International Journal of Multiphase Flow 58 (2014) 185–194

Contents lists available at ScienceDirect

International Journal of Multiphase Flow

journal homepage: www.elsevier.com/locate/ijmulflow

Finite-volume droplet trajectories for icing simulation

T.C.S. Rendall *, C.B. Allen

Department of Aerospace Engineering, University of Bristol, Bristol BS8 1TR, UK

ARTICLE INFO

Article history:
Received 17 November 2012
Received in revised form 22 August 2013
Accepted 23 August 2013
Available online 6 September 2013

Keywords:
Particle tracking
Icing
CFD
Droplet motion

ABSTRACT

During a Lagrangian icing simulation a large number of droplet trajectories are calculated to determine the water catch, and as a result it is important that this procedure is as rapid as possible. In order to arrive at a method with minimum complexity, a finite-volume representation is developed for streamlines and extended to incorporate the equations of motion for a droplet, with all cells being crossed in a single timestep. However, since cells vary greatly in size, the method must be implicit to avoid an awkward stability restriction that would otherwise degrade performance. An implicit method is therefore implemented by carrying out iterations to solve for the crossing of each CFD cell, so that the droplet motion is tightly coupled to the underlying flow and mesh. By crossing every cell in a single step, and by using the mesh connectivity to track the droplet motion between cells, any need for costly searches or containment checks is eliminated and the resulting method is efficient. The implicit system is solved using functional iteration, which is feasible for the droplet system (which can be stiff) by using a particular factorisation. Stability of this iteration is explored and seen to depend primarily on the maximum power used in the empirical relationship for droplet drag coefficient $C_D = C_D(Re)$, while numerical tests confirm the theoretical orders of accuracy for the different discretisations. Final results are validated against experimental and alternative numerical water catch data for a NACA 23012 aerofoil.

1. Introduction

Calculation of droplet or solid particle trajectories through a CFD mesh is a problem that arises across a wide range of disciplines, ranging from icing simulation, to combustion modelling, chemical engineering, fire sprinkler systems and haemodynamics. Modern techniques apply CFD to calculate the flow field surrounding droplets, and then integrate Newton's second law to determine the trajectory. However, this means using the flow field information (which usually consists of just the velocity, but it might also include temperature or other flow parameters), which is Eulerian in nature, within a Lagrangian calculation, and hence requires a method of tracking.

Water catch calculations used in icing analysis may be conducted in either a Lagrangian (Ruff and Berkowitz, 1990) or Eulerian (Beaugendre et al., 2003; Bourgault et al., 1997) frame of reference. The Lagrangian approach is more common, and requires droplet trajectories to be calculated, whereas the Eulerian approach solves transport equations to determine the water catch on the surface of the object directly. For small droplets an Eulerian frame of reference is likely to be superior, but for larger droplets the assumption of a volume weighted droplet quantity breaks down. Larger droplets are also likely to exhibit behaviour that is more easily expressed in a Lagrangian frame, such as splashing, and regulatory bodies will soon require certification under these conditions. This work describes methods to perform trajectory calculations for the Lagrangian approach.

For these computations the form of the underlying flow solution is very important. If it is a potential flow solution (Ruff and Berkowitz, 1990), which is often used for a rapid analysis, then it is straightforward to evaluate the flow velocity at any arbitrary position from the velocity potential. In comparison, a CFD volume mesh (Potapczuk, 1992) requires the droplet to be tracked as it passes through the mesh from cell to cell. An interpolation within each cell for the flow velocity may be used for increased accuracy. It is expected that as the development and usage of CFD progresses, together with a need to analyse more elaborate shapes, tracking droplets through a volume mesh will become more common in icing calculations.

Tracking a massless particle (to give a streamline) through a finite-volume mesh is in principle straightforward, but requires an efficient method for finding which CFD cell in a mesh contains the particle, as many millions of droplets may ultimately need to be tracked. The level of complexity is increased by introducing the droplet equations of motion, as these can come with timestep stability restrictions, and for efficiency it is necessary to resolve these numerical limitations with the method of tracking. It is this combination of tracking and numerical integration that is of

* Corresponding author. Tel.: +44 117 331 5639.
E-mail address: thomas.rendall@bristol.ac.uk (T.C.S. Rendall).

0301-9322/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ijmultiphaseflow.2013.08.007

principal interest here; for the tracking method to work, every cell must be crossed in an integer number of timesteps (here, always one), which means the integrator must be able to handle any timestep, even if the system is stiff.

This work starts by describing the simplest possible way to represent streamlines on a finite-volume mesh, before incorporating the equations of motion to give a trajectory. The final result is a method that is fast, adaptive to the volume mesh resolution and without timestep restrictions. The basic methods are first and second order accurate in space and time, but these orders may be increased at the expense of more restrictive stability bounds if needed, and alternative integrators can also be fitted in the face intersecting framework.

2. Implicit mesh dependent streamlines and droplet trajectories

The nonlinear ordinary differential equations that describe the trajectory of a water droplet are usually solved by a numerical integration technique. Many methods use an explicit single or multi-stage approach, with Runge–Kutta and predictor/corrector techniques being common, although for the stiff equations that result from low mass particles integrators designed for stiff systems are preferred. Nonlinearity arises both from the dependence of the droplet drag coefficient on the Reynolds number, and the link to the dynamic pressure of the relative flow speed between the droplet and the air.

The choice between an explicit and an implicit integration scheme is fundamental (Kim and Elangovan, 1986). An explicit scheme imposes a stability restriction on the timestep, while an implicit scheme is left largely unfettered in this regard and the timestep only needs to be guided by accuracy considerations. An implicit droplet scheme is comparatively much simpler than an implicit CFD scheme, and requires in the simplest case only the solution of a single nonlinear equation for each timestep. The cost and difficulty of simulating large numbers of trajectories of small droplets has been noted previously, both by Kim and Elangovan (1986) who suggested a switch towards semi-implicit schemes, and Caruso (1993) who advocated parallel processing of trajectories.

2.1. Streamlines

The operations required to calculate a droplet trajectory are closely related to those used to find a streamline, so for simplicity this case shall be considered first. Many streamline methods involve using cell neighbours to compute velocity interpolations, and then integrate the velocity vector to find the streamline, usually taking multiple integration steps within a cell. Although the apparent accuracy of this approach may appear good (with a large number of steps in a cell giving a smoothly curving streamline) the actual accuracy still depends on the underlying flow velocity. The flow equations are typically solved using a finite-volume (FV) discretisation, and this uses a constant velocity within a cell (often with a gradient based face reconstruction procedure), so although an attempt to increase the accuracy of the streamline beyond that of the solver may be visually pleasing it is not always based on the underlying discretisation. Further, it is clear that there exists a consistent 'finite-volume' streamline that uses the cell velocities and no other information. It is this fundamental FV streamline description that presents an efficient and accurate way of tracing streamlines and trajectories.

A FV streamline consists only of straight line segments pointing along the direction of the flow velocity in each cell (note that if the containment cell velocity is taken, the method is first order), while points along the streamline are defined by the intersections of these segments with cell faces, as illustrated in Fig. 2. Importantly FV streamlines do not require integration of any differential equation to be traced out. They are defined only by the discrete flow solution and the mesh, i.e. they are geometrical in nature rather than mechanical. Streamlines may also be considered as the trajectories of massless particles; since these particles have no mass, they accelerate instantaneously to match the flow velocity in any cell of the mesh. This instantaneous acceleration leads to the discontinuities in velocity between cells.

The process of finding a streamline in this manner is simple and uses only the connectivity information already required by the flow solver. Two kinds of connectivity are required: (1) edge-cell indexing and (2) cell-edge indexing. Edge-cell indexing allows the neighbouring cells (or boundary conditions) of a particular edge to be found, while cell-edge indexing allows the neighbouring edges of a particular cell to be found. The data required for edge-cell connections are already used by the flow solver, so no work is required here, while cell-edge connectivity may be determined exactly from the edge-cell data in a simple preprocessing step.

With this connectivity in hand, the streamline is found by first placing a seed point within a known cell. Intersection points of the velocity ray from this seed point to the infinite planes of all the cell faces are then found, with the nearest one that lies in the positive direction along the velocity vector finally being chosen as the correct point. Once the exit edge has been found, the next seed point is this intersection point, and the next velocity ray is determined by the velocity in the cell on the other side of the edge. Alternatively, the other side of the edge may be a boundary, in which case the intersection point is recorded and the process stops. Exactly one step is therefore required per cell, and the cost of finding a streamline is proportional to the mesh size; the only searching that is needed takes place over the boundary edges of a cell and since this is usually a small number (3–6) that is roughly constant within the mesh this does not add a significant searching expense. Visually it can be seen that a number of postprocessing tools, commercial and in-house, use this technique.

The complete process for a streamline is:

1. Pick a starting point in a known cell.
2. Using the cell-edge connectivity calculate all intersections between the velocity vector for that cell with the faces of that cell.
3. Select the nearest intersection point in the positive direction along the velocity vector as the next point for the streamline. Alternatively, select negative intersections for the upwind streamline.
4. Use the edge-cell connectivity to locate the cell on the other side of this edge, or terminate if this is a boundary edge, and then move into this cell.

In a small number of cases the intersection between the velocity vector and a face may be extremely flat. In this case, roundoff errors can cause intersections in the positive direction to be mistakenly found as negative. To escape this error the method of Haselbacher et al. (2007) is applied; the actual point of intersection is left unchanged and the streamline moved into the cell on the other side of the edge in question.

Overall this method achieves first order accuracy if the cell flow velocity is used to intersect the faces. This can be improved to second order if a gradient extrapolation is used to the face, so that

$$\mathbf{v}_f = \mathbf{v}_c + \nabla\mathbf{v} \cdot \mathbf{d} \tag{1}$$

where $\mathbf{d}$ is the vector from the cell centre to the face intersection point. The consistent velocity to use to intersect the faces of the cell