Towards Automated, Adaptive, and Mesh-free CFD Modelling for Rotorcraft: Point Cloud Generation

Tao Zhang
University of Leicester
Leicester, U.K.

George Barakos
University of Glasgow
Glasgow, U.K.

ABSTRACT

This work presents a novel point cloud generation framework, a key development towards a fully automated, adaptive, and mesh-free CFD workflow for complex engineering applications. The novelty of the method is the introduction of the Signed Distance Function (SDF) to guide advancing point layers in the near-body region. Insertion/removal mechanisms of points were also proposed. These ensure high-quality boundary layer resolution in the near-body region, regardless of the complexity and topology of the geometry. For the off-body region, Cartesian points are employed for smooth and adaptive point distributions. Compared to conventional advancing front point generation, the proposed method ensures surface-norm point distributions with consistent layer structures, which are critical for boundary layer resolution. Compared to the strand mesh generation, the current method offers greater flexibility with few restrictions on inter-layer connections. The proposed approach is tested for various 2D and 3D benchmark geometries, along with mesh-free modelling results using the generated point clouds.

INTRODUCTION

At present, meshing is still a strong bottleneck of automated CFD processes (Ref. 1). Poor and physics-violating meshes easily lead to inaccurate and non-physical modelling results. A particular challenge is the necessity of high-quality, body-fitted mesh elements for boundary layer resolutions. Today, the meshing process requires extensive user experience, and is iterative because of the lack of fore knowledge of flow solutions. Moreover, when the flow modelling is coupled with dynamics, structural elasticity, or design optimisation, where deformations and motions are necessary, meshing challenges amplify.

Over the decades, substantial efforts have been dedicated to advanced meshing approaches, and there have been fruitful outcomes in structured, unstructured, Cartesian, hybrid, and adaptive mesh generation (Refs. 2, 3). Along with interfacing techniques for extra flexibility, e.g. sliding/overset interfaces, we have achieved robust and accurate CFD modelling for many excessively complex engineering problems. Nevertheless, human interventions are inevitable during the meshing or re-meshing processes, especially for boundary layers. In this respect, the strand mesh approach (Refs. 4, 5) has made important progress towards automated near-body meshing, but it involves complex procedures and algorithms to maintain connectivity and quality.

Mesh-free CFD methods, with significantly lower dependency on meshes, represent a promising pathway towards a fully automated CFD workflow. Popular mesh-free approaches, whether built upon Lagrangian descriptions (e.g. Smoothed Particle Hydrodynamics (Refs. 6,7)) or generalised differential/integral operators (e.g. Differential Quadrature (Ref. 8) and Element-free Galerkin (Ref. 9)), operate on the most fundamental element: points. Even without additional information provided by meshes, e.g. control volumes and local biasing directions, these methods are capable of discretising and solving the governing flow equations simply upon points. The task of point cloud generation/adaptation is no doubt easier, and much more flexible than mesh generation. Mesh-free methods have found wide success in complex CFD problems, e.g. particle transport, multi-phase flows, star formulation etc. Nevertheless, intensive research efforts are still needed to improve the efficiency, accuracy, and sometimes convergence properties of these mesh-free methods, their weak dependency on mesh elements instability is a critical advantage for CFD automation.

Point cloud generation is neither trivial nor readily automatic, especially for aerospace applications such as rotorcraft modelling. Its principle is to place points where necessary to provide information, for the flow quantities with appropriate numerical schemes. Over the years, various point generation approaches (Ref. 10) have been developed for mesh-free modelling, e.g. advancing front point generation (Refs. 11, 12), Cartesian points (Ref. 13), and particle packing (Ref. 14). These methods are very good at filling computational domains

with points that have smooth and controllable distributions. However, for aerospace applications, they fall short in the critical boundary layer resolution.

Within the boundary layer, the flow is characterised by strong gradients in the surface normal direction, and their resolution is critical for aerodynamic force predictions and for modelling of critical flow features like separation. To accurately resolve the physics, the numerical scheme must acquire information in the stream-wise and stream-normal directions. Mesh-based schemes can handle this tricky situation via body-fitted elements, and via stream-wise, and stream-normal mesh connectivity. For mesh-free modelling with conventional point cloud generation, there is no guarantee that points exist along these critical directions, and there is no guarantee that information will be sought from these directions. This is a significant obstacle in applying mesh-free methods in aerospace problems. Many previous studies used points from meshes, but this compromises the mesh-free approach.

This work presents a novel automated, adaptive, and mesh-free CFD workflow, with particular focus on automatic point cloud generation methodology, driven by the Signed Distance Function (SDF). For near-body cloud generation, this proposed novel method blends ideas of advancing layer and strand mesh generation, and the challenge of determining surface-normal directions was overcome via the SDF. Point removal/insertion mechanisms on the advancing layers were also introduced to regulate the point distributions. Regardless of the geometric and topological complexity, this new approach ensures shape-conforming and surface-normal-preserving point distribution in the near-body region, which are critical for the boundary layer resolution. The off-body computational domain is filled with points adopted from Cartesian cell centres. Demonstrations of point clouds generation for complex 2D/3D geometries with various topologies are presented. Mesh-free modelling simulations using these point clouds were also carried out to verify and validate the proposed methodologies and implementations. The results showcase an important step towards a fully automatic, adaptive, and mesh-free CFD framework.

COLLOCATION-BASED MESH-FREE CFD METHODS

This work focuses on the Differential Quadrature (DQ) (Ref. 8) or Generalised Finite Difference Method (GFDM) (Refs. 15, 16) type mesh-free CFD methods, because of their excellent boundary resolution, and their similarity to the popular Finite Volume Methods in the formulations. The idea of these collocation-based, mesh-free discretisation schemes is to construct differential operators using shape functions upon discrete points.

Given an arbitrary and sufficiently smooth function $\phi(\mathbf{x})$ with $\mathbf{x} = (x, y, z)$ in a Cartesian system, known at a set of n_i discrete points centring at point i, $\Phi_i = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), ..., \phi(\mathbf{x}_{n_i})]^T$ in a local collocation ($j = 1, 2, ...n$), a local approximation can be constructed using a linear combination of shape functions $\mathbf{N_i}(\mathbf{x}) = [N_{i,1}, N_{i,2}, ..., N_{i,n_i}]$, such as:

$$\hat{\phi}(\mathbf{x}) = \mathbf{N_i}(\mathbf{x})_{(1\times n_i)} \Phi_{i(n_i\times 1)}. \tag{1}$$

Then the spatial derivatives of ϕ can be estimated from the approximation as:

$$\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}} \approx \frac{\partial \hat{\phi}(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \mathbf{N_i}(\mathbf{x})}{\partial \mathbf{x}} \Phi_i, \tag{2}$$

where $\frac{\partial \mathbf{N_i}(\mathbf{x})}{\partial \mathbf{x}}$ is the spatial derivative of the shape functions. This collocation-based differential operator in Equation (2) can be used for the discretisation of PDEs, in a manner similar to classic finite differences scheme. The approximation accuracy depends on the shape functions, as well as, the point collocations.

This principle can be applied to discretise the compressible Reynolds Averaged Navier-Stokes (RANS) equations for aerodynamic simulations. The inviscid part of the governing flow equations can be written in strong form for a Cartesian frame of reference as:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0, \tag{3}$$

where $\mathbf{U}$ is the vector of conservative flow variables, and $\mathbf{F}$ is the convective flux term.

Assuming a local point collocation centring at point i, with the local support (i.e. all points but the collocation centre i) denoted by the set $\mathbf{S}_i$ ($i \notin \mathbf{S}_i$), the flux gradients can be approximated at the collocation centre point i with shape functions as:

$$\frac{d\mathbf{U}_i}{dt} = -\sum_{j\in \mathbf{S}_i} \sum_{v=1}^{d} \frac{\partial N_{i,j}}{\partial x^v}(\mathbf{x}_j) \mathbf{F}^v_j, \tag{4}$$

where the superscript v denotes the number of spatial dimensions from 1 up to $d = 3$. $j \in \mathbf{S}_i$ are the supporting points for the collocation centre point i, and their Cartesian coordinates are $\mathbf{x}_j$. In this work, we have adopted complete second-order polynomials to approximate the flux terms, and the approximation was achieved using a weighted least-square approach. A novel weight-balancing scheme (Ref. 17) was adopted to improve the approximation upon highly irregular point clouds.

Due to the hyperbolic nature of the equations, the discretisation in Equation (4) needs further numerical dissipation or upwind schemes for stability. The upwind schemes can be incorporated by evaluating the flux terms at the middle point of i and j, which are denoted as $j - \frac{1}{2}$:

$$\frac{d\mathbf{U}_i}{dt} = -\sum_{j\in \mathbf{S}_i} \sum_{v=1}^{d} \frac{\partial N_{i,j}}{\partial x^v}(\mathbf{x}_{j-\frac{1}{2}}) \mathbf{F}^v_{j-\frac{1}{2}}, \tag{5}$$

where the flux terms $\mathbf{F}_{j-\frac{1}{2}}$ are reconstructed at the middle point, and Riemann solvers can be used to account for the upwind properties. In this work, Osher's scheme (Ref. 18) was adopted.

Equation (5) actually resembles the finite volume scheme (FVM) in the local point cloud, if we take the term $\frac{\partial N_{i,j}}{\partial x^v}(x^v_{j-\frac{1}{2}})$ as a virtual edge term. This similarity with FVM schemes is an important advantage for numerical implementations, as established FVM methods and codes can be readily transformed to mesh-free versions with minor modifications on the geometric terms. The realisation of hybrid, mesh-based/mesh-free methods is also feasible.

However, it should be noted that in Equation (5), the flux term between points i and j is not always reciprocal, as point i may not necessarily be in the point cloud centring at point j. Reciprocity is a necessary condition for flow conservation (Ref. 16). To alleviate this problem, this work has enforced reciprocal contributions between all points, i.e. adding i to j's support, if not already included. This additional restriction drives the present mesh-free scheme towards 'edge-based' schemes. This does not deteriorate the mesh-free flexibility, as the collocation or stencil edges are highly flexible and diverse (including crossover connections (Ref. 16)), and these connections are associated with adjustable weights.

The present work solves the ODE system of Equation (5) with an implicit, dual time-stepping scheme (Ref. 17), similar to conventional mesh-based FVM methods. The resulted non-linear system was linearised to second-order accuracy, and the subsequent linear system was solved using the Generalised Conjugate Gradient (GCG) approach with an OpenMP acceleration. The underlying Jacobian matrix depends solely on the flux reconstruction and on the Riemann solver, when there is no change in the clouds.

This work also included viscous terms and turbulence closures following the same approximation. For the viscous terms, the least-square approach could directly compute gradients and higher-order derivatives, but this requires higher-order basis functions. To accommodate lower-order basis functions, the current work adopted gradient computations which are similar in unstructured FVM (Ref. 19). The derivatives of the viscous flux are then approximated using the least-square approach. The resulted viscous flux is then averaged at the midpoint of i and j.

RANS turbulence closures were included in the discretisation as additional source terms. The one-equation Spalart-Allmaras (SA) model (Ref. 20) was used in the present work. The convective and diffusive terms were evaluated following the same procedures described for the inviscid and viscous terms.

NEAR-BODY POINT CLOUD GENERATION USING THE SIGNED DISTANCE FUNCTION (SDF)

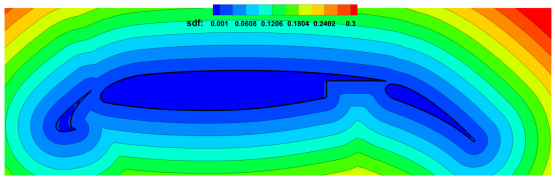
Signed Distance Functions (SDF) for point cloud generation

The Signed Distance Function (SDF) of a specific geometry Ω denotes the closest distance at any location $\mathbf{x}$ to the geometry boundary $\partial\Omega$, with the sign denoting whether the point is inside ($-$) or outside ($+$) the geometry. This is illustrated in Figures 1(a) to 1(b) for realistic 2D/3D shapes. The SDF of geometry Ω can be written as an implicit function $L(\mathbf{x})$:

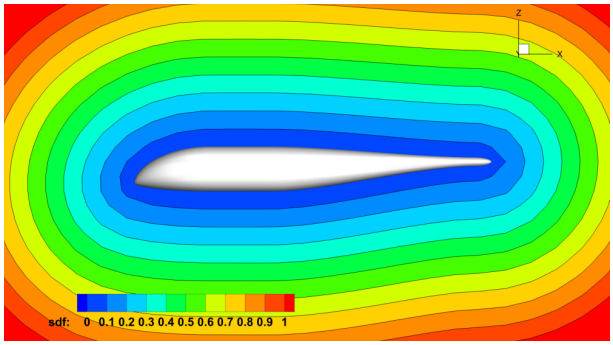
$$L(\mathbf{x}) = min(dist(\mathbf{x}, \partial\Omega)), \tag{6}$$

where $min(dist(\mathbf{x}, \partial\Omega))$ is the closest distance from $\mathbf{x}$ to the geometry boundary $\partial\Omega$, and this distance is positive when $\mathbf{x}$ is outside Ω, negative when $\mathbf{x}$ is inside Ω.

The SDF field can be regarded as the result of wavefront propagation from the wall surface at a constant speed, i.e. the contour lines in Figures 1(a) to 1(b). While near the wall surface, the contour lines closely follow the wall shapes regardless of the complexity. In practice, SDF is closely related to the concept of level set, and $L = 0$ is an implicit representation of the geometry boundary itself. As the waves travel further away, the fronts gradually merge to de-featured shapes, and eventually become a circle or sphere at infinity. The transition is continuous and smooth.



(a) Multi-element aerofoil.



(b) The ROBIN fuselage.

Figure 1. Signed Distance Function (SDF) contours of different 2D/3D geometries.

One can immediately realise that these contour lines could be ideal mesh layers for boundary layer resolution. These contours are shape conforming and follow a surface-norm growth, which could provide excellent gradient representations along wall normal directions. In practice, there have been attempts applying SDF to mesh generation (Refs. 5, 21, 22), but the application is not immediately straightforward. This is mostly due to convergence and divergence of SDF gradients. Near concave corners, the SDF gradient convergence results in ridge points, lines, and surfaces (or medial axes) in the SDF field, i.e. locations that have multiple closest points to the wall surface. Mesh elements would collide at these locations leading to intersections. The divergence of the SDF near convex

corners, on the other hand, leads to ever-growing size mismatches between elements as they move away from the wall surfaces. Special procedures for intersection detection, size check, mesh topology changes, or smoothing are inevitable for high-quality meshes. This adds further complexity and computational costs to the process. Previous studies on automated mesh generation have hence exploited SDF mainly to assist e.g. the blocking topology generation (Ref. 21) and guide mesh smoothing (Ref. 22).

However, these are not at all restrictions for mesh-free methods operating on points. There is no need for rigorous connectivity or topologies in point clouds, and colliding points can be simply removed or merged. For most mesh-free methods, what matters is to ensure enough points in critical regions to provide information. SDF can hence serve as a powerful tool for point cloud generation for mesh-free methods.

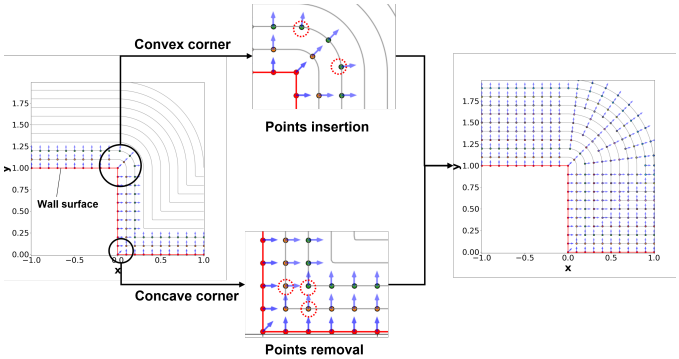


Figure 2. Demonstration of SDF-guided point cloud generation with point insertion and removal. The arrows represent local SDF gradients as layer advancing directions.

The new idea for point cloud generation, is to place points on the wall surface, i.e. the initial SDF wavefront, and let the points propagate with the wavefront until the near-body region is filled. Points may be removed when they collide near the SDF convergence regions, and new points may be added when the gaps expand too much. This is illustrated in Figure 2 for a step shape with convex and concave corners.

The method can be seen as a new advancing layer, point cloud generation method, but also shares similarities with the strand mesh approach. Compared to conventional advancing front point generation, the proposed method ensures surface-normal point distributions and provides layer structures in the near-body region, which is critical to fine boundary layer resolution. Compared to the strand mesh generation, this method is obviously more flexible, as there is no strict requirement for one-to-one matching between layers.

SDF-guided point cloud generation

The proposed cloud generation method grows points layer-by-layer from the wall surface, driven by the SDF and its gradients. The growth of one point layer require several inputs from the previous layer, including the previous layer points (the initial layer being the wall surface), their advancing directions, the growth law, and layer-dependent tolerance values to guide insertion, removal, and SDF enforcement. The process begins from advancing the previous layer (Step 1). The new layer is then examined for point insertion to eliminate large gaps (Step 2). The layer SDF will then be computed and enforced since there are newly inserted points (Step 3). Then, points will be checked for removal if they are too close (Step 4). Finally, the points will have their final coordinates, SDF values, and advancing directions confirmed and written, for the growth of the next layer (Step 5).

Point advancing

To propagate the points, the proposed method requires the computation of the SDF and its gradients. The SDF values are used to locate each point layer and correct the point positions, if necessary, while the gradients provide the propagation or advancing directions for each point from one layer to the next.

The computation of SDF values, or wall distance, is not at all a new problem in CFD. The computation has many established options ranging from brute-force approaches to PDE solutions (Ref. 23). The SDF computation during point cloud generation is not a wasted or extra effort, as the minimal wall-distance information is also required by many turbulence closures.

The computation of the SDF gradients is less common and requires some extra considerations. For simple shapes and their combinations, we can get the SDF gradients via analytical solutions. Yet for most non-trivial shapes, a more general approach is through finite differences, local polynomial approximations, or even machine learning regressions (Ref. 24), though the computation may be expensive.

The present work has adopted a simple and straightforward estimation of advancing directions. For a near-body point $\mathbf{p}$, its SDF value is first computed through brutal force, hence its nearest point $\mathbf{p}_s$ on the wall surface is also identified. The advancing direction $\mathbf{n}_p$ at point $\mathbf{p}$ is estimated as

$$\mathbf{n}_p = (\mathbf{p} - \mathbf{p}_s)/L_p, \tag{7}$$

where $L_p = |\mathbf{p} - \mathbf{p}_s|$ is the SDF value at $\mathbf{p}$. If multiple $\mathbf{p}_s$ are identified for $\mathbf{p}$, i.e. $\mathbf{p}$ is at a ridge position, the $\mathbf{n}_p$ will be set as $\mathbf{0}$. This will tag this specific point for later removal.

When it comes to wall surface gradient estimation, Equation (7) would become invalid since $L_p = 0$. A simple solution is to shift the SDF computation relative to a shrank geometry (i.e. a negative SDF level) for the wall surface gradient estimation. Alternatively, one can simply adopt surface normal vectors computed from CAD models. The computations of SDF and its gradients are not of negligible cost when the point amount increases, but it should be highlighted that this is limited in the near-body region, and such computations can be ideally parallelised.

A point $\mathbf{p}_j^{l-1}$ ($l-1$ is the layer index and j is the point index in this layer) in an existing layer is advanced altogether to a temporary layer $\mathbf{p}_j^{l*}$ following the SDF gradients:

$$\mathbf{p}_j^{l*} = \mathbf{p}_j^{l-1} + ds_l \mathbf{n}_j^{l-1}, \tag{8}$$

where ds_l is a prescribed growth or advancing distance. $\mathbf{n}_j^{l-1}$ is the advancing direction indicated by the SDF gradients as in Equation (7). The step of Equation (8) is very similar to the strand mesh approach (Ref. 4) where all points are advanced altogether, but the advancing vector is provided by the SDF gradients, rather than layer surface normal vectors. Another key difference is that, existing points in the temporary layer $l*$ can be removed, and additional points can be inserted where necessary. These are crucial features handling the collision of points near concave corners and over-expansions of points near convex edges. The next step is to evaluate the layer points to determine whether point removal or insertion are needed.

Point insertion method

Point insertion is needed when the gaps between adjacent points in a layer become too large. This is common for convex corners or any regions where SDF gradients are diverging. The insertion of points is only needed within each layer, since adjacent layers have fixed distributions corresponding to the SDF. Due to the lack of the connectivity information in each layer, the insertion of points needs some special considerations.

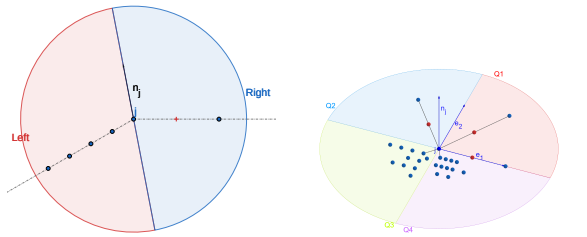
This work proposes a simple approach based on a special distance metric, i.e. the maximum point size S_{max}, to drive the point insertion. This metric finds the farthest distance in the neighbourhood of a central point amid irregular point distributions. It requires simple grouping of the points in advance, and then compares the nearest distance in each group.

There are many options of grouping points based on coordinates, distance, or clustering. In this work, the points are grouped into two groups in 2D, i.e. left and right relative the local advancing vector $\mathbf{n}_j$, as illustrated in Figure 3(a). In 3D, the points are grouped according to their quadrants after projected to a plane normal to $\mathbf{n}_j$, as shown in Figure 3(b). The projection is trivial as close points in the same layer are almost co-planar. These quadrants are determined via a local transformation to basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$. The choice of $\mathbf{e}_1$ is arbitrary and can be defined using the nearest point. $\mathbf{e}_2$ can then be determined as $\mathbf{n}_j \times \mathbf{e}_1$. In the present work, we have adopted four quadrants, but finer division is obviously possible.

The maximum point size S_{max} is then defined as the maximum value of the minimal distances in each group:

$$S_{max} = max(d_{min}^1, d_{min}^2, ..., d_{min}^k, ..., d_{min}^{n_g}), \tag{9}$$

where $d_{min}^k = min(dist(\mathbf{p}_j^{l*}, \{\mathbf{p}_k\}))$ is the minimal distance between point $\mathbf{p}_j^{l*}$ and its nearest neighbour in the group $\{\mathbf{p}_k\}$.



(a) Points divided into two groups in 2D. Red dots are newly inserted points.

(b) Points divided into four groups in 3D according to their local quadrants. Red dots are newly inserted points.

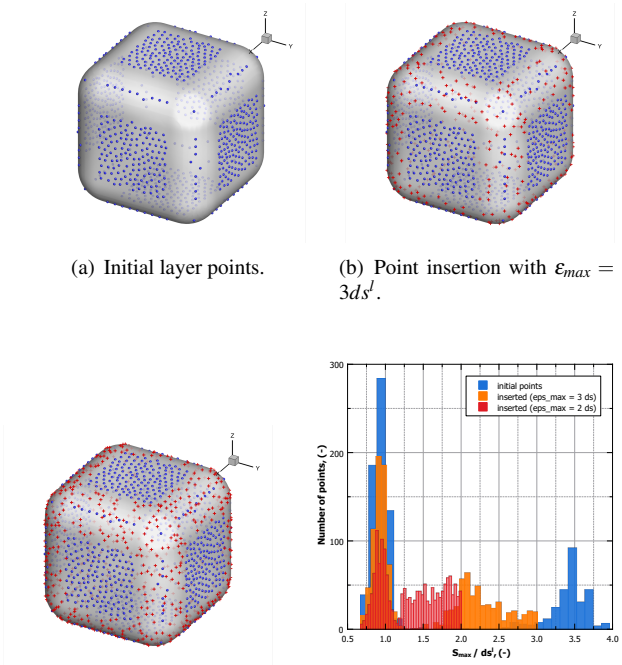
Figure 3. Layer points grouping in 2D and 3D for point insertion. $\mathbf{n}_j$ is the advancing vector at point j.

n_g is the total number of groups for points in the neighbourhood of $\mathbf{p}_j^{l*}$.

S_{max} is then compared to a tolerance ε_{max}^l, which could be a function of ds_l and the layer ID. If $S_{max} > \varepsilon_{max}^l$, a new point will be inserted in the middle between $\mathbf{p}_j^{l*}$ and the corresponding farthest point located by S_{max}. Then all nearby points, including the newly inserted point, will have their S_{max} updated and checked for insertion again. This process will be iterated until no point is identified for insertion in this layer.

The approach is illustrated in Figures4(a) to 4(d) for a cube shape after several layers of growth. It is obvious in Figure 4(a) that large gaps exist near the cube edges and vertices, because of the diverging SDF gradients. Figure 4(b) shows the point insertion with the criterion $\varepsilon_{max} = 3ds^l$. New points are denoted with red crosses. It is clear that insertion focused on the cube's edges and vertices, driven by the S_{max} metric. With a stricter criterion $\varepsilon_{max} = 2ds^l$, more points are inserted to fill the gap regions. The effectiveness is further quantified with the histogram of the S_{max} on this layer (Figure 4(d)). Before the insertion, the S_{max} clearly had two clusters, corresponding to the cube face points (low S_{max}) and the edge/vertex points (high S_{max}). After the insertion, the S_{max} distribution moved towards the lower-value region with an obvious cut-off at ε_{max}. These indicate that the insertion has effectiveness improved the point smoothness on this layer.

Note that this simple point insertion approach could further benefit from smoothing procedures, such as Laplacian or force equilibrium (Ref. 22), but point movements should be restricted to the newly inserted points and within the current SDF level. There are also more complex point insertion methods e.g. discrete hole searches (Ref. 25) and local triangulation. Another option is to maintain mesh connectivity within each layer, since initial surface meshes are available from e.g. STL files. This essentially converts the insertion/removal procedure into 1D or 2D mesh adaptation within each layer. Though handling such reduced-dimension adaptation seems easier than 3D mesh generation/adaptation, the management of connectivity information is not a trivial task and slightly violates the mesh-free flexibility. The current approach was adopted for its effectiveness in filling gaps and distance-based simplicity.



(a) Initial layer points.

(b) Point insertion with $\varepsilon_{max} = 3ds^l$.

(c) Point insertion with $\varepsilon_{max} = 2ds^l$.

(d) S_{max} histogram before and after the point insertion.

Figure 4. Demonstration of layer point insertion for a 3D cube. Dots are initial points and red '+' are inserted points.

Point removal method

The removal of points is straightforward and is decided upon the minimal point sizes S_{min}. In the temporary layer $l*$, a point is removed if $S_{min} < \varepsilon^l_{min}$, where ε^l_{min} is a tolerance that may depend on ds_l. This metric removes points that are too close. At ridge locations, since we have set the advancing vector to zero, only one earliest point will remain and all overlapping points in later layers will be removed.

For an arbitrary point j, its minimal point size for point removal is defined as:

$$S_{min} = min(dist(\mathbf{p}^{l*}_j, \{\mathbf{p}^l, \mathbf{p}^{l-1}\})), \tag{10}$$

where $\{\mathbf{p}^l, \mathbf{p}^{l-1}\}$ is the collection of the current and previous layers' points, excluding $\mathbf{p}^{l*}_j$ itself, and $dist()$ is the Euclidean distance. S_{min} is hence the distance between point j and its nearest neighbour, whether it is in the current layer or in previous layers. Previous layers are involved because the SDF gradients are not theoretically accurate and may cause overshoot near ridge locations, resulting in overlapping with points in previous layers. Once $S_{min} < \varepsilon^l_{min}$ is detected, point j is removed from the current layer (and will never appear in future layers). Then all nearby points in the current layer $l*$ have their S_{min} updated and are checked for removal again. This repeats until there is no point needs to be removed.

SDF corrections

Due to inaccuracy in the SDF gradient estimation and point insertion, points generated in the new layer $l*$ may not conform to the intended SDF level. A simple iterative correction procedure is hence introduced to enforce the SDF as:

$$(\mathbf{p}^{l*}_j)^{m+1} = (\mathbf{p}^{l*}_j)^m + C_s(\delta L_j)^m(\mathbf{n}^{l*}_j)^m, \tag{11}$$

where m is the iteration counter. $\delta L_j = L^l - L^{l*}_j$ is the difference between the intended SDF level L^l for layer l, and the actual SDF level L^{l*}_j of point j in this level. δL_j is also the trigger of this correction procedure in Equation (11), when δL_j is greater than a tolerance value ε_{Lj}. $\mathbf{n}^{l*}_j$ is the advancing vector at the current point j. $0 < C_s < 1$ is a relaxation factor for safer iterations. Equation (11) is a Newton-alike iteration, and it usually converges to $\delta L_j < \varepsilon_{Lj}$ after about 5 to 10 iterations.

After all insertion, removal, and SDF corrections are completed, the current layer l is ready for output. The final SDF values and advancing vectors will be computed as inputs for the next layer.

OFF-BODY CARTESIAN ADAPTIVE POINT CLOUD GENERATION

The proposed SDF-guided point cloud generation is capable of filling the computational domain around arbitrary shapes and topologies. However, this is unnecessary outside the boundary layer and near-wall regions. In off-body regions, without the need for boundary resolution, it is more suitable to use a uniform point distribution. In addition, adaptation is preferred to capture flow features e.g. vortices, shock waves, shear layers etc. Cartesian points are hence ideal choices given these considerations.

For point cloud generation purposes, this work uses the cell centres of Cartesian cells as point elements of the cloud. While refining, the parent point at $\mathbf{x}$ is deactivated and replaced by its children nodes (4 in 2D and 8 in 3D) as $\mathbf{x} \pm \mathbf{h}/\mathbf{2}$, where h is Cartesian size of this parent node. Coarsening is easily done by reversing this process. This choice maximises the similarities with classic Cartesian grid approaches for point cloud generation purposes. The hierarchical node information are also stored and handled via quad- and octrees in 2D and 3D, respectively. The cloud generation and adaptive refining/coarsening operations are hence highly efficient. This background Cartesian cloud also helps with locating points and finding neighbours.

For off-body point cloud generation, the computational domain is first filled with a coarse and uniform Cartesian cloud. The Cartesian points are then recursively refined to track down the outer layer of the near-body cloud, as well as any locations of special interest. This is done through classic Cartesian trees. The Cartesian points stop refining once the target point is within its local size h. The Cartesian points are then balanced, to ensure that no neighbouring points have a level difference greater than 2. A blanking metric is then necessary to deactivate points that fall within the near-body region, and

this is readily achieved through the SDF. The near-/off-body clouds are then blended together for the mesh-free CFD.

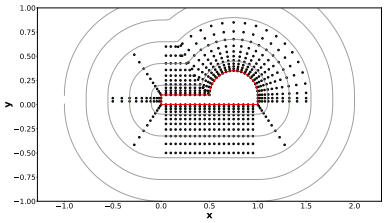
There are many options for interfacing between the near- and off-body clouds. Point distributions are not ideally smooth near the transition from near- to off-body, despite efforts to match the near-body distributions. A straightforward approach is to construct differential operators without distinguishing whether the points are near- or off-body. This is feasible given that mesh-free approaches have strong tolerance for irregular points, especially when equipped with the weight-balancing (Ref. 17). This, however, may still cause convergence or accuracy difficulties, if there are irregularities that strongly violate the physics. Another approach is to exchange values via interpolation and halo points. This is similar to the overset approach for mesh-based methods. The benefits are that the regular cloud distributions can be maintained for respective near- and off-body regions, and it adds another layer of flexibility to handle domain motions and deformations. The drawback is of course associated with the interpolation errors. These are topics that need further evaluations in future work. In the present work, the interfacing was done via direct mesh-free reconstruction.

DEMONSTRATION AND EVALUATION OF NEAR-BODY POINT CLOUD GENERATION

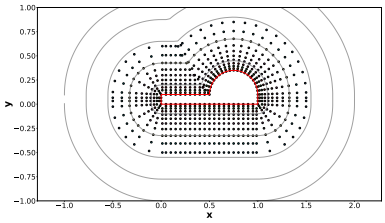
This section demonstrates the proposed point cloud generation with various 2D and 3D shapes. These shapes have simple definitions and representative features that are challenging for mesh generation. Point cloud generation is first demonstrated with a 2D shape, as shown in Figures 5(a) and 5(b). This is a simple yet representative 2D shape combining a rectangle and a semicircle. It has both convex and concave corners, as well as smooth and sharp geometry transitions, and is a good assessment of the proposed point cloud generation approach. In Figures 5(a) and 5(b), points of the same colour are of the same level, and the contour lines represent the SDF field.

Figure 5(a) adopted only the removal mechanism. Points in adjacent layers displayed clear coherent structures thanks to the layer-by-layer advancing. It is clearly shown here that the removal mechanism effectively avoided point collision near the concave corner. However, the gap near each convex corner grew larger at further layers. Nevertheless, these are eliminated when the insertion was introduced in Figure 5(b). The gaps were filled with new points, and smooth distributions are ensured.

For 3D demonstrations, this work used a cube shape with a concave corner as shown Figure 6(a). This is a geometry with very simple definitions, yet its mesh generation is not a trivial task. This shape provides a good mixture of sharp convex/concave corners and edges, and is a good case for the proposed point cloud generation in 3D. Figure 6(a) also shows the input surface points (STL files). Figures 6(b) to 6(c) present the 2nd and 5th layers of points. The transparent surfaces are the corresponding SDF iso-surfaces, and the arrows denote



(a) Point cloud generation with removal mechanisms only.



(b) Point cloud generation with both removal and insertion mechanisms.

Figure 5. Demonstration of SDF-guided point cloud generation for a simple 2D shape. Points of the same layer have the same colour. Contour lines denote the SDF field.

the advancing vectors at the points. It is clear that the points were conformal to the SDF iso-surfaces. Figure 6(d) provides a slice through the concave corner, which clearly shows the point removal in the concave corner and the point insertion for the convex corners. Most points between layers also displayed good coherent structures thanks to the layer advancing.

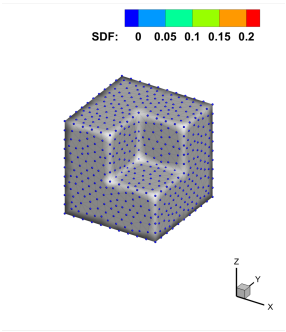
MESH-FREE CFD RESULTS

This section presents CFD results, based on clouds generated from the proposed framework. These provide important validation results for the efficacy of the proposed point generation for mesh-free modelling, with emphasis on boundary layer resolutions.

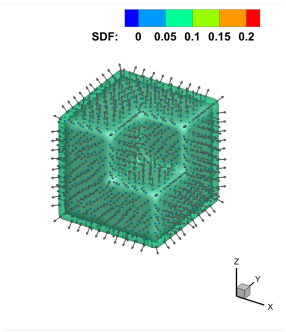
Transonic Flow past Biplane NACA0012 Sections

This subsection presents simulation results of transonic flows past two staggered, biplane NACA0012 aerofoils. The present work focused on two cases as detailed in Table 1. For both cases, the first aerofoil's leading edge was placed at the origin. The first case D1 was a laminar flow case at $Re = 500$, $Ma_\infty = 0.8$, and $AoA = 10^\circ$, as shown in Figure 7(a). In this case, it is known that (Ref. 26) the upper surface of the upper aerofoil would experience strong separation, because of the flow angles and blockage of the lower aerofoil. The second case D2 was a transonic turbulent case with $Re = 9 \times 10^6$, $Ma_\infty = 0.7$, and $AoA = 1.49^\circ$, as shown in Figure 8(a). In this case, two shock waves would form: one above the upper aerofoil and the other between the two aerofoils.

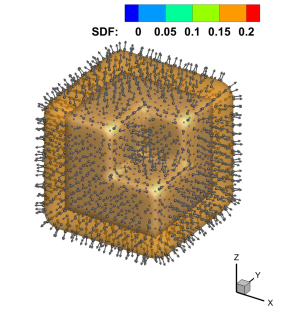
For both cases, despite the simple geometry, mesh-based modelling would still turn to overset or unstructured approaches



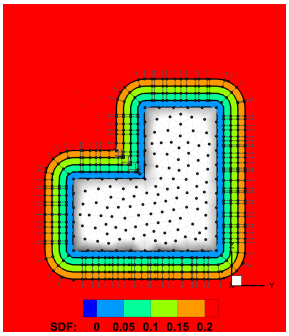
(a) Cube shape with a concave corner.



(b) Points distribution on layer 2.



(c) Points distribution on layer 5.



(d) Points distributions on a 2D slice through the concave corner. The contour lines denote the SDF field.

Figure 6. Point cloud generation for a cube with a concave corner.

because of the disconnected bodies. The meshing process also needs careful consideration for the complex flow physics. It is hence of interest to see if the fully automated mesh-free modelling workflow can handle the geometric and physical complexities. Moreover, for case D2, this section will also demonstrate the proposed framework's flexibility in adaptive modelling in the off-body region exploiting the Cartesian points.

Table 1. Biplane aerofoils test cases and flow conditions.

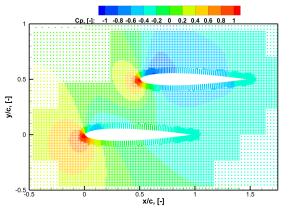
Case	Second aerofoil offset $(\delta x, \delta y)$	AoA(deg)	Ma_∞	Re	Off-body Adaptation
D1	(0.5C, 0.5C)	10	0.8	500(Laminar)	N
D2	(0.5C, -0.5C)	1.49	0.7	9×10^6(RANS)	Y

For near-body cloud generation of both cases, there were 248 points on each of the aerofoil surfaces. It should be highlighted that the two aerofoils were recognised as a single initial layer with the SDF value $L = 0$, despite their disconnected topology. The near-body cloud grew 20 layers following the gradients until SDF $L = 0.05C$, where C is aerofoil chord length. The near-body growth followed an exponential growth with first layer height corresponding to $y^+ = 1$ for RANS simulations. In this case, it was unnecessary to fill the entire domain with near-body points, because of the large distance between the two aerofoils.

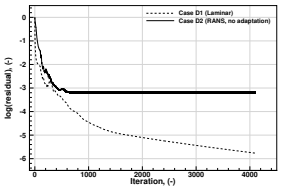
The computational domain was then filled with Cartesian points with successive refinements around the wall boundaries. The Cartesian points matched the outer layer of the near-body points, and all points with $L < 0.05C$ were deactivated. For the laminar case D1, there were in total 28,156 points in the computational domain. For the RANS case D2, the initial point number was 28,900, but later adaptation gradually increased the off-body points.

Flow solutions of case D1 are shown in Figure 7(a) with the computational point coloured by pressure coefficients. With the SDF-guided near-body points and the Cartesian off-body points, the mesh-free modelling offered smooth pressure solutions, especially the low pressure region due to flow separation on the upper surface of the upper aerofoil. Figure 7(b) shows the L2 norm convergence history of the residual for both cases D1 and D2 without adaptation. The laminar case D1 showed smooth convergence to the order of -6 thanks to the high viscosity, while the RANS case D2 stabilised around the order of -3 with minor oscillations. Note that both simulations were assumed steady, but the shock waves in case D2 induced small unsteadiness during the modelling.

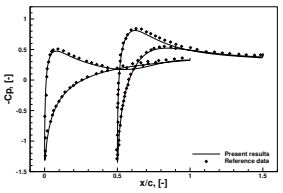
For more quantitative validation, Figure 7(c) presents the surface pressure coefficient distributions, while Figure 7(d) presents the surface skin friction coefficient distributions. Comparisons were made against reference modelling data using mesh-based finite volume methods (Ref. 26). The pressure and skin friction results on both aerofoils have shown excellent agreement between the current mesh-free modelling and mesh-based FVM results. These results establish confidence of the current method for complex flow problems, and highlight its ability in accurately handling boundary layer flows.



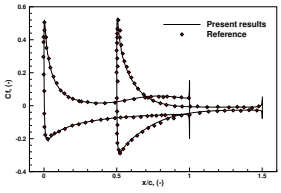
(a) Mesh-free flow solutions.



(b) Flow convergence of D1 and D2(baseline).



(c) Surface Cp comparisons. Reference data are mesh-based FVM simulation results (Ref. 26).

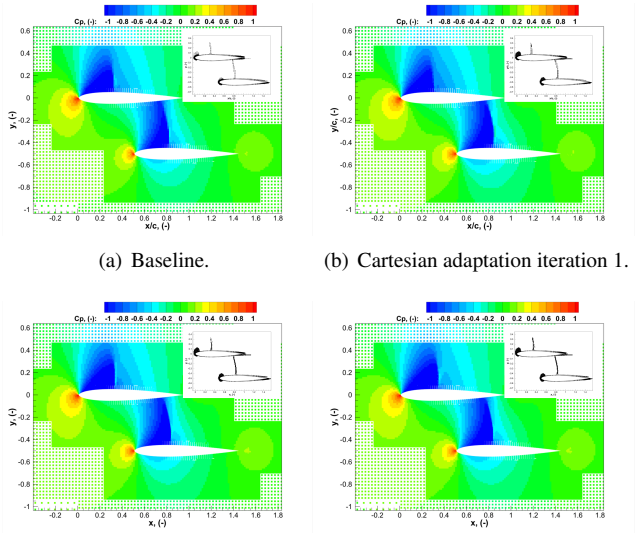


(d) Mesh-free flow solutions.

Figure 7. Case D1 of Table 1, laminar flow past the double foils at Ma = 0.8.

For case D2, the point adaptation was driven by density gradients to improve shock resolution. During adaptation, all Cartesian points that have dimensionless density gradients greater than 2.0 were identified, and their Cartesian levels were increased by one. All children nodes inherited the same

flow values from the parent node. Adaptive flow solutions of case D2 are shown in Figures 8(a) to 8(d). These are again computational point clouds coloured with pressure coefficient solutions. The solutions presented excellent shock wave resolutions above and between the two aerofoils. Figure 8(a) shows the baseline solution without adaptation. The diagram in the top right corner illustrated the identified points for adaptation via density gradients. These points closely followed the two shock waves. Later adaptive modelling from Figures 8(b) to 8(d) repeated the same procedures, and the total point numbers gradually increased from 28,900 to 30,577, 31,366, and 31,420. After 3 iterations, the shock resolutions were much sharper as shown in Figure 8(d).



(a) Baseline. (b) Cartesian adaptation iteration 1.

(c) Cartesian adaptation iteration 2. (d) Cartesian adaptation iteration 3.

Figure 8. Case D2 of Table 1, adaptive RANS modelling of flow past the double foils at Ma = 0.7.

More quantitative comparisons in terms of surface pressure distributions are shown in Figure 9. Comparisons with reference mesh-based modelling (Ref. 26) again showed excellent agreement. For the baseline solution, there were small discrepancies near both shock waves. The adaptation gradually improved the shock resolutions and drove the solutions towards the mesh-based reference results. Regardless, small discrepancies remained, especially for the second between the two aerofoils. The solution could be improved by involving near-body adaptation as demonstrated in (Ref. 17), but the current work only employed adaptation in the off-body Cartesian regions for simplicity. The near-body regions adaptation is compatible with the point insertion mechanisms, but the detailed implementation requires future investigations.

ROBIN Fuselage

For 3D validation, the presented work adopted the classic ROBIN helicopter fuselage model (Refs. 27, 28) with the pylon installed. This is a classic aerodynamic shape with analytical definitions using super-ellipses. The fuselage alone is a

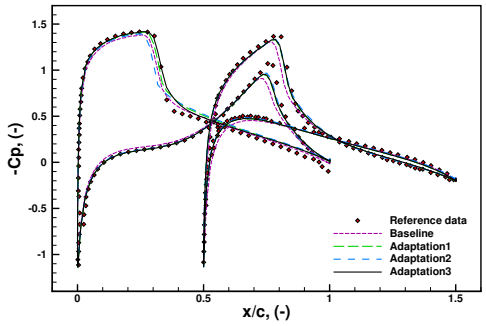


Figure 9. Surface Cp comparisons. Reference data are mesh-based FVM simulation results (Ref. 26).

simple and smooth shape, but its combination with the small pylon creates concave features that requires attention when meshing. For the mesh-free modelling here, the near-body point cloud was generated using the proposed SDF-guided approach, while the off-body points were generated using Cartesian points in 3D.

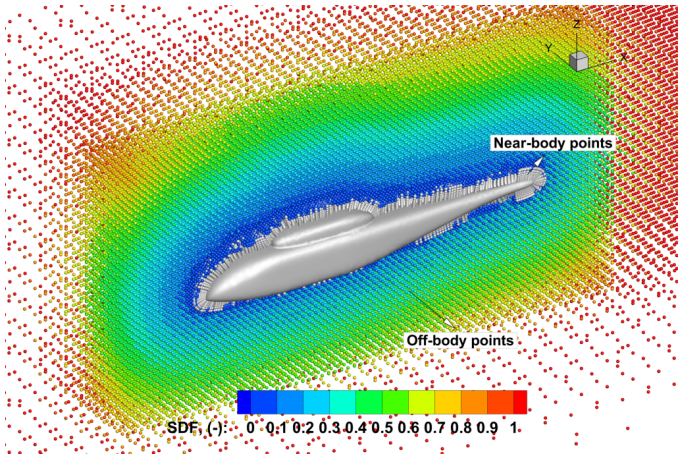


Figure 10. Point cloud generation around the ROBIN geometry. White points are near-body points, and points coloured by SDF values are off-body Cartesian points.

Figure 10 presents point cloud generation around the robin geometry. The input was a surface PLY (Polygon File Format) file with 4,877 points with smooth distributions. In total 17 layers were grown following the SDF in the near-body region untill SDF $L = 0.05$ (the fuselage has a non-dimensional length of 2.0 in the simulations). The off-body Cartesian points were successively coarsened towards the far-field. All Cartesian transitions were smoothed to ensure the change ratio never exceeds 2.0 in all directions. Points within the near-body region were deactivated for the modelling. In total, there were 284,905 points in the computational domain, and most points were placed near the wall surfaces.

The simulations were carried out at $Ma = 0.2$ and $Re = 4.6 \times 10^6$ using steady RANS modelling with the SA model. The modelling was advanced using implicit dual-time stepping with a CFL number of 20.0. Figure 11 presents the pres-

sure solutions on the fuselage surface along with the surface point distributions. Note that the surface solution was interpolated from the computational points for visualisation purposes. The modelling clearly captured the stagnation at the fuselage nose and the pylon.

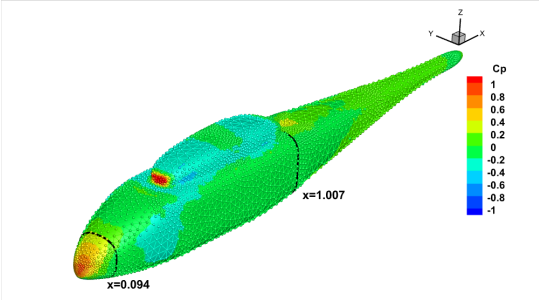
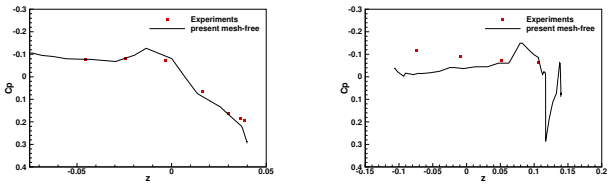


Figure 11. Surface Cp distributions.

For more quantitative comparisons, Figures 12(a) and 12(b) present the pressure distributions on the slices x=0.094 and x=1.007 as denoted in Figure 11, respectively, with experimental validation. At the slice x=0.094, the modelling results showed excellent agreement with the experimental data. The discrepancies at x=1.007 were larger. This slice was place at the back of the pylon, where intricate flow details present and the solution usually depends on the turbulence model used. Overall, this case demonstrated the efficacy of the proposed point generation for 3D applications. The modelling resolution could benefit from improved point density and adaptation.



(a) Cp distributions at x = 0.094. (b) Cp distributions at x = 1.007.

Figure 12. Surface pressure comparisons with experiments (Ref. 27).

Caradonna-Tung Rotor in hover

The point cloud generation and mesh-free CFD framework was also demonstrated using the classic hovering Caradonna-Tung rotor case (Ref. 29). The flow was solved in a non-inertial rotating reference frame. This adds a source term to the RHS of the governing equations, but has minor impact on the mesh-free discretisation, highlighting the mesh-free scheme's strong compatibility with conventional models. The near-body point clouds were generated via the SDF-guided approach until 0.1 C from the wall surfaces (C is the blade chord length). The off-body region was filled with Cartesian points with point distributions matching the near-body clouds. The off-body region was successively coarsened further away from the blade surfaces.

RANS simulations were carried at a tip Mach number of 0.612 and a collective pitch of 8 degrees. The Reynolds number was 2.6 million based on the tip speed and the blade chord length. The simulations adopted the S-A turbulence closure. The modelling employed adaptation in the off-body region, exploiting the Cartesian points. Regions near the rotor wake was gradually refined according to the local Q-criterion. The final point clouds had about 0.75 million points in total in the computational domain. This was still a coarse point cloud for demonstration and validation purposes.

The resolved flow solutions are presented in Figure 13. These are filtered computational points that have higher Q-criterion values to highlight the rotor wake. It is clearly shown that the adaptive mesh-free modelling managed to capture clear and strong wake sheets and tip vortices, despite the coarse point clouds. The current modelling maintained the tip vortices for about 200 degrees along the azimuth. Finer point clouds and continued adaptation can of course improve the resolution.

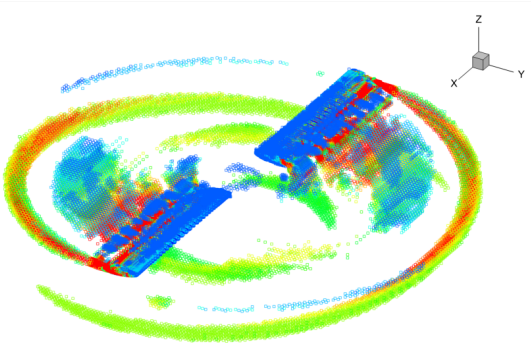


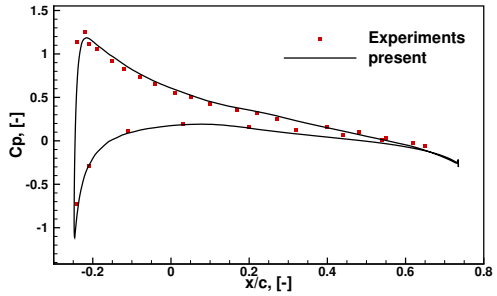
Figure 13. Mesh-free flow solutions of the Caradonna-Tung rotor in hover. These are points with dimensionless Q-criterion $>$ 0.0005, coloured with turbulence viscosity.

For more quantitative comparisons, blade surface pressure coefficients were extracted and presented in Figures 14(a) and 14(b). These are pressure distributions at 0.89 and 0.96 spanwise stations, respectively. The mesh-free modelling showed excellent agreement with the experimental measurements at both stations, highlighting the modelling accuracy.

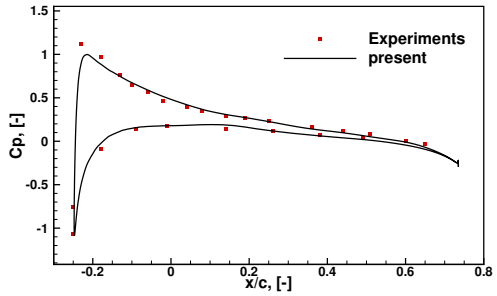
CONCLUSIONS

This work presented a novel point cloud generation framework for mesh-free CFD modelling. The development focused on boundary layer resolution, automation, and adaptation. We demonstrated the proposed approaches with various 2D and 3D shapes with complex geometric and topological features. We also demonstrated mesh-free CFD results using the proposed point cloud generation.

In the near-body region, we introduced the Signed Distance Function to guide point generation layer-by-layer. This was shown to facilitate boundary conforming and surface normal point distributions amid highly complex geometries and



(a) Station $y/b = 0.89$.



(b) Station $y/b = 0.96$.

Figure 14. Blade surface pressure comparisons with experiments (Ref. 29).

topologies. The proposed point removal and insertion mechanisms were highly effective in handling concave/convex features, as well as drastic changes in SDF topologies.

In the off-body region, we adopted Cartesian cell centres as computational points for the mesh-free modelling. The Cartesian points demonstrated great flexibility to interface with arbitrary near-body points, and to provide uniform point distributions. Their excellent adaptivity and efficient tree implementations also enabled easy implementation of adaptive mesh-free CFD modelling.

The modelling of various 2D and 3D benchmark cases, with various complex geometric and physical complexities, further demonstrated the suitability of the proposed point generation for mesh-free CFD. The modelling results showed excellent agreement between reference results, amid critical flow physics e.g. flow separation and shock waves. These highlighted the excellent boundary layer resolutions of the modelling thanks to the SDF-guided point cloud generation in the near-body region. Off-body adaptive mesh-free modelling for shock capturing was also showcased exploiting the Cartesian points.

The proposed point generation still needs input of a few critical and case-dependent parameters, e.g. point insertion/removal tolerances, initial surface point distributions, and point growth rates. These may deteriorate the complete automation of the entire modelling workflow. Future work will aim to reduce these parameters and correlate them with geometry or flow characteristics. Future work will also focus on quantifying the computational complexity and parallel implementation of the proposed point generation. Investigations of improved point insertion/removal algorithms and point smoothing methods are also planned.

Author contact: Tao Zhang tz77@leicester.ac.uk George Barakos george.barakos@glasgow.ac.uk

ACKNOWLEDGMENTS

The authors would like to acknowledge the Sulis Tier 2 HPC platform hosted by the Scientific Computing Research Technology Platform at the University of Warwick, funded by EPSRC Grant EP/T022108/1 and the HPC Midlands+ consortium. The authors also wish to acknowledge the support of the U.K. Vertical Lift Network (EPSRC Grant EP/M018164/1).

REFERENCES

1. Slotnick, J. P., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D. J., "CFD vision 2030 study: a path to revolutionary computational aerosciences," Tech. rep., 2014.

2. Chawner, J. R. and Taylor, N. J., "Progress in geometry modeling and mesh generation toward the CFD vision 2030," *AIAA Aviation 2019 Forum*, 2019, p. 2945.

3. Baker, T. J., "Mesh generation: Art or science?" *Progress in aerospace sciences*, Vol. 41, No. 1, 2005, pp. 29–63.

4. Sitaraman, J., Lakshminarayan, V. K., Roget, B., and Wissink, A. M., "Progress in strand mesh generation and domain connectivity for dual-mesh CFD simulations," *55th AIAA Aerospace Sciences Meeting*, 2017, p. 0288.

5. Roget, B., Sitaraman, J., Lakshminarayan, V., and Wissink, A., "Prismatic mesh generation using minimum distance fields," *Computers & Fluids*, Vol. 200, 2020, pp. 104429.

6. Gingold, R. A. and Monaghan, J. J., "Smoothed particle hydrodynamics: theory and application to non-spherical stars," *Monthly notices of the royal astronomical society*, Vol. 181, No. 3, 1977, pp. 375–389.

7. Lucy, L. B., "A numerical approach to the testing of the fission hypothesis," *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, Vol. 82, 1977, pp. 1013–1024.

8. Shu, C., Ding, H., Chen, H., and Wang, T., "An upwind local RBF-DQ method for simulation of inviscid compressible flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 18-20, 2005, pp. 2001–2017.

9. Belytschko, T., Lu, Y. Y., and Gu, L., "Element-free Galerkin methods," *International Journal for Numerical Methods in Engineering*, Vol. 37, No. 2, 1994, pp. 229–256.

10. Suchde, P., Jacquemin, T., and Davydov, O., "Point cloud generation for meshfree methods: An overview," *Archives of Computational Methods in Engineering*, Vol. 30, No. 2, 2023, pp. 889–915.

11. Löhner, R. and Onate, E., "An advancing front point generation technique," *Communications in numerical methods in engineering*, Vol. 14, No. 12, 1998, pp. 1097–1108.

12. Löhner, R. and Oñate, E., "A general advancing front technique for filling space with arbitrary objects," *International journal for numerical methods in engineering*, Vol. 61, No. 12, 2004, pp. 1977–1991.

13. Pahar, G. and Dhar, A., "A robust volume conservative divergence-free ISPH framework for free-surface flow problems," *Advances in water resources*, Vol. 96, 2016, pp. 423–437.

14. Shimada, K., *Physically-based mesh generation: automated triangulation of surfaces and volumes via bubble packing*, Ph.D. thesis, Massachusetts Institute of Technology, 1993.

15. Oñate, E., Idelsohn, S., Zienkiewicz, O., and Taylor, R., "A finite point method in computational mechanics. Applications to convective transport and fluid flow," *International journal for numerical methods in engineering*, Vol. 39, No. 22, 1996, pp. 3839–3866.

16. Kwan-yu Chiu, E., Wang, Q., Hu, R., and Jameson, A., "A conservative mesh-free scheme and generalized framework for conservation laws," *SIAM Journal on Scientific Computing*, Vol. 34, No. 6, 2012, pp. A2896–A2916.

17. Zhang, T. and Barakos, G. N., "Assessment of implicit adaptive mesh-free CFD modelling," *International Journal for Numerical Methods in Fluids*, 2024.

18. Osher, S. and Solomon, F., "Upwind difference schemes for hyperbolic systems of conservation laws," *Mathematics of computation*, Vol. 38, No. 158, 1982, pp. 339–374.

19. Mavriplis, D. J., "Unstructured-mesh discretizations and solvers for computational aerodynamics," *AIAA journal*, Vol. 46, No. 6, 2008, pp. 1281–1298.

20. Spalart, P. and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th aerospace sciences meeting and exhibit*, 1992, p. 439.

21. Xia, H., Tucker, P., and Dawes, W., "Level sets for CFD in aerospace engineering," *Progress in Aerospace Sciences*, Vol. 46, No. 7, 2010, pp. 274–283.

22. Persson, P.-O. and Strang, G., "A simple mesh generator in MATLAB," *SIAM review*, Vol. 46, No. 2, 2004, pp. 329–345.

23. Tucker, P., "Differential equation-based wall distance computation for DES and RANS," *Journal of computational physics*, Vol. 190, No. 1, 2003, pp. 229–248.

24. Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S., "Deepsdf: Learning continuous signed distance functions for shape representation," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

25. Seidel, T., König, C., Schäfer, M., Ostermann, I., Biedert, T., and Hietel, D., "Intuitive visualization of transient groundwater flow," *Computers & geosciences*, Vol. 67, 2014, pp. 173–179.

26. Kennett, D., *On the development of a meshless method to study multibody systems using computational fluid dynamics*, Ph.D. thesis, University of Liverpool, 2013.

27. Freeman, C. E. and Mineck, R. E., "Fuselage surface pressure measurements of a helicopter wind-tunnel model with a 3.15-meter diameter single rotor," Tech. rep., NASA / TM-80051, 1979.

28. Woodgate, M. A. and Barakos, G. N., "An implicit hybrid method for the computation of rotorcraft flows," *The Aeronautical Journal*, Vol. 122, No. 1256, 2018, pp. 1522–1556.

29. Caradonna, F. X. and Tung, C., "Experimental and analytical studies of a model helicopter rotor in hover," *European rotorcraft and powered lift aircraft forum*, No. A-8332, 1981.

CLASSIFICATION & COPYRIGHT

The paper must be unclassified for release to the public and cleared by the appropriate company and/or government agency if necessary. CEAS and its national member societies shall be allowed to publish the paper. The copyright information is stat-ed on the Forum website https://www.rotorcraft-forum.eu/ and on the front page of this template. All papers presented at the ERF will be published as ERF proceedings. In addition, they will be available in a freely accessible web-based repository 2 years after the respective conference.