Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona,Spain, July 9-13, 2018 **ICCFD10-2018-0331**

Educing coherent flow structures from unstructured meshes

J. I. Cardesa*, O. Lehmkuhl**,D. Mira**, G. Houzeaux**
Corresponding author: ji.cardesa@upm.es

* ETSI Aeronáuticos, Universidad Politécnica de Madrid, Madrid, Spain.
** BSC-CNS, Barcelona, Spain.

Abstract: We present a method to time-track regions of a flow. A flow region, or structure, is defined as a collection of spatially-connected grid points which share a user-defined property of interest. The technique, which originated in time-resolved direct numerical simulations of turbulent flows on structured grids [1], is adapted in the present study to unstructured meshes. We use a finite element simulation [2] of a sphere's wake on an unstructured tetrahedron grid as the basis to develop the code that educes flow structures and links them with those in temporally-adjacent fields. We validate the extraction process against the original code for structured grids by using an identical input field for both codes. Finally, we observe the linking of structures between two flow fields deteriorates as expected when we increase the temporal distance between the fields, providing confidence in the performance of our tracking algorithm.

Keywords: Computational Fluid Dynamics, Data Post-processing, Structure Dynamics, Coherent Structures, Finite Elements.

1 Introduction

The numerical simulation of turbulent flows initiated a new era in the study of fluid motion. Initially, its use was limited to the computation of basic mean quantities. With time, however, the spacial structure of static flow features came within reach of visualization tools, while more recently it has become ordinary to see movies of flow simulations. Yet extracting and interpreting the wealth of information inherent to these movies remains a challenging task.

In recent studies of fundamental turbulence research, a technique to tackle this specific problem has been developed. By tracking individual flow structures and organizing their complex spatio-temporal evolution into graphs, it is now possible to carry out a statistical analysis over dynamical events in the flow. The technique was initiated on data sets from temporally-resolved Direct Numerical Simulations (DNS), where high-order schemes based on structured meshes are used to simulate flows in canonical scenarios such as triply periodic domains or plane channel flows. This approach has enhanced our possibility to uncover dynamical patterns within the flow which would otherwise be mere speculation based on eye inspection of movies from flow simulations [1, 3]. Yet the vast majority of flow simulations these days, particularly those of industrial relevance, are carried out on unstructured meshes and complex flow geometries. Extending the code from structured to unstructured meshes requires solutions which are the object of the present study.

2 Sphere wake simulation

The wake of a subcritical sphere [4] was computed using ALYA, a multi-physics finite element (FE) code developed at BSC [2]. A finite element Galerkin approximation is used for the space discretization with a non- incremental fractional step method to stabilize pressure. This allows for the use of finite element pairs that do not satisfy the inf-sup condition, such as equal order interpolation for the velocity and pressure used in this work. The convective term is discretized using the EMAC discretization [5] that allows to conserve kinetic energy, momentum and angular momentum. Temporal discretization is performed through an explicit conservative third-order Runge-Kutta scheme [6] combined with an eigenvalue based time step estimator [7]. The formulation is closed by an appropriate expression for the subgrid-scale viscosity. In this work, the eddy-viscosity model proposed by Vreman is used [8].

The simulation was done with a grid made of $\mathcal{N}_{els} = 9625923$ elements, all of which were tetrahedral, and a total of $\mathcal{N}_{nodes} = 1691060$ nodes. The streamwise, spanwise and vertical velocities, u, v, w, respectively, were stored for $N = 100$ consecutive time points, each one spaced by 10 numerical time steps t_s from the previous one.

3 Problem Statement

The starting point for the testing of structure tracking is a sequence of N flow fields with a time gap Δt between them that is sufficiently small. Typically, we are interested in a specific scalar quantity θ that is represented on the grid of the simulation (x_i, y_i, z_i) at the discrete times t_n, where $1 \leqslant n \leqslant N$. For example, we may want to explore the dynamics of flow regions where the temperature θ is above the melting point θ_m of the boundaries. Our process to accomplish this study involves three steps.

Step 1 For each t_n, identify those regions of the flow where $\theta > \theta_m$, which we call interchangeably structures or objects. This is done with a connected-component labeling algorithm [9]. Each object j at t_n will contain a set of grid points, expressed as the voxel list $V_j(t_n)$. Going from the discrete flow fields of $\theta(x_i, y_i, z_i, t_n)$ to as many voxel lists $V_j(t_n)$ as there are objects in the sequence of fields is the goal of the structure extraction process. It can be carried out one field at a time. We also store $m(t_n)$, the number of objects at t_n.

Step 2 For each object j at t_n with voxel list $V_j(t_n)$, find the objects (if any) from lists $V_{1 \leqslant j \leqslant m(t_{n+1})}(t_{n+1})$ which intersect with $V_j(t_n)$ and store the intersection volume of each such *connection*. This linking process provides the list of objects at t_{n+1} with which each object at t_n is connected, and what the intersection volume for each connection is. It requires taking always two consecutive time points at once and produces $N - 1$ link files.

Step 3 Load the $N - 1$ link files into memory and deduce the dynamics of each individual object from its birth to its death during the interval $t_1 \leqslant t_n \leqslant t_N$. Organize the spatio-temporal interactions between different objects into branches belonging to graphs, as explained in [1].

Step 3 will not be discussed in the present study, as it is identical for structured and non-structured meshes, so that it represents no further difficulty with respect to the existing code that we started working with. Step 1 is where most changes need to be introduced, and we start by outlining those.

3.1 Structure extraction

In our flow simulation based on finite elements, the grid information consists of the following items:

- $\mathcal{N}_{els}$: number of elements
- $\mathcal{N}_{nodes}$: number of nodes
- x_i, y_i, z_i: position of the nodes along the coordinate axes, where $1 \leqslant i \leqslant \mathcal{N}_{nodes}$
- $\mathcal{M}_i$: number of nodes in element i, where $1 \leqslant i \leqslant \mathcal{N}_{els}$
- $\mathcal{A}_{i,j}$: node reference number of *j-th* node delimiting element i. $1 \leqslant i \leqslant \mathcal{N}_{els}$ and $1 \leqslant j \leqslant \mathcal{M}_i$.

With the above items, everything about the grid geometry is known. However, structure extraction will involve applying a connected-component labeling algorithm [9], which requires knowing at each grid point who the neighbors are. The neighbors will be probed one after the other in order to check the $\theta(x_i, y_i, z_i) > \theta_m$ condition. Whereas on a 3D structured grid the 26 neighbors to grid point (x_i, y_i, z_i) are immediately obvious by looping each coordinate from $i = -1$ to $i = 1$, from the above grid information we need to first deduce the neighbor list for each grid point. Moreover, we need to decide weather to take as grid point definition the center of an element or the element's nodes. Since FE approximate the solution of the governing equations at the nodes, we retained the nodes, rather than the element centers, as the building blocks of our tracking code - more on this in section 3.2. Taking as definition that *two nodes are considered neighbors if they share an element*, we deduce the following grid properties:

- $\mathcal{L}_i$: number of nodes neighboring node i, where $1 \leqslant i \leqslant \mathcal{N}_{nodes}$
- $\mathcal{P}_{i,j}$: node reference number of j-*th* node which is a neighbor of node i. $1 \leqslant i \leqslant \mathcal{N}_{nodes}$ and $1 \leqslant j \leqslant \mathcal{L}_i$.

All the required information to infer $\mathcal{L}_i$ and $\mathcal{P}_{i,j}$ is contained in $\mathcal{M}_i$ and $\mathcal{A}_{i,j}$. We note in passing that since our grid was made of purely tetrahedral elements, $\mathcal{M}_i = 4$ except at the boundaries where $\mathcal{M}_i = 3$. However, we kept the code open to more complex grids and retained the general form of $\mathcal{M}_i$ throughout.

Our definition of neighboring nodes ensures that by iteratively looping through the neighbors of the neighboring nodes, eventually the entire domain is probed. This condition is necessary, since without it a single object could mistakenly be identified as two differently-labeled yet spatially-contiguous objects. Our definition does allow, however, for two nodes A and B to be neighbors, yet A is closer to some node C than to B without A and C being considered neighbors (for lack of A and C being on a common element). This does not pose any issues from the labeling point of view, since other neighbors of node A will be linked to C through common elements and hence C can be reached from A provided the intermediate nodes comply to the $\theta(x_i, y_i, z_i) > \theta_m$ criterion.

In order to test the code for object extraction in an unstructured mesh, we generated $\mathcal{L}_i$ and $\mathcal{P}_{i,j}$ for a cubic structured grid with N_C grid points in each direction, where $N_C = 512$. In such a rectangular mesh, $\mathcal{L}_i$ can take values 7, 11 or 17 if node i is at a cube vertex, edge or face, respectively, and 26 otherwise. We stored a velocity component u computed on the N_C^3 grid with a triply-periodic, pseudo-spectral DNS [10]. Since the code to be tested requires the input scalar field to be a 1D array, we reshaped the velocity field $u(i, j, k)$, where all three indices span the range 1 to 512, to array $u(i)$ where $1 \leqslant i \leqslant 512^3$. The object extraction routine for unstructured grids uses a connected-component labeling routine written around arrays $u(i)$, $\mathcal{L}_i$ and $\mathcal{P}_{i,j}$, with parameter $\mathcal{N}_{nodes}$ set to 512^3. In the routine for structured grids, the code takes in $u(i, j, k)$ and N_C only. Both routines search for objects matching the condition $u > u_{threshold}$. For various values of $u_{threshold}$, both routines identified the same objects. To increase the geometrical complexity of the objects to be identified, we applied both routines on fields of u^2 and u^3 probing different thresholds with identical results. Finally, we found the objects of both our codes were the same as those identified by function `bwlabeln` from MATLAB® when applied to array $u(i, j, k)$ after binarizing it with the corresponding threshold. These findings validate our new code for educing structures.

3.2 Temporal linking

Step 2 outlined in section 3 involves a comparison of node lists V_j at times t_n and t_{n+1} to compute the intersection volumes (if any) between objects at two points in time. Detecting the intersection is straightforward, since it only requires a node-by-node comparison to find a shared node. Computing the intersection volume required assumptions. Whereas the volume of an element is immediately determined from the coordinates of its delimiting nodes, the volume that a node is representative of needs to be computed somehow. From x_i, y_i, z_i, $\mathcal{M}_i$ and $\mathcal{A}_{i,j}$, we compute the following grid properties:

- $\mathcal{V}_j$: volume of element j, where $1 \leqslant j \leqslant \mathcal{N}_{els}$
- $\mathcal{Q}_i$: number of elements containing node i, where $1 \leqslant i \leqslant \mathcal{N}_{nodes}$
- $\mathcal{B}_{i,j}$: element reference number of j-*th* element containing node i. $1 \leqslant i \leqslant \mathcal{N}_{nodes}$ and $1 \leqslant j \leqslant \mathcal{Q}_i$.

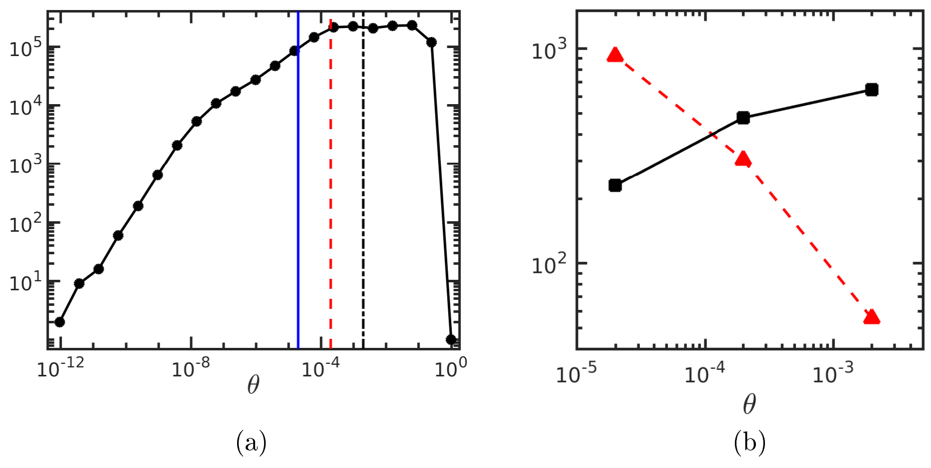

Figure 1: (a) Histogram of $\theta = v^2 + w^2$ from a single field of the sphere's wake, with zeros having been removed. θ in code units. The vertical lines correspond to the three selected thresholds in this study. (b) Volume (triangles) and number (squares) of all educed objects at the 3 selected θ. Data from a single field.

For a node i we define its volume ν_i as follows:

$$\nu_i = \sum_{j=1}^{\mathcal{Q}_i} \mathcal{V}_{\mathcal{B}_{i,j}} / \mathcal{M}_{\mathcal{B}_{i,j}}. \tag{1}$$

In words, the volume of a node is the sum of the scaled volumes of all elements that contain that node. An element's scaled volume is the volume of the element divided by the number of its nodes. Our crude definition makes the computation of a node volume a very fast process compared to more elaborate definitions based on spatial interpolation of the node boundaries, while being itself a type of interpolation which adapts the size of a node to the size of its surrounding elements.

Since the FE method provides an approximation to θ at the nodes, we decided to use these values of θ when checking the $\theta > \theta_m$ condition in the object extraction step. The value of θ with respect to the threshold is important, and no accuracy loss is caused by our method on θ. Instead, the accuracy loss we incur is in the grid point's volume computation. We note that an alternative approach would have been to interpolate the value of θ at the element's geometric center, and then use the latter as the definition of a grid point. The grid point's volume in that case would be the element's volume, which is an exactly derived quantity from x_i, y_i and z_i. In the alternative approach, the accuracy loss would be due to the interpolation of θ and none on the grid point's volume, while we opt for accuracy loss on the latter while none on θ. Our choice is justified by the fact that the exact volume of the object intersections is irrelevant. We require these volumes solely to compare them with other intersections - in step 3 of section 3 - and finding the largest one. The bias error introduced in Equation 1 will affect the absolute value of the intersections, yet their ranking will be affected to a to a lesser extent.

We now proceed to educe objects from our data set described in section 2. We chose to focus on the quantity $\theta = v^2 + w^2$, which is a marker for strong cross-stream motions typical of a sphere's wake. We plot on Figure 1(a) the distribution of θ as a histogram computed from a single field of our time sequence. We extract flow objects for three thresholds indicated on Figure 1(a) as vertical lines. The volume and number of objects as a function of the chosen threshold is shown on Figure 1(b). It can be seen that the volume of the extracted objects decreases as the threshold θ is increased, while the number of extracted objects increases.

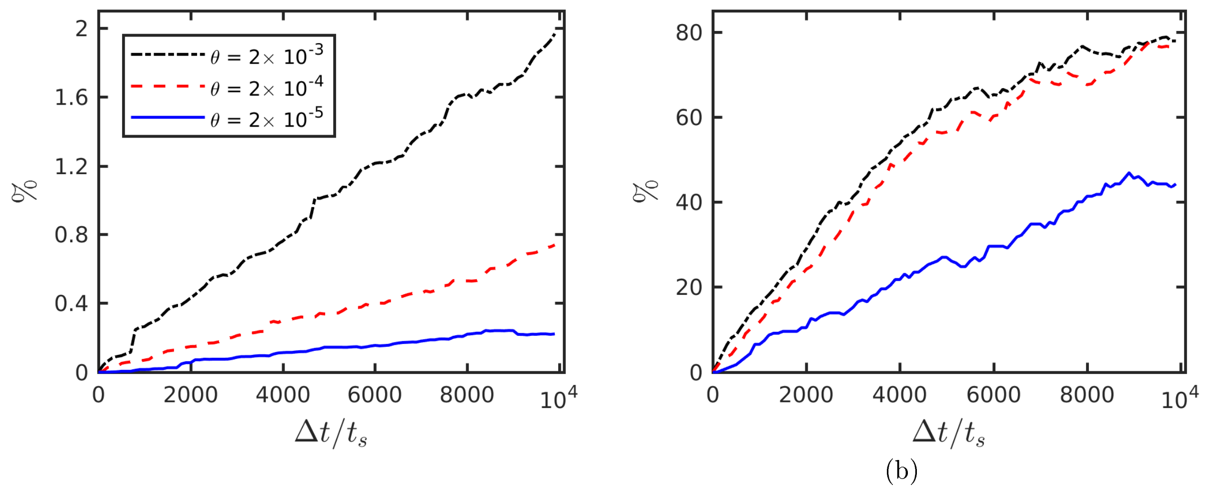

Figure 2: (a) Percentage of volume of objects at t_1 that is lost (unconnected) as the objects are linked with those at $t_1 + \Delta t$. Data for the 3 thresholds indicated in the legend and corresponding to those depicted on Figure 1 as vertical lines. t_s is the numerical time step of the code. (b) same as (a), but for the number of lost objects scaled by $m(t_1)$, the number of objects at t_1.

To assess the performance of the linking routines, we first carry out step 2 from section 2 by comparing object lists $V_i(t_1)$ with $V_i(t_1)$. This simple test is found to yield that no object is unconnected and that all objects connect with one object only. Next, we carry out step 2 from section 2 by comparing object lists $V_i(t_1)$ with those from $V_i(t_2)$, $V_i(t_3)$, $V_i(t_4)$ and so on, until $V_i(t_{100})$. Our expectation is that as the time gap is increased, more objects from $V_i(t_1)$ will become unconnected with those further in time. The unconnected volume corresponding to those objects should also increase with Δt. Both expectations are seen to be confirmed on Figure 2.

We show on Figure 2(a) the evolution of the unconnected volume for increasing $\Delta t/t_s$, where t_s is the numerical time step. After 10^4 time steps, only 2% of the volume of the objects present at t_1 has been lost for the highest threshold - and even less for lower thresholds. Figure 2(b) shows that in terms of lost objects, their fraction with respect to the initial number $m(t_1)$ is much higher than the lost volume fraction, indicating that a large number of structures carry a very low percentage of the total object volumes. This is characteristic of what has been observed in educed structures from DNS, where a preliminary step involves removing many objects - most - which are very small in size and are a result of underlying noise. Their temporal coherence is limited, and this is reflected on Figure 2(b) confirming that the temporal linking routines are producing results that exhibit the expected behavior.

4 Future work

When searching for an appropriate threshold, step 1 from section 3 is usually repeated by scanning many thresholds and obtaining the so called percolation curves for a given scalar field [11, 3]. Before we could do this, in addition to validating our code as done in the present study, we need to choose a spatially inhomogeneous threshold as a consequence of the flow's underlying inhomogeneity. Typically, this is done based on the spatially inhomogeneous mean and root-mean-square of that quantity, which are statistics that we have not converged for this simulation, which was intended solely for the purpose of code validation. Having reached the present point in the post-processing code maturity, we will explore an ALYA simulation of the turbulent jet flame (DLR A flame [12]). Here, the heavier computations involved in converging statistics for accurate threshold determination and executing step 3 from section 3 will be used to study aspects of the flame dynamics.

References

[1] A. Lozano-Durán and J. Jiménez. Time-resolved evolution of coherent structures in turbulent channels: characterization of eddies and cascades. *J. Fluid Mech.*, 759:432–471, 2014.

[2] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, et al. Alya: multiphysics engineering simulation toward exascale. *J. Comput. Phys.*, 14:15–27, 2016.

[3] J. I. Cardesa, A. Vela-Martín, and J. Jiménez. The turbulent cascade in five dimensions. *Science*, 357(6353):782–784, 2017.

[4] I. Rodríguez, R. Borell, O. Lehmkuhl, C. D. Pérez Segarra, and A. Oliva. Direct numerical simulation of the flow over a sphere at Re = 3700. *J. Fluid Mech.*, 679:263–287, 2011.

[5] S. Charnyi, T. Heister, M. A. Olshankii, and L. G. Rebholz. On conservation laws of Navier-Stokes Galerkin discretizations. *J. Comput. Phys.*, 337:289–308, 2017.

[6] F. Capuano, G. Coppola, L. Rández, and L. de Luca. Explicit Runge-Kutta schemes for incompressible flow with improved energy-conservation properties. *J. Comput. Phys.*, 328:86–94, 2017.

[7] F. X. Trias and O. Lehmkuhl. A self-adaptive strategy for the time integration of Navier-Stokes equations. *Numer. Heat Tr. B-Fund.*, 60:116–134, 2011.

[8] A. W. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: algebraic theory and applications. *Phys. Fluids*, 3670(16), 2004.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[10] J. I. Cardesa, A. Vela-Martín, S. Dong, and J. Jiménez. The temporal evolution of the energy flux across scales in homogeneous turbulence. *Physics of Fluids*, 27(11):111702, 2015.

[11] A. Lozano-Durán, O. Flores, and J. Jiménez. The three-dimensional structure of momentum transfer in turbulent channels. *J. Fluid Mech.*, 694:100–130, 2012.

[12] W. Meier, R. Barlow, Y.L. Chen, and J.Y. Chen. Raman/Rayleigh/LIF measurements in a turbulent CH4/H2/N2 jet diffusion flame: experimental techniques and turbulence-chemistry interaction. *Combust. Flame*, 123:326–343, 2000.