirical study in Figure 1.27 shows that the oscillatory response of the RSVS in 2D can also be found in 3D (c.f. Section **??**). These oscillations are likely to be detrimental to the optimisation behaviours, however these can be resolved the same way they were in 2D: through design variable combination. Figure 1.27d shows that, by grouping cells and changing multiple VOS values as a single "mode", a smoother response is achieved. Defining the exact grouping of VOS cells and their relative response can be pre-calculated or optimal groups can be found for each VOS response layout; these processes were derived in 2D in Section **??**.

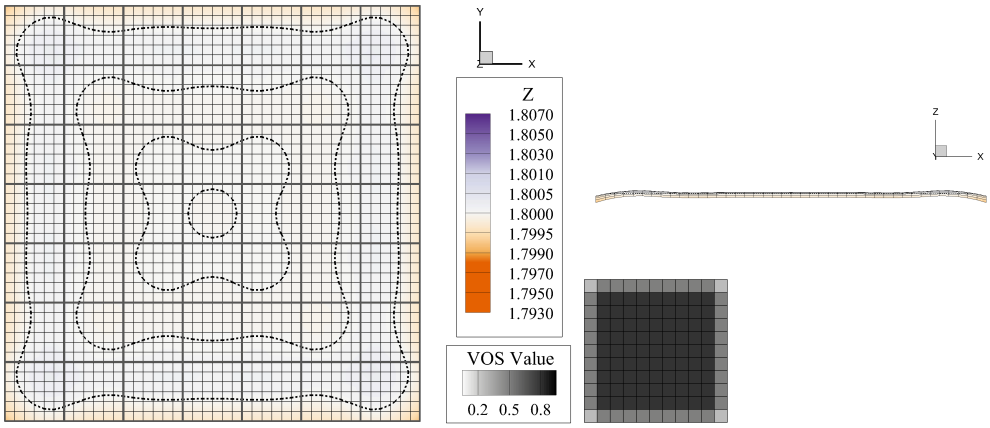
1.5.3 Exploiting Topological Flexibility for Part Design

One drawback of the RSVS is that it is a constructive approach; while this is a necessary feature to enable topological optimisation it means that existing surface and volume meshes cannot be reused. The implication is that the 3D-RSVS cannot be used on large scale industrial geometries, because, while it is theoretically capable to reproduce the entire geometry, in practice it would be an extremely difficult endeavour. Instead, it is suggested that only a portion of a complex geometry would be parameterised, allowing the optimisation of a part without affecting the rest of the geometry; the 3D-RSVS and the original discretisation of the geometry being merged through mesh surgery. Figure 1.28 shows what this approach would look like for the optimisation of winglets for the NASA common research model (CRM) [24].

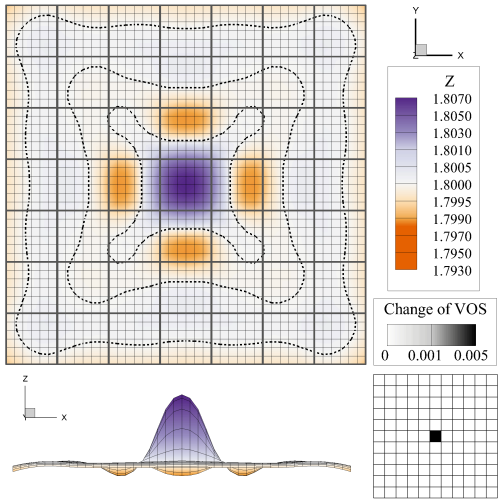
Merging the original surface and the partial surface defined by the RSVS relies on compatible discretisations. Thanks to the flexibility in the shape of VOS cells and topology of the snaking grid; the 3D-RSVS is capable of matching the discretisation of



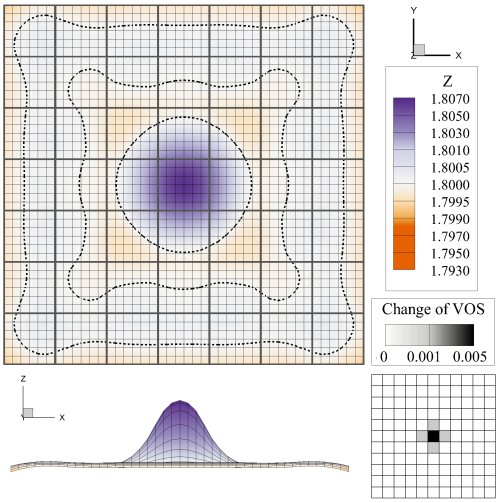
(a) 3D-RSVS surface used for modal analysis. The volume fractions are 0.8 in the flat faces, 0.503 at the edges and 0.26 in the corners.



(b) Analysis of the flatness of the original top surface of the geometry



(c) Oscillatory response of the surface.



(d) Response smoothed with a 20% disturbance in neighbouring cells.

Figure 1.27: Empirical analysis and smoothing of the response of the 3D-RSVS to small changes in volume fractions

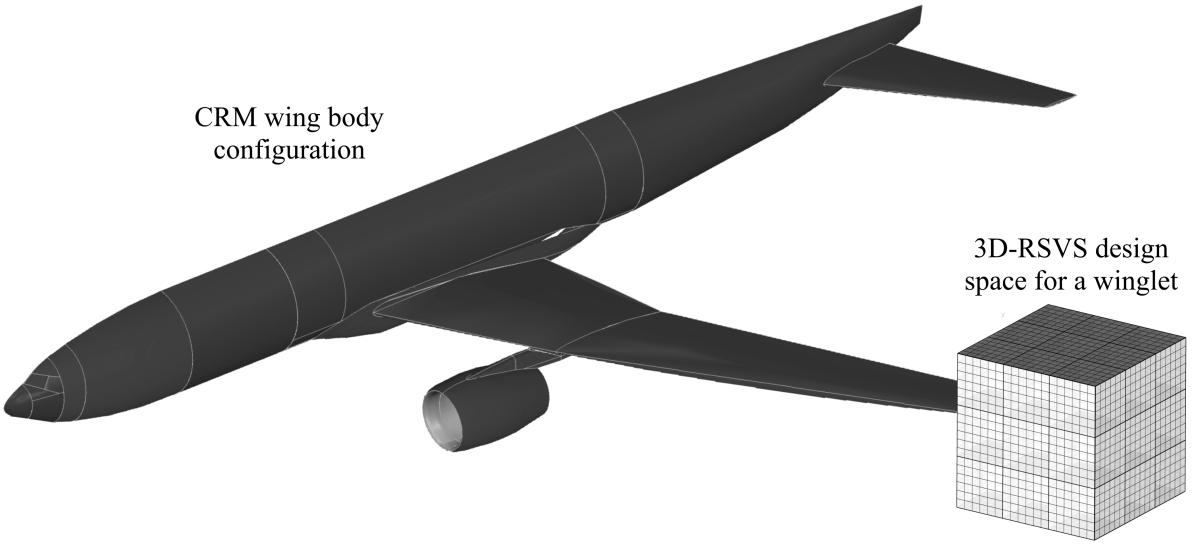


Figure 1.28: RSVS design space embedded in the CRM wing-body-tail configuration. In this configuration only the winglet portion of the design would be modified by the 3D-RSVS.

the original mesh permitting the merge of the geometries at their boundary. Starting from a high quality volume mesh, the region of the design space parameterised by the RSVS can be excavated, the TetGen mesher can then be used to generate a "good enough" mesh which conforms to the RSVS boundary and the cavity inside the original mesh. While this approach would not be suitable for drag prediction purposes, it would provide an indication of relative performance sufficient for optimisation.

1.6 Path to Robust Design Using the 3D-RSVS

While the flexibility afforded by the r-surface and the CMC-like smoothness properties of the 3D-RSVS make it a natural fit for optimisation frameworks some limitations remain with the current implementation. Addressed in the following sections are three key challenges, and possible mitigations, that prevent efficient topology optimisation using the 3D-RSVS. Those are:

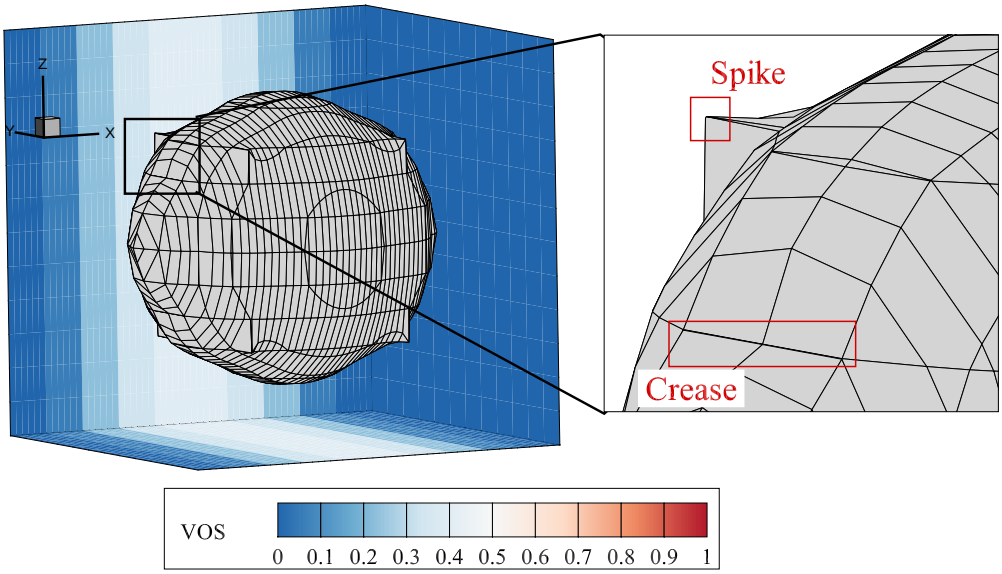
- convergence and stability issues in the form of oscillations due to the SQP step;
- the computational cost of repeatedly solving the quadratic program (QP);
- smoothness of the final geometry being insufficient and only controlled by the snaking grid.

1.6.1 Convergence and Stability

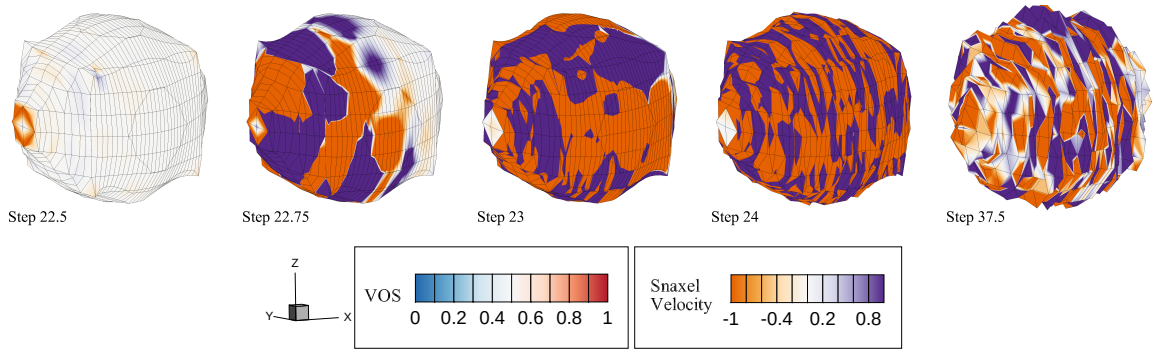
Optimisation using the 3D-RSVS in general and gradient based optimisation in particular, require a smooth response of the parameterisation to small changes of design variables. While the link to CMCs and minimal surfaces suggest that the analytical response of the 3D-RSVS is smooth away from topological changes; it is a significant challenge to get the discrete formulation to reproduce those properties. In its current state the parameterisation does not converge well: without reliable convergence small changes in VOS may lead to significant difference to the profile. Two main types of convergence failures have been observed: premature VOS convergence on profiles with non-minimal area; and profile instability. These are shown in Figure 1.29. These convergence difficulties are the result of the properties of the discrete mathematical program and its derivatives due to the evolution on the snaking grid: the interaction of the floating point mathematics of the SQP and the discrete, integer mathematics of the r-surface are a source of numerical issues.

To understand those issues, it is important to note the difference between the "global" problem solution that is desired; and the "local" solution which exists for every possible discretisation. Indeed, the design variables of the global problem are internal variables of the snaxels (d_i the normalised distance) and these change with each change of topology or each crossing of grid vertices: from a pure optimisation stand-point each of these discretisations is a separate optimisation problem with different design variables. With this realisation, it naturally follows that each of these separate problems, it they can solve the volume constraints, has at least one local minimum defined by those design variables. While many of those are technically "local" minima, in practice they lie at snaking grid vertices ($d_i = 0$ or 1), where the r-surface changes the connectivity for which this geometry is no longer optimal.

In itself this process is not problematic, however it becomes an issue when taking into consideration the properties of the derivatives as neighbouring snaxels converge on the same point. Both in two and three dimensions the objective function ($f(X,Y)$) relies on a square root term whose first derivative is undefined at the origin and whose second derivative is hyperbolic (Equation 1.12). Properties of the edge-length derivatives were discussed in Section **??** (c.f. Figures **??** and **??**); and motivated the use of a small constant ε to stabilise those derivatives around the origin; this process is shown in Equation 1.13.



(a) Premature convergence.



(b) Unstable oscillations.

Figure 1.29: Examples of the convergence issues of the 3D-RSVS.

$$f(X,Y)=\sqrt{X^2+Y^2} \qquad \frac{\partial f}{\partial X}=\frac{X}{\sqrt{X^2+Y^2}} \qquad \frac{\partial^2 f}{\partial X^2}=\frac{Y^2}{\left(X^2+Y^2\right)^{3/2}} \tag{1.12}$$

$$f_\varepsilon(X,Y)=\sqrt{X^2+Y^2+\varepsilon} \qquad \frac{\partial f_\varepsilon}{\partial X}=\frac{X}{\sqrt{X^2+Y^2+\varepsilon}} \qquad \frac{\partial^2 f_\varepsilon}{\partial X^2}=\frac{Y^2+\varepsilon}{\left(X^2+Y^2+\varepsilon\right)^{3/2}} \tag{1.13}$$

The presence of undefined and infinite values at the boundaries of validity of a discretisation are a significant computational hazard and must be avoided. Now considering the properties of the stabilised derivatives when neighbouring snaxels are lying at the same snaking grid vertex ($X=Y=0$), the first derivative is 0 and the second derivative is a large positive number. Depending on the state of the VOS constraints this point can be an artificial minimum which satisfies the KKT conditions; unfortunately these artificial

Table 1.3: Limits and values of the objective function and the two stabilisation schemes.

	$f(X,Y)$ (eq.1.12)	$f_\varepsilon(X,Y)$ (eq.1.13)	$f_\sigma(X,Y)(eq.1.14)$
h	0	$\sqrt{\varepsilon}$	$\sqrt{\sigma_X^2+\sigma_Y^2}$
$\frac{\partial h}{\partial X}$	$\to \pm 1$	0	± 1
$\frac{\partial^2 h}{\partial X^2}$	$\to \pm\infty$	$1/\sqrt{\varepsilon}$	$\sigma_Y^2/\sqrt{\sigma_X^2+\sigma_Y^2}$

minima are indistinguishable from the actual global minima which may lie close or at a snaking grid vertex. The presence of these minima explains the premature convergence on geometries with sub-optimal area.

Both the original and stabilised second derivatives get very large as both X and Y reduce to 0; this can be the initial cause of profile instability. Indeed these large values sometimes cause very large (10^7) snaxel velocities which if left unchecked would lead to rapid traversal of the snaking grid based on false information. In the current implementation this is avoided by applying aggressive reduction of the step size which inflicts an additional convergence penalty on the 3D-RSVS with no guarantee that it will not lead to some oscillations.

$$\begin{gathered}\text{Defining } X_\sigma = X + sign(X)\sigma \text{ for } X \text{ and similarly } Y_\sigma \text{ for } Y \\ \text{where}: sign(x) = 1 \ \forall x \in \mathbb{R}^+ \text{and } sign(x) = -1 \ \forall x \in \mathbb{R}^- \\ f_\sigma(X,Y) = \sqrt{X_\sigma^2+Y_\sigma^2} \qquad \frac{\partial f_\sigma}{\partial X} = \frac{X+sign(X)\sigma}{\sqrt{X_\sigma^2+Y_\sigma^2}} \qquad \frac{\partial^2 f_\sigma}{\partial X^2} = \frac{Y^2+2\sigma|Y|+\sigma^2}{\left(X_\sigma^2+Y_\sigma^2\right)^{3/2}}\end{gathered} \tag{1.14}$$

A theoretically better stabilising equation is suggested in Equation 1.14: by avoiding a rapid change of the first derivative from approximately 1 to 0 as X tends to 0 the second derivative can be better behaved. Limits and values of these approaches to objective function stabilisation are compiled in Table 1.3 and their behaviour is shown in Figure 1.30. By having a non 0 gradient at the origin the frequency of spurious minima may be decreased. The challenge of this formulation remains the definition of the $sign(0)$ which is both -1 and 1 depending on the approach direction. While not currently implemented because of this perceived limitation, this formulation of the objective could improve the reliability of 3D-RSVS convergence.

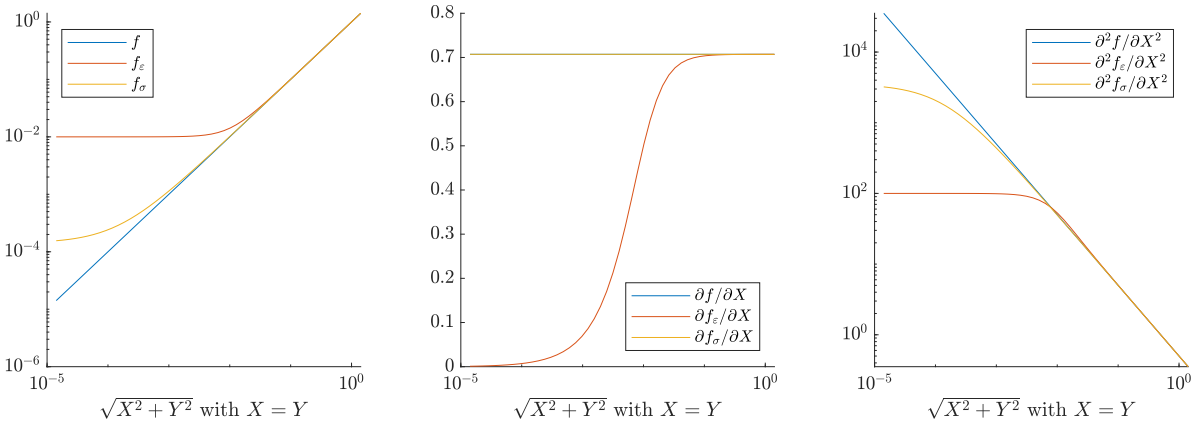


Figure 1.30: Evolution of the natural objective and the two proposed stabilising methods, and their derivatives close to $X = Y = 0$

1.6.2 Solution of the Quadratic Program (QP)

SQP methods are well known for their quadratic convergence on a constrained minima, even when using approximations of the Hessian; this property is very powerful and makes them the obvious choice to tackle optimisation of smooth functions. The “Surface Evolver”, certainly the most widely used and studied discrete CMC solver, uses a Hessian based quadratic solver, stating that local convergence may be achieved in as low as four steps [7]. In the case of the 3D-RSVS, for large numbers of design variables (10^3 snaxels and above) the computational cost of the parameterisation is dominated by finding sequential solutions to the QP. Normally Newton based methods make up for this limitation by requiring very few steps to converge; unfortunately because of the convergence challenges detailed in the previous paragraphs, the number of steps for RSVS convergence can be large (hundreds).

In order to mitigate the computational cost of the QP solution the structure of the Hessian matrix can be exploited to accelerate the solution process. Indeed the 3D-RSVS system is sparse, with the width of the central diagonal unchanging with system size. This property means that sparse matrix algebra can be used to speed up and reduce the memory footprint of the QP solution process; current experiments suggest that the cut-off for a reduction in computational cost is between three and four thousand snaxels.

Algorithms used for sparse mathematics are also more easily parallelisable than their dense equivalents; the structure of the problem means that regions of the design space can be more easily separated. The matrix mathematics library used to solve the 3D-RSVS QP, Eigen, supports parallel sparse solvers out of the box [12].

While the cost of the QP can be reduced, for large surfaces it is likely to remain

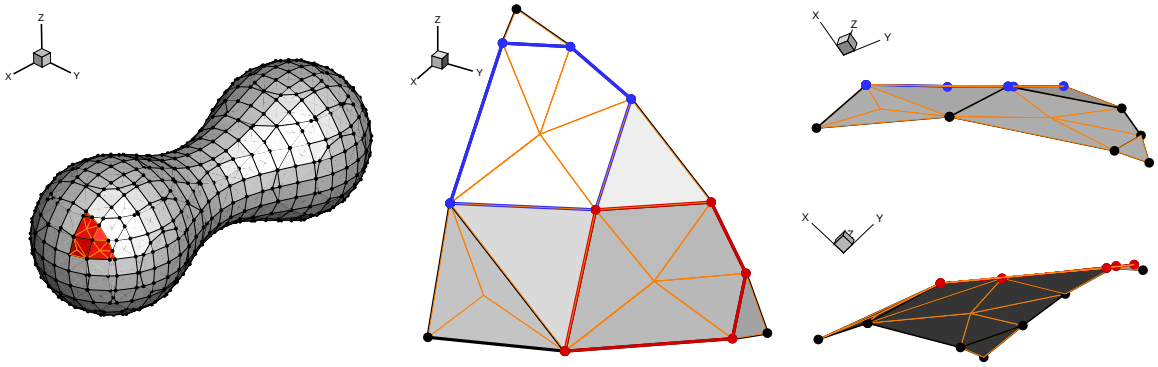


Figure 1.31: Examples of flat faces in curved region of the design space.

a bottle neck; as such it may be necessary to consider other optimisation algorithms. A host of methods have been used and specifically tuned for large scale constrained minimisation: sequential linear programming, constrained conjugate gradient methods, interior point methods, multipoint exponential approximation (MPEA) or the quadratic multipoint exponential approximation (QMEA) [10].

The main limitation of non-quadratic methods is that they do not specify a step length, which can lead to slow convergence if a robust trust region method is not available. It is interesting to note that the limits on step size already in place in the snaking algorithm implicitly set a trust region far from the optimum. Since the benefits of quadratic programming lie mostly close to the optima, it may be beneficial to swap dynamically between algorithms; exploiting the simplicity and cheap cost of linear solvers in the far field, only to use the SQP to arrive rapidly at the minimum once the algorithm is in the neighbourhood of a minimum.

1.6.3 Smoothness of the Output Surface

Since the r-surface evolves on a predetermined grid according to specific rules, the surface vertex distribution is not explicitly controlled. In practice, this leads to surface triangles with poor aspect ratios and wildly different sizes which is generally not conducive to robust physical modelling. While the surfaces generated by the 3D-RSVS are in general smooth, the lack of explicit control over the centroid of cells often leads the parameterisation to generate flat faces even in curved regions of the geometry. This behaviour creates artificially flat groups of triangles in a part of the geometry which, according to the governing equation, should have a smooth curvature. Figure 1.31 shows an example of large, flat, pentagonal faces in a region of otherwise uniform curvature.

The flattening observed in Figure 1.31 could be resolved by providing the centroid with its own design variable, limiting movement to be in a normal direction to the face that spawned it. This additional design variable, would enable the SQP to smooth out any panelling created by the interaction of the SQP and the centroid calculation process. The increase in number of design variables due to the addition of almost 1 per face, would be offset by a reduction in the density of the snaking grid.

While it reduces the impact of the triangulation, this approach does not improve the control over element size and shape. This approach also has a major downside in that it looses guarantees of water-tightness and the absence of self intersection.

1.6.4 De-restricting the Surface

The benefits of using a restricted surface are undeniable when it comes to resolving the topology of a geometry and the connectivity between VOS cells. The efficiency and reliability of the integer collision detection process make it possible to consider very complex geometries with little computational cost; however that is not the only requirement of a three dimensional topology parameterisation method. The accumulation of issues at the interface between the integer mathematics of the restricted surface method and the continuous solutions of the governing equation suggest that a radically different approach may be warranted. Indeed convergence, stability and smoothness difficulties are all, in part, attributable to the use of a restricted surface. This is not to say that it is not possible to solve these issues within the restricted framework, but it may be more efficient and robust to consider alternatives.

Inspired by the “Surface Evolver” [6] and its descendants, the possibility to take the r-surface off the grid is considered. De-restricting the surface would lose the very efficient topology control, but gain smoother profile progression. For this reason a two step process is proposed: first the 3D-RSVS is evolved as described in this chapter to resolve topology and cell connectivity; then it is separated from the snaking grid to finish its convergence without topology change. By separating the surface from the grid, element distribution can be controlled; convergence is no longer hampered by changes of design space due to changes of connectivity; and fewer QP solutions will be necessary.

Using the idea of de-restricting the surface, the complete 3D-RSVS process would become:

1. **Resolve topology through the restricted surface:** Use the existing process on a coarse snaking grid to rapidly converge the topology and cell connectivity.

2. **Place snaxels in "surface space":** generate a pseudo snaking grid which is only a set of edges normal to the current surface. This allows: maximum code re-use; the creation of implicit trust-regions, through the length of each edge; and enforcement of the VOS boundary, ensuring that the snaxels stay on VOS cell boundaries. At this stage, the surface may be refined.
3. **Evolve the "de-restricted" surface:** iterate the position of the snaxel along the pseudo-grid edges, according to the SQP. The edges of the pseudo-grid need not be constant through the convergence process allowing simultaneous resolution of the constraints, minimisation of the objective and redistribution of surface elements. A method similar to the CMC-CVT of Pan et al. [26] is suggested.

1.7 Summary

In its current state the 3D-RSVS cannot compete with existing parameterisations, it does not have the robustness expected of such methods. Fortunately the causes of these limitations are understood, evidence shows that they are not intrinsic to the formulation, and, ways to resolve them have been outlined.

The original goal of three dimensional topology optimisation for aerodynamic bodies is very close: the necessary flow solvers, meshers and optimisation methods are all already in regular use even. While the missing link remains a parameterisation which will describe arbitrary topologies with sufficient smoothness, compactness and flexibility for aerodynamic application; this chapter has shown the feasibility of parameterising shape and topology to an aerodynamic optimisation standard.

Beyond its native application in aerodynamic parameterisation, the scope of the 3D-RSVS is much broader. It appears that the formulation of the 3D-RSVS lets it bridge the gap between level set methods and traditional density methods; while establishing a footing for the parameterisation in differential geometry. The natural formulation of the RSVS gives rise to very intuitive behaviours, and the simplicity of the definition leads to a very general method. The intuitiveness of the method makes it suited for applications in which it is controlled by a designer, notably for prototyping. In fact the 3D-RSVS has the potential to be a framework for the design of smooth surfaces: new types of constraints, and, changes of objective function could impart on the 3D-RSVS most of the features of a CAD framework.

BIBLIOGRAPHY

[1] M. J. AFTOSMIS, M. J. BERGER, AND J. E. MELTON, *Robust and efficient Cartesian mesh generation for component-based geometry*, AIAA Journal, 36 (1998), pp. 952–960.

[2] N. AMES, *CART3D*. `https://www.nasa.gov/ames-partnerships/technology/technology-opportunity-cart3d`. [Online; accessed 01/12/2019].

[3] S. BISCHOFF AND L. P. KOBBELT, *Snakes with topology control*, Image Rochester NY, (2003), p. 10.

[4] BOEING, *737 MAX Gallery*. `http://www.boeing.com/commercial/737max/#/design-highlights/max-efficiency/max-at-winglet/`. [Online; accessed 13/06/2019].

[5] P. T. BOGGS AND J. W. TOLLE, *Sequential Quadratic Programming*, Acta Numerica, 4 (1995), pp. 1–51.

[6] K. A. BRAKKE, *The Surface Evolver*, Experimental Mathematics, 1 (1992), pp. 141–165.

[7] K. A. BRAKKE, *The Surface Evolver and the stability of liquid surfaces*, Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 354 (1996), pp. 2143–2157.

[8] D. A. BURDETTE AND J. R. MARTINS, *Design of a transonic wing with an adaptive morphing trailing edge via aerostructural optimization*, Aerospace Science and Technology, 81 (2018), pp. 192–203.

[9] C. BUSKENS AND H. MAURER, *SQP-methods for solving optimal control problems with control and state constraints: Adjoint variables, sensitivity analysis and real-time control*, Journal of Computational and Applied Mathematics, 120 (2000), pp. 85–108.

[10] R. A. CANFIELD AND M. S. ELDRED, *Quadratic multipoint exponential approximation for optimization and uncertainty quantification*, AIAA Scitech 2019 Forum, (2019), pp. 1–8.

[11] EIGEN, *Dense Decomposition Benchmark*. `https://eigen.tuxfamily.org/dox/group__DenseDecompositionBenchmark.html`. [Online; accessed 23/11/2019].

[12] ——, *Multi Threading*. `https://eigen.tuxfamily.org/dox/TopicMultiThreading.html`. [Online; accessed 23/11/2019].

[13] ——, *Sparse Decomposition Tutorial*. `https://eigen.tuxfamily.org/dox/group__TutorialSparse.html`. [Online; accessed 23/11/2019].

[14] A. V. FIACCO AND Y. ISHIZUKA, *Sensitivity and stability analysis for nonlinear programming*, Annals of Operations Research, 27 (1990), pp. 215–235.

[15] P. E. GILL, W. MURRAY, AND M. A. SAUNDERS, *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*, SIAM Journal on Optimization, 12 (2002), pp. 979–1006.

[16] G. GUENNEBAUD, B. JACOB, ET AL., *Eigen v3*. http://eigen.tuxfamily.org, 2010.

[17] E. T. INSTITUTE, *Helicoidal Soap Bubble*. `https://plus.maths.org/content/abel-prize-2019`. [Online; accessed 28/11/2019].

[18] S. KHOSRAVI AND D. W. ZINGG, *Aerostructural Optimization of Drooped Wings*, Journal of Aircraft, (2017), pp. 1–8.

[19] L. P. KOBBELT AND S. BISCHOFF, *Parameterization-free active contour models with topology control*, The Visual Computer, 20 (2004), pp. 217–228.

[20] D. A. MASTERS, N. J. TAYLOR, T. RENDALL, AND C. B. ALLEN, *Three-Dimensional Subdivision Parameterisation for Aerodynamic Shape Optimisation*, 55th AIAA Aerospace Sciences Meeting, (2017), pp. 1–17.

[21] D. A. MASTERS, N. J. TAYLOR, T. C. S. RENDALL, AND C. B. ALLEN, *Multilevel Subdivision Parameterization Scheme for Aerodynamic Shape Optimization*, AIAA Journal, 55 (2017), pp. 3288–3303.

[22] T. MCINERNEY AND D. TERZOPOULOS, *T-snakes: Topology adaptive snakes*, Medical Image Analysis, 4 (2000), pp. 73–91.

[23] R. McNeel and Associates, *Rhinoceros 3D*. https://www.rhino3d.com/. [Online; accessed 28/11/2019].

[24] NASA, *Common Research Model*. https://commonresearchmodel.larc.nasa.gov/geometry/stp-files/. [Online; accessed 03/12/2019].

[25] F. Palacios, M. R. Colonno, A. C. Aranake, A. Campos, S. R. Copeland, T. D. Economon, A. K. Lonkar, T. W. Lukaczyk, T. W. R. Taylor, and J. J. Alonso, *Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design*, 51st AIAA Aerospace Sciences Meeting, (2013).

[26] H. Pan, Y.-K. Choi, Y. Liu, W. Hu, Q. Du, K. Polthier, C. Zhang, and W. Wang, *Robust modeling of constant mean curvature surfaces*, ACM Transactions on Graphics, 31 (2012), pp. 1–11.

[27] A. D. J. Payot, *RSVS3D*. https://github.com/payoto/rsvs3d. [Online; accessed 08/06/2020].

[28] J. Pérez, *A New Golden Age of Minimal Surfaces*, Notices of the American Mathematical Society, 64 (2017), pp. 347–358.

[29] D. Piker, *Kangaroo Physics*. https://www.food4rhino.com/app/kangaroo-physics. [Online; accessed 28/11/2019].

[30] G. Piola, *Ferrari formula 1 front wing*. https://www.formula1.com/content/fom-website/en/latest/technical/2015/9/ferrari-sf15-t---low-downforce-front-wing/_jcr_content/featureContent/manual_gallery/image1.img.2048.medium.jpg/1441192443879.jpg. [Online; accessed 2017-04-05].

[31] H. Si, *TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator*, ACM Transactions on Mathematical Software, 41 (2015), pp. 1–36.

[32] J. Slotnick, A. Khodadoust, J. Alonso, W. Gropp, and D. Mavriplis, *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*, tech. rep., 2014.

[33] soapbubble.dk, *Catenoid Soap Bubble*. https://www.soapbubble.dk/en/articles/former. [Online; accessed 28/11/2019].

[34] A. TECHNOLOGIES, *Viscous CART3D*. https://www.aerion-tech.com/cart3dv-docs/. [Online; accessed 01/12/2019].

[35] E. L. THOMAS, D. M. ANDERSON, C. S. HENKEE, AND D. HOFFMAN, *Periodic area-minimizing surfaces in block copolymers*, Nature, 334 (1988), pp. 598–601.

[36] H. WENTE, *Counterexample to a conjecture of H. Hopf*, Pacific Journal of Mathematics, 121 (1986), pp. 193–243.

[37] H. C. WENTE, *An existence theorem for surfaces of constant mean curvature*, Journal of Mathematical Analysis and Applications, 26 (1969), pp. 318–344.

List of Acronyms

Common acronyms

AI artificial intelligence

ANN artificial neural network

CAD computational aided design

CFD computational fluid dynamics

FEM finite element method

FFD free-form deformation

GPL General Public License

ML Machine Learning

NACA National Advisory Committee for Aeronautics

NASA National Aeronautics and Space Administration

NURBS non-uniform rational B-Splines

POD proper orthogonal decomposition

RMS root mean squared

RMSE root mean squared error

RBF radial basis function

SVD singular value decomposition

UIUC University of Illinois in Urbana Champaign

Acronyms for optimisers and optimisation methods

ACO ant colony optimisation

BFGS Broyden-Fletcher-Goldfarb-Shanno

CG conjugate gradient

DE differential evolution

EA evolutionary algorithm

GSA gravitational search algorithm

GA genetic algorithm

IPM interior point method

KKT Karush-Kuhn-Tucker condition

L-BFGS limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS)

MDO multi-disciplinary optimisation

MMA Method of Moving Asymptotes

MMO multi-modal optimisation

MOOP multi-objective optimisation problem

NLP non-linear program

PSO particle swarm optimiser

QD quality diversity

QP quadratic program

SALP simplex algorithm for linear programming

SNOPT Sparse Non-linear OPTimizer

SQP sequential quadratic programming

Structural topology optimisation

CSD computational structural dynamics

STO structural topology optimisation

BESO bi-directional ESO

ESO evolutionary structural optimisation

FE finite element

HJ Hamilton-Jacobi

LSF level-set function

LSM level-set method

SIMP solid isotropic material with penalisation

Specialist acronyms

ADODG aerodynamics design optimisation discussion group

ASO aerodynamic shape optimisation

ATO aerodynamic topology optimisation

CMC constant mean curvature

CRM NASA common research model

DNS direct numerical simulation

LES large eddy simulation

MLSO multi-level subdivision optimisation

NS Navier-Stokes

SLIC simple line interface contour

PLIC piecewise linear interface contour

RANS Reynolds-Averaged Navier-Stokes

RSVS restricted snakes volume of solid

3D-RSVS restricted surface volume of solid

r-snake restricted snake

r-surface restricted surface

SPH smoothed particle hydrodynamics

VLM vortex lattice method

VOF volume of fluid

VOS volume of solid

Kulfan's WTT Kulfan's Wind Tunnel Tolerance