JOURNAL OF AIRCRAFT
Vol. 44, No. 2, March–April 2007

Wake Vortex Model for Real-Time Flight Simulation Based on Large Eddy Simulation

Graham T. Spence,* Alan Le Moigne,† David J. Allerton,‡ and Ning Qin§
University of Sheffield, Sheffield, S1 3JD England, United Kingdom

DOI: 10.2514/1.23761

An alternative approach is presented to represent a counter-rotating vortex pair for wake vortex interaction in a real-time flight simulation. Rather than using an analytical model, the model directly accesses a precomputed three-dimensional time-varying dataset. This dataset is generated using implicit large eddy simulations on unstructured grids. The methodology enables a realistic vortex decay to be reproduced with the apparition of short- and long-wave instabilities. The vortex model based on these simulations is designed to run on a personal computer and is integrated with a six-degree-of-freedom real-time flight simulator. The paper combines the use of large eddy simulation data as the basis of a wake vortex encounter model with methods to access the dataset in real time. The second half of this paper describes the data compression scheme implemented to reduce the size of the time-varying dataset. Additionally, rapid data access and issues regarding the real-time data management aspects are discussed. Analysis of the performance of the model is made and preliminary comparisons with a traditional analytical wake vortex model are discussed.

Nomenclature

B	=	wing span
b_0	=	reference length, initial vortex spacing
d	=	distance
F	=	local field in original mesh
F'	=	local field in simplified mesh
g	=	gravitational acceleration
M	=	mass of aircraft
M_∞	=	freestream Mach number
N	=	number of data points
n_{cand}	=	number of prefetching candidates
$\boldsymbol{p}$	=	position vector
Re	=	Reynolds number
r_c	=	core radius
s	=	spanwise load factor, taken equal to $\pi/4$ here
T_n	=	tetrahedron index
t	=	dimensional time
t^*	=	nondimensional time
t_0	=	reference time
V	=	aircraft speed
$\boldsymbol{v}$	=	velocity vector
v_n	=	vertex index
w_0	=	reference velocity, descent speed of a vortex pair
Γ_0	=	initial circulation
ε_f	=	local field error
ν	=	kinematic viscosity
ρ	=	air density
ω	=	vorticity

Received 13 March 2006; revision received 9 June 2006; accepted for publication 4 August 2006.

*Research Associate, Automatic Control and Systems Engineering, Mappin Street. Member AIAA.
†Research Associate, Mechanical Engineering, Mappin Street. Member AIAA.
‡Professor, Automatic Control and Systems Engineering, Mappin Street.
§Professor, Mechanical Engineering, Mappin Street. Associate Fellow AIAA.

I. Introduction

THE effect of aircraft generated trailing wake turbulence on an encountering aircraft can be hazardous, particularly during the takeoff and landing phases, as seen in the American Airlines Flight 587 accident over New York on 12 November 2001 [1]. Wake vortices [2,3] generated by civil transport aircraft are the result of the lift generated by the aircraft wing and tail and so their formation is unavoidable. Consequently, aviation authorities impose strict minimum separation rules for departures and approaches in an attempt to minimize the likelihood of encountering wake vortices. However, these rules also restrict the capacity of airports. With the continued growth in air traffic, there is increasing interest toward gaining a greater understanding of wake vortex behavior, not only in terms of safety but also from the perspective of increasing airport capacity. The ability to model the effect of aircraft trailing wake vortices in real time provides several benefits: 1) training flight crew to cope with potentially hazardous situations, including wake encounters; 2) aiding the determination of acceptable safety limits or passenger comfort aspects in the event of a wake encounter; and 3) validation of wake vortex models in comparison with events that have been experienced during real-life wake encounters.

Several analytic models of the post *roll-up* flowfield of aircraft-wake vortices exist. A summary of these vortex models is presented by Gerz et al. [4]. These models are suitable for real-time use and some have been used in previous flight simulation studies [5–8]. However, until recently, most real-time wake turbulence encounter simulation studies have been limited to models that describe the velocity field in two dimensions. Although computationally cheap, such approaches lack the ability to include the three-dimensional flow structures that are observed in actual wake turbulence [9]. Real-time studies of airplane encounters with analytic models of perturbed wake vortex systems are now emerging [10]. An attempt at using large eddy simulation (LES) data to form the basis of an analytic vortex model has also been presented [11]. This study derived 2-D velocity flowfields from 3-D LES data. 3-D instability effects were then created by varying the flowfield in a sinusoidal fashion. Until now, no real-time aircraft-wake encounter simulations have attempted to directly access the data from LES to provide a real-time model of a wake vortex system. This is the major contribution described in this paper. A similar model has been used in a study of the influence of aircraft encountering wake vortices in a convective boundary layer [12], however, it is not a real-time model. The idea of

coupling computational fluid dynamics (CFD) data into flight simulations can be traced back to the mid 1990s [13]. The generation of CFD *datamaps* was suggested as a technique to integrate the flow of air around ships during simulations of close proximity helicopter maneuvers. More recently, combined ship airwake and aircraft simulations [14,15] have become feasible due to advances in CFD and computational resources.

It has been demonstrated that the decay of aircraft trailing wake vortices can be simulated using CFD methods, in particular, LES. Such computations [12,16–21] require significant resources and time. However, their results have shown the LES can be used to capture three-dimensional wake vortex decay in detail. We employ a similar technique although ours is based on an unstructured grid approach [22], which allows a clustering of the grid points in the proximity of the vortex cores and a coarser spacing for the rest of the domain, thus saving on the computing cost.

If the computed flowfields can be conveniently stored and organized for rapid access, this information can be retrieved in real time to provide the basis for an alternative model of wake vortex decay for use in flight simulation. Such a model may potentially provide added realism and flexibility to wake vortex encounter simulations. Ideally, the interaction of the aircraft and the decaying wake vortex may also be accounted for using LES, but this is difficult to achieve in the real-time simulation. Nevertheless, incorporating a more realistic wake model combined with a strip theory model for the interaction is a significant step toward improving flight simulation models. If the LES simulations can be considered sufficiently representative of real wake decay, it should also be possible to generate different datasets for the encounter simulations. Unfortunately, the use of LES data in a real-time encounter simulation introduces significant challenges, because computation of the wake model is significantly more complex than analytic models. In summary, these challenges include: 1) how to access the data quickly; 2) organization and management of memory; 3) positioning and orienting the vortex data in the simulated environment; and 4) integration of the flight dynamics model with the flowfield so that the wake encountering aircraft experiences the precomputed flowfields.

The real-time system presented in this paper shows how various techniques can be integrated to directly fly through LES generated data. The system is based on using a PC to host the wake vortex model and its source LES time-varying dataset. A compression scheme, based upon the simplification of meshes and its underlying data, is applied to the entire dataset because the volume of the datasets can be considerably large. Data compression is feasible because the spatial resolution of the LES data exceeds the detail required by the aircraft-wake interaction model. Postcompression, the data points are extracted from the mesh and converted into a form in which unstructured data points can be located rapidly. In this case, the data are organized by spatial data structures. For this encounter model, hard disk accesses introduce an unacceptable performance reduction so memory management methods are used to swap in relevant data time steps before computation of the aircraft-wake encounter model. This approach allows the software to access the time-varying dataset in a timely and I/O efficient manner, without significantly affecting the real-time performance of the flight simulation. As with previous wake encounter simulations, the velocity field of the wake system remains unperturbed by the encountering aircraft (uncoupled model).

The first part of the paper focuses on the CFD aspects, describing the methodology employed in the large eddy simulations of the wake vortices. The second part describes the integration of this time-varying dataset into a research flight simulator, covering the reduction in the volume of data while still retaining acceptable data resolution for the wake vortex encounter simulation. Software techniques used for fast data point queries and the memory management of individual time steps are presented. The flowfield data extraction techniques used in the flight simulator are described, together with discussion of the performance and validation issues. Finally, the model is compared with a two-dimensional analytical wake vortex model.

II. Large Eddy Simulation of a Pair of Wake Vortices

A. Numerical Methods

The LES calculation is performed with the CFD code uns-MERLIN, which has been jointly developed by Cranfield University and the University of Sheffield. It is a compressible unsteady three-dimensional finite-volume Euler/Navier–Stokes solver working on unstructured and hybrid grids. It is parallelized using the message passing interface (MPI) standard, Metis being used for the domain decomposition. The time integration is performed with an explicit three-stage Runge–Kutta scheme.

An implicit LES approach is employed, that is, no subgrid scale model is used. Instead, we rely on the numerical dissipation to act as a subgrid scale model to transfer energy from the resolved scales to the modeled ones. This is similar to the concept of monotone integrated large eddy simulation (MILES) [23]. Stumpf [16] used MILES for simulating the decay of vortices in the far field. His solver is a pure advection scheme which omits the calculation of diffusive fluxes. We follow a similar approach, because it is believed that the flow problem is essentially a convection dominated problem, and the large scale turbulence is a result of the flow instabilities due to the convection terms. Thus uns-MERLIN is used in the unsteady Euler mode in this work.

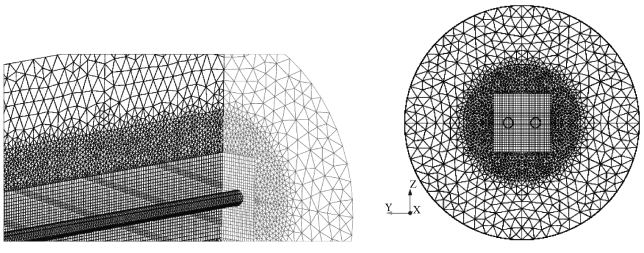
The computational domain considered has the form of a horizontal cylinder, its length being aligned with the axes of the vortices. It is 408 m long to correspond to the wavelength of a Crow instability [9]. The grid is unstructured, which makes it possible to cluster grid points in the region where the vortices are and to provide a coarser resolution away from the vortices to reduce the computational effort of the simulations. From our numerical investigation [22], it was found that a mesh consisting of a core of Cartesian structured grid surrounded by tetrahedra to the far-field boundary was less numerically dissipative than a fully tetrahedral grid and hence was more accurate to simulate the evolution of the vortices, as compared with experimental vortex descending trajectories. Two views of such a grid are shown in Fig. 1.

Using a grid that combines coarse and fine resolution for a 2-vortex simulation is only meaningful if the vortices stay in the refined region of the grid. However, two counter-rotating vortices induce on each other a downward velocity, so that the vortex system descends with time. To obtain fixed vortices in the computational domain, a Lagrangian referential "attached" to the vortices has to be employed in the simulation. This is achieved by imposing a freestream constant vertical velocity at the circumferential boundary, equal and opposite to the theoretical descent velocity [4]

$$w_0 = \frac{\Gamma_0}{2\pi b_0} \tag{1}$$

The vortices that have been simulated are based on a Boeing 747 first generation (−100, −200, or −300) aircraft in the approach condition (2000 ft, 155 kn) and with a relatively low weight (248,000 kg) for which $\Gamma_0 = 565\ \mathrm{m^2/s}$ and $b_0 = 47\ \mathrm{m}$ according to [4]

$$\Gamma_0 = \frac{Mg}{\rho s B V} \quad \text{and} \quad b_0 = sB \tag{2}$$



a) b)

Fig. 1 View of the computational grid. a) cut through the grid; b) extremity of the grid.

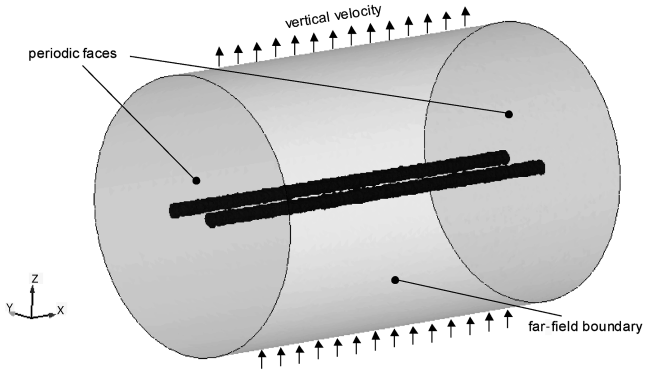


Fig. 2 Boundary conditions.

These values have been chosen to match one of the cases simulated by Holzäpfel et al. [17]. For this vortex system, $w_0 = 1.91$ m/s, that is, the vortices "see" an ascending flow with a Mach number $M_\infty = 0.005624$. The circumferential boundary condition is hence set as a characteristic-based far-field boundary condition with those freestream conditions. The two faces at both extremities of the cylinder are set as periodic boundaries to simulate a pair of long vortices. The boundary conditions are illustrated in Fig. 2.

The flowfield is initialized as a superposition of two Lamb–Oseen vortices with a core radius $r_c = 4$ m. Some turbulence is added to the initial vortex flowfield to represent the turbulence generated by an aircraft passing through the air as well as the ambient atmospheric turbulence. The turbulence is simulated by the addition of a random perturbation (white noise) to the three components of velocity following the approach of Holzäpfel et al. [17].

B. Time Evolution of the Wake Vortices

All the CFD simulations are run for a total real time of 200 s. The computation takes on the order of one week running on 30 processors of a Linux cluster. Figure 3 shows four snapshots of the flowfield at different times. The time is nondimensionalized as follows:

$$t^* = t/t_0 \tag{3}$$

where

$$t_0 = \frac{2\pi b_0^2}{\Gamma_0} = 24.6 \text{ s} \tag{4}$$

The structure of the vortices is shown with two isosurfaces of vorticity magnitude: the isosurface in dark gray corresponds to the high value of vorticity magnitude to represent the vortex cores while the isosurface in light gray shows the low vorticity of the ambient

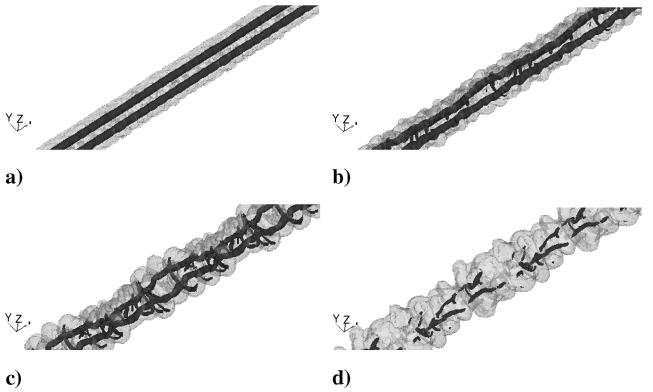


Fig. 3 Time evolution of the pair of vortices. Isosurfaces of vorticity magnitude ($\|\omega\|$) are shown: $\|\omega\| = 5 \times 10^{-4}$ in dark gray, $\|\omega\| = 5 \times 10^{-5}$ in light gray. Several computational domains have been put end to end to show the periodicity of the calculation. Approximately two computational domains can be seen in each picture. a) $t^* = 2.02$ ($t = 49.7$ s). b) $t^* = 4.09$ ($t = 100.6$ s). c) $t^* = 6.11$ ($t = 150.3$ s). d) $t^* = 8.08$ ($t = 198.8$ s).

turbulence generated by the decay of the vortices. Although not clearly visible in Fig. 3, short-wave (elliptic [24]) instabilities start to form at around $t^* = 2.2$. Long-wave instabilities (Crow [9]) are more evident in Figs. 3b and 3c and develop at around $t^* = 3.2$. They cause the vortex cores to come closer to each other. The deformation of the vortices produces "ribs" of vorticity underneath the vortex cores (Fig. 3c) as observed in other simulations [17,21]. Experiments by Leweke and Williamson [24] in a water tank revealed that these vorticity structures are created by the short-wave length instabilities. For each wavelength of these instabilities, two counter-rotating vortices are created perpendicular and beneath the two main vortices. These transverse vortices are well reproduced in the direct numerical simulation (DNS) of that experiment [21]. Although our simulations represent a different case (our ratio $r_0/b_0 = 0.076$ and Reynolds number $Re = \Gamma_0/\nu = 15 \times 10^6$ are representative of actual wake vortices as opposed to the values 0.2 and 2700, respectively, in the experiment), similar vorticity structures were found in their simulations. Finally the decay of the main pair of vortices is observed in Fig. 3d.

III. Integration of the LES Data into a Flight Simulator

A. Processing of the Dataset

1. *Dataset Compression*

Two hundred seconds of wake vortex decay are simulated by LES and a solution file is saved approximately every 0.5 s of simulated flow, yielding a dataset size of approximately 20 GB. The time needed to access this unprocessed data would be prohibitive for the real-time access required by the wake encounter simulation. A compression scheme is implemented to reduce the run-time memory footprint of this time-varying dataset. With this approach, the wake vortex model is manageable (in terms of memory management) on a dedicated desktop PC. The resolution of the strip model (the distance between adjacent wing strips) representing the encountering aircraft is lower than that of the LES dataset, and a coarser, or compressed representation of the dataset may be adequate for real-time applications. The fine resolution of the CFD grid is necessary to obtain an accurate LES solution. Although it is possible to coarsen the LES dataset (postcomputation), it is not acceptable to use a coarser CFD grid to perform the LES simulations. Although the computational effort of generating the dataset is reduced and possibly the compression step described in this section could be omitted, the physics of the vortex decay would not be correctly modeled.

Lossy compression techniques provide much higher compression ratios than traditional lossless techniques. However, to use *lossy-block* compression methods, such as wavelet transforms [25], unstructured datasets have to be resampled on to a structured grid. This constraint significantly increases the dataset memory footprint *before* compression and would require the implementation of a fast run-time decompression scheme.

Rather than resampling each time step of the dataset and using a block compression method, a *mesh simplification* technique [26] is used to reduce the size of the dataset. The mesh simplification process is based on the incremental operation of *edge collapsing* the underlying tetrahedra and is illustrated in Fig. 4. To collapse an edge in a tetrahedral mesh a vertex is moved to an adjacent vertex. This operation reduces several tetrahedra to zero volume (degenerate), which can be removed from the mesh. Data values associated with the modified vertex are updated to reflect the modified vertex location. As a result of the edge collapse, a field error in the data is introduced. The field error refers to the amount by which the local data in a modified mesh have deviated from the local data in the original mesh. Because the simplified mesh has been modified by the removal of a selected vertex, a linear tetrahedral interpolation is used to determine a local field error at a point within the tetrahedron that was distorted to occupy the space left as a result of degenerate tetrahedra. The local field error is defined in Eq. (5) and in this case provides a measure of the error of the velocity magnitude at a point in the simplified field F', relative to the value in the original field, F.

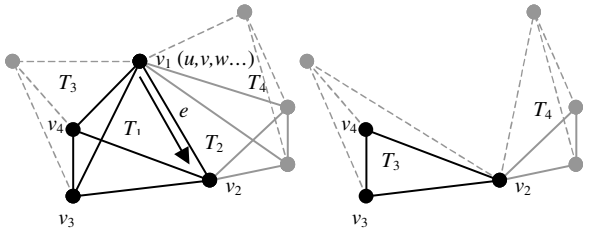


Fig. 4 An example of tetrahedral edge-collapsing operation. Edge e is collapsed from vertex v_1 to vertex v_2, removing v_1 and tetrahedra T_1 and T_2.

$$\varepsilon_f = |F[\sqrt{(v_x^2 + v_y^2 + v_z^2)}] - F'[\sqrt{(v_x^2 + v_y^2 + v_z^2)}]| \qquad (5)$$

Although the local field error has been defined in terms of the velocity magnitude, the local field error could be composed of any variety of variables. For example, the value of vorticity could be used to define the local field error. Mesh simplification proceeds with an initial step of determining all of the possible field errors resulting from the collapse of each existing edge. These potential edge collapse errors are arranged in a priority queue. Edges with the lowest predicted local field error are collapsed first, and edges resulting in a high local field error are only collapsed in cases when a user desires aggressive compression. Typically, the individual meshes of each time step are reduced to approximately 15% of their original size while still retaining the important wake vortex flow characteristics.

Although mesh simplification techniques tend to not achieve the compression ratios of lossy-block methods, this method of data reduction is attractive for two reasons. First, mesh simplification techniques can be applied to nearly all mesh topologies (assuming a particular cell can be decomposed into tetrahedra), making this data reduction approach independent of the mesh type. Secondly, computational resources are not required for any data decompression tasks during the real-time wake encounter calculations.

2. *Data Organization for Fast Queries*

For the purpose of embedding the CFD generated data into the flight simulator, each time step is transformed into a pure point set by removing any mesh connectivity information. Not only does this assist in further data reduction, it provides the data as a *cloud* of points. By providing the lookup data in a point-only format, the flight simulator can accommodate data from a variety of sources (CFD data, experimental wind tunnel data, field data from light detection and ranging, and different vector field simplification techniques). Organizing and searching for a data point in an arbitrary three-dimensional field of points can be tackled by the use of hierarchical spatial data structures. These data structures capture spatial coherence of data and lead to efficient search routines based on data locations and spatial proximity.

Spatial data structures such as bsp-trees [27], oct-trees [28], kd-trees [29], and many others [30], together with accompanying spatial query operations provide excellent ways to partition and spatially organize point data sets as hierarchies, or trees. Access to the data points stored at the leaves involves navigating tree structures and is on average completed in $\log N$ time.

Three candidate spatial data structures (kd-trees, bsp-trees, and oct-trees) were evaluated for data point organization in this project [31]. Overall, the kd-tree [29,32] provided the most effective solution to spatial data organization of the individual LES time steps, due to its traversal times and efficient memory usage. The performance of the adaptive kd-tree, which has been implemented to access arbitrary data points in a three-dimensional point field, is presented in Sec. III.D.3. Other fast data querying techniques exist, such as straightforward array based indexing; however, trees provide a flexible method for accessing both structured and unstructured data.

B. Dataset Management

Typically, it is impractical to store large time-varying datasets within the physical RAM of a workstation. In order for the flight simulator to sustain an iteration rate of at least 50 Hz, a mechanism is required in which data stored on hard disk can be swapped into the main memory sufficiently in advance of its use in any wake interaction computations. Data *prefetching* techniques have been used to build such functionality into the wake encounter application and are presented in the following sections.

1. *Disk I/O Efficiency*

Each time step in the dataset is stored as an individual file. To optimize the process of reading individual time steps from hard disk, a pointerless version of a kd-tree has been developed that uses byte offsets to locate nodes in a contiguous block of memory as shown in Fig. 5. Using this method, the memory format of the kd-tree on disk maps exactly to the format required in RAM by the wake interaction calculations. Such memory layouts allow individual time steps to be loaded in a single system call.

This approach supports efficient disk I/O. For example, an 8 MB contiguous memory formatted kd-tree can be loaded in approximately 0.2 s on the host computer. This equates to a data transfer rate of about 40 MB per second, which is close to the theoretical average sustained data transfer rate of the hard disk of 60 MB per second (disk specification: ATA-100 interface, 7200 rpm; 8.5 ms access time; 683 Mbits per second internal sequential transfer rate). Using a contiguous memory data format minimizes the number of accesses to the hard disk and a pointerless tree representation eliminates any postloading data processing (modifying pointers).

2. *Application Controlled Memory Management*

To create a viable application framework for larger-than-RAM datasets, an efficient data I/O system is integrated with the flight simulator software. A strong case has been made for entirely removing the normal operating system behavior from the task of paging within an application [33]. The argument is that virtual memory schemes of most operating systems have been designed for general-purpose use and consequently, they are inefficient for specific applications. Application controlled segmentation schemes [34,35] have been shown to improve performance over standard virtual memory schemes, but with the constraint that each segment is smaller than the available physical memory.

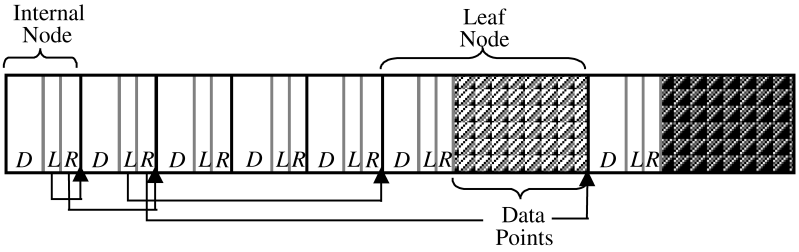


Fig. 5 Simplified contiguous memory layout. L and R represent the left and right child memory byte offsets, respectively. D is a node's internal data section containing the spatial partitioning information. Any remaining space in a leaf node is the allocated space for data point storage (patterned areas).

Although it has been suggested that virtual memory is the antithesis to real-time computing [36], it cannot be practically removed or disabled because general-purpose operating systems are dependent on such schemes. To minimize the virtual memory interference of the host operating system, the wake encounter application contains an application segmented cache. Each LES data time step is regarded as an individual cache segment. A data cache of a user-defined size can be allocated and then segmented into equal size chunks as defined by the size of the largest time-step file. This design simplifies the implementation, ensuring that when overwriting a segment with a new data block, no adjacent segments are overwritten. Once the cache system is initialized, it handles all of the time-step requests for memory required by the simulation, leaving memory allocation for noncache related tasks under the control of the operating system. The contiguous kd-tree memory layout further assists with data caching because procedures within the cache are simplified to single memory writing operations.

3. Data Prefetching

As previously mentioned, time steps requested by the wake encounter calculations have to be resident in physical memory before any computations can proceed. *Speculative prefetching* methods attempt to predict which pages, or in this case data blocks, are required in main memory before computational access and can vastly improve the performance of an application.

When considering an I/O process that transfers megabytes of data from disk to RAM, it is likely that *on-demand* disk operations will severely disrupt the desired iteration rate. It is undesirable to combine the real-time wake vortex encounter calculation task with the time-step I/O task, within a single computational process. Consequently, a multiple threaded application design has been developed to alleviate the task blocking nature of I/O as shown in Fig. 6.

The first processing thread generates and executes data lookup queries to obtain the relevant wake vortex flow field data from the application cache. A second thread is dedicated to prefetching candidate time steps from secondary memory. For wake vortex encounters, the prefetching task can be reduced to a one-dimension problem by making use of a decay time line. An encountering aircraft can only access data along this time line, which represents the recorded instants of the decaying wake (see Sec. III.C.2). The basis of this speculative prefetching method employs a sliding scale based on a directional travel bias on the wake axis. Rather than fetching all n_{cand} time-step candidates in the direction of travel along the wake axis, this scheme uses the relative velocity of both aircraft along the wake axis to adjust an initially balanced window by an amount determined by a simple scale as shown at the end of Sec. III.C.2.

C. Flight Simulator Integration

1. Flight Simulator Overview

The wake vortex model has been integrated with a six-degree-of-freedom research flight simulator [37]. The real-time simulator is distributed over eight PCs, connected via a local area network. The modular approach of partitioning the flight simulation software over several computers allows integration of an additional dedicated wake vortex model hosting computer into the flight simulator with minimum disruption to the baseline flight model.

2. Wake Vortex Data Positioning

To simulate flight through the wake vortex system trailing behind the wake generating aircraft, the 3-D blocks of data must be positioned appropriately to ensure that the encountering aircraft accesses the relevant instant of wake vortex decay. We can consider a vector emanating backwards from the wake generating aircraft to represent the approximate direction of the wake vortex system. This vector is termed the wake axis, $\boldsymbol{v}_{\text{wake}}$, and can be defined as a unit vector in a direction opposite to a unit flight vector, $\boldsymbol{v}_{\text{gen}}$. The current time step can be determined by calculating the distance d_{proj} of the encountering aircraft behind the wake generating aircraft-wake roll-up point $\boldsymbol{p}_{\text{roll up}}$ in the wake system axis. This is achieved by projecting the position of the encountering aircraft, $\boldsymbol{p}_{\text{enc}}$, onto the vector defining the wake system axis, $\boldsymbol{v}_{\text{wake}}$, using a vector dot product operation as shown in Eq. (6).

$$d_{\text{proj}} = \boldsymbol{v}_{\text{wake}} \cdot (\boldsymbol{p}_{\text{enc}} - \boldsymbol{p}_{\text{roll up}}) \tag{6}$$

Using distance d_{proj} and the speed of the leading (wake generating) aircraft, the time behind the wake roll-up point can be calculated to determine the appropriate time step in the dataset. To recreate the effect of flying through the data blocks, a data tiling system has been created. Tiling the data allows seamless repetitive transitions when the aircraft travels between data blocks.

The tiling concept provides a grid that fixes the initial locations of the data blocks in the environment world coordinates (Fig. 7). Tile locations are oriented (in both azimuth and elevation) along the wake axis, with the size of each tile equating to the length of the CFD domain to ensure no gaps exist between adjacent tiles. Using a fixed reference point, $\boldsymbol{p}_{\text{initial}}$, to define the start of the tiling grid, the distance that the wake generator has traveled from the tile reference point can be determined. The placement location of a particular tile can be calculated as outlined in Eqs. (7–11).

First, the distance between the wake roll-up point and the following aircraft along the wake axis is calculated using a dot product operation.

$$d_1 = \boldsymbol{v}_{\text{wake}} \cdot (\boldsymbol{p}_{\text{enc}} - \boldsymbol{p}_{\text{roll up}}) \tag{7}$$

Secondly, the distance that the roll-up point of the leading aircraft has moved from the initial start position is computed.

$$d_2 = \text{vector dist}(\boldsymbol{p}_{\text{initial}}, \boldsymbol{p}_{\text{roll up}}) \tag{8}$$

The number of tiles that fit into the distance d_3 (the distance from initial start position to the following aircraft) is determined by dividing d_3 by the CFD domain length.

$$d_3 = d_2 - d_1 \tag{9}$$

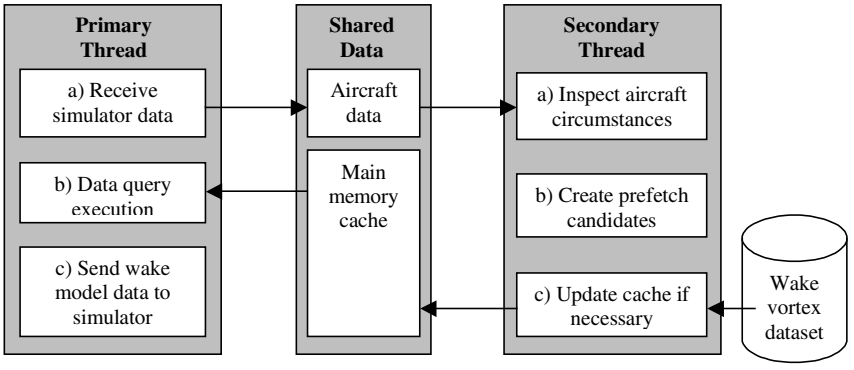


Fig. 6 Illustration of the dual-threaded application design, with arrows representing the direction of dataflow between the threads. Both threads require access to individual segments of the main memory cache, and so operations on a segment require an exclusive memory lock.

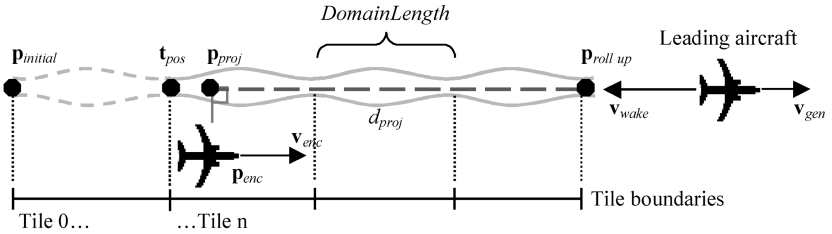


Fig. 7 Diagram of the various point locations, vectors, and distances used in the determination of the currently accessed time step and the current tile location.

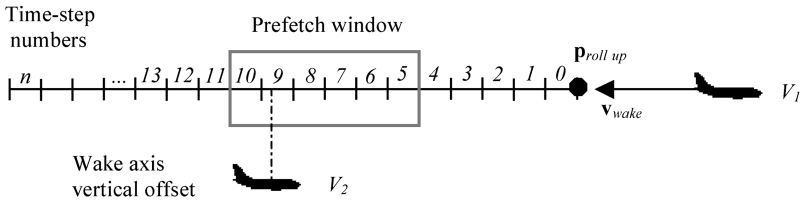


Fig. 8 Diagram showing the leading aircraft with velocity V_1, and the wake encountering aircraft with velocity V_2. In this example, V_2 is greater than V_1; therefore the time-step prefetching bias is directed toward younger wake decay instants.

$$\text{tile number} = \text{integer}\left(\frac{d_3}{\text{domain length}}\right) \tag{10}$$

Finally, the position of the tile in world coordinates, $\boldsymbol{t}_{\text{pos}}$, is calculated by projecting a point from the initial start position along the flight vector of the leading aircraft.

$$\boldsymbol{t}_{\text{pos}} = \boldsymbol{p}_{\text{initial}} + [(\text{tile number} \times \text{domain length})\boldsymbol{v}_{\text{gen}}] \tag{11}$$

The positioning of data blocks at any particular tile location is independent from the current decay instant or time step. This allows the referenced time step to change while an aircraft encountering the vortex is midtile, not just on tile boundaries. Such a feature is feasible because the change in the velocity field between two adjacent time steps is small; therefore unrealistic jumps in the forces acting upon the encountering aircraft are negligible. To emulate the descent of the wake system, an offset from the wake axis is added to the position of a data tile for each encountered time step (Fig. 8). The magnitude of this offset is determined using the wake descent velocity w_0 (Sec. II. A) and the current decay instant.

3. *Wake Vortex Interaction Model*

The effect of the wake vortex flowfield on the encountering aircraft is calculated by a strip theory method [6,38]. The notable modification in the strip calculations used in this study is how the velocities are retrieved from the time steps of a dataset. Each point that defines a lifting surface strip is tested for intersection with a bounding box defining the domain of an instant of wake decay data. Points intersecting with the domain are then used as query points for searches in the kd-tree representing the current time step. The searches will locate vectors in the wake flowfield that are closest to each of the strip points; these vectors are then transformed into the aircraft body axis system for use in the strip calculations.

D. Model Performance

Previous sections have described the building blocks used to develop the LES based wake vortex encounter simulation. The different methods used to generate the offline vortices have a significant impact on the real-time simulation.

1. *System Scalability*

The use of an LES based wake vortex model raises questions as to the limits on the size of any potentially usable dataset, particularly,

1) the limit imposed upon cache size so that it only resides in physical memory;

2) the number of time steps that can be fitted into the cache;

3) the effect of compression of the dataset on these limits;

4) the number of time steps that can be stored on the hard drive.

Table 1 summarizes the limits on single dataset sizes, assuming a maximum cache size of 1 GB and 100 GB of disk space. Although it is possible to increase the size of the cache to its maximum limit, it can be shown that a larger cache does not necessarily result in increased application performance [39,40]. An upper limit for the size of an individual time step can be set at 102 MB with the current hardware. Justification of this limit is provided by the prefetching routines, which require sufficient space in the cache to work successfully, with 10 cache segments being considered adequate. In addition, loading times of file sizes larger than 102 MB become significant. To meet the performance requirements of the application, wake encounter simulations need to be restricted to cases where the rate of change of the currently referenced time step is greater than the associated file loading time. The majority of civilian transport aircraft-wake encounters satisfy this restriction.

2. *Mesh Simplification Analysis*

The collapse of each edge during the mesh simplification stage results in the loss of a vertex containing wake vortex flow information. From the data presented in Table 2, it is evident that, as the local field error is increased beyond 0.1 m/s, there is no significant improvement in compression. It is shown that a point is reached where a relatively small gain in the number of collapsed edges introduces a much larger local field error. For the wake vortex flowfields in question, edge collapses that introduce a 5.0 m/s error into the flowfield are unacceptable, where such an error may be approximately 10% of the maximum velocity magnitude encountered.

3. *Data Access Performance*

Mesh simplification allows data point access times to be improved by allowing the creation of deeper trees for the same memory cost.

Table 1 Time-varying dataset limits imposed by the hardware used

Time-step file size, MB	Max. number of cached time steps	Max. number of time steps on disk	Time-step loading time, s
1	1024	102,400	0.025
8	128	12,800	0.2
20.5	50	5,000	0.51
102	10	1,000	2.55
512	2	200	12.5

Table 2 Effect of tolerated local field error ε_f, on mesh size, number of permitted edge collapse operations, and the resulting file size (mesh connectivity removed)

Local field error, m/s	Tetrahedra	Edges collapsed	File size, bytes
0.0	1,871,728	0	7,631,932
0.001	1,548,411	56,523	6,275,380
0.01	879,902	169,108	3,573,340
0.05	473,663	237,014	1,943,596
0.1	387,141	252,896	1,562,428
0.5	311,125	266,060	1,246,492
1.0	299,930	267,927	1,201,684
5.0	297,237	268,361	1,191,268

Figures 9a and 9b highlight the effect that tree depth has on data point retrieval performance and memory usage. It is clear that the deeper the tree, the shorter the point access time, although access times increase for very deep trees. However, before that point is reached, and as shown in Fig. 9b, deeper tree depths require significantly more memory but with minimal benefit in terms of search performance. For the current kd-trees, a good balance was achieved with a tree depth of 11.

The time to resolve the strip theory calculations becomes a significant factor as the number of searches increase, to the point where it becomes difficult to sustain the real-time iteration period. For example, using a tree depth of 11 resulted in average access times for individual data points of 2×10^{-5} s. If 100 strips are used to represent the aircraft, point searches absorb 2 ms of the 20 ms frame.

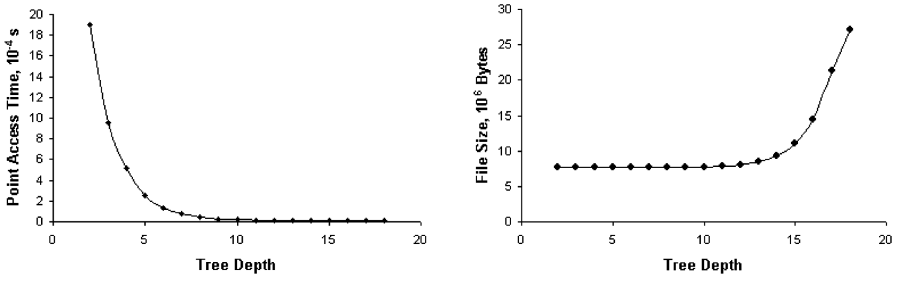
4. *Cache and Data Prefetching Analysis*

To assess the effectiveness of the prefetching mechanism, a comparison was made with a version of the software implemented with on-demand time-step loading, as shown in Figs. 10a and 10b. As expected, the on-demand version a) shows significant spikes in the iteration rate. These spikes coincide with the time when the encountering aircraft required access to a new time step (in this test, the encountering aircraft was catching up with the leading aircraft). The prefetching version b) of the software exhibits a steadier iteration rate. Mean iteration rates of 49.87 and 49.89 Hz, and variances of 14.38 and 5.47 were recorded for the on-demand and prefetching versions, respectively. It is clear that the performance of the multithreaded prefetching version of the software is significantly better than the on-demand version. The smaller spikes observed in Fig. 10b were caused by other periodic operating system kernel tasks. Clearly, the use of a real-time operating system would reduce these variations in performance.

5. *Simulated Wake Encounter*

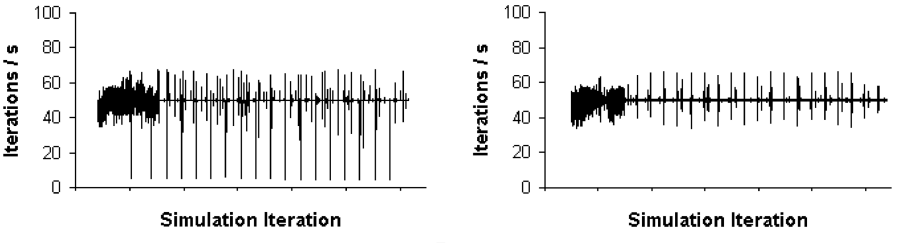
To assess the response of the LES based wake vortex model, a simulated flight was made through a wake vortex system. Two vortex models were used for comparison. One model was an analytic superposition of two Lamb–Oseen vortices, using a similar approach as described in some previous wake encounters [5–8], while a second model directly accessed the LES dataset. In both vortex models, the wake vortex generating aircraft was a Boeing 747-100, with an initial wake circulation of 565 m^2/s. The wake vortex system was initialized straight and level and the encountering aircraft (a Boeing 737 model) was set to intersect with the vortex system at a 15-deg angle in the horizontal plane, approximately 3.7 km (2 nm) behind the 747. Wake induced roll angles for each vortex model are presented in Fig. 11. Both models cause roll upsets of approximately 40 deg and each event lasts approximately 10 s. The roll angle history for each model is very similar for this instant in the life span of the wake vortex system. This is expected and shows that the strength and the form (parallel vortices) of the wake vortex modeled with LES has approximately the same rate of decay as an analytical wake vortex eddy-dissipation model [41].

The simulated flight illustrated in Fig. 11 highlights the similarities of the models during the stages of wake decay when the vortices remain parallel. In contrast, the high fidelity LES data model is intended to provide access to three-dimensional data, including wake instabilities beyond the parallel vortex phase of the wake. Further simulated flights were studied to confirm that the wake encountering aircraft was subjected to the forces created by the three-dimensional flow features after the wake instabilities developed. The dataset instant representing 150 s of wake decay (see Fig. 3c) was chosen for the flight tests. To emphasize the effect of the instabilities, an autopilot was used to hold the encountering aircraft near to the vortex cores, but sufficiently far away so that the autopilot could maintain a steady heading and altitude. In Fig. 12a the encountering aircraft was initialized to fly parallel to the wake and in the same horizontal plane as the two vortices. Additionally, the right wingtip was relatively close to the core of the left vortex. The recorded lateral accelerations



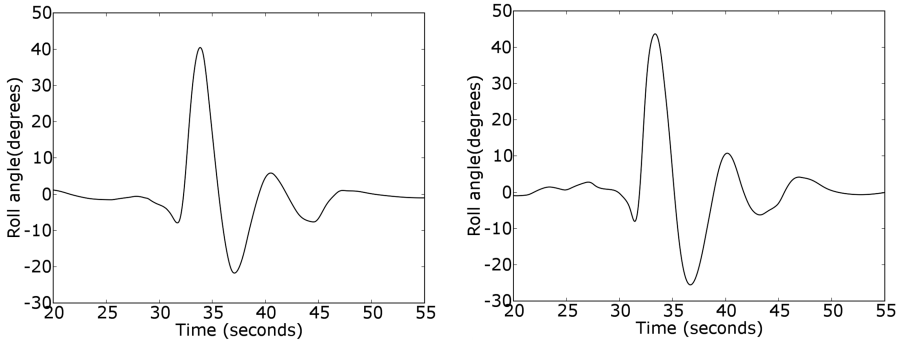
a) b)

Fig. 9 The relation of kd-tree depth against data point access time a) and the respective file size b).



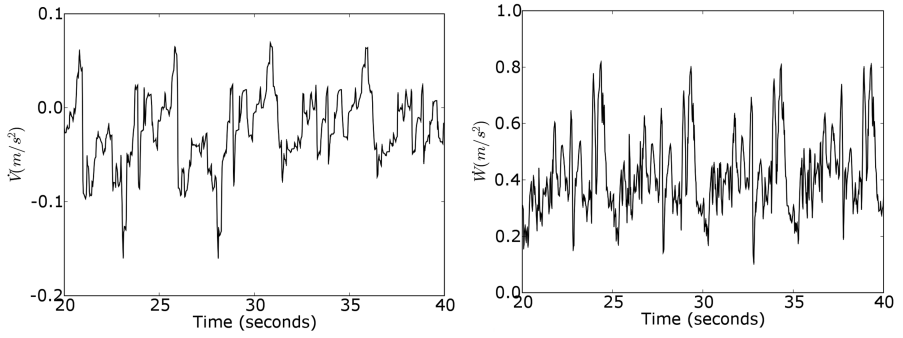
a) b)

Fig. 10 Comparison showing the effect of using *on-demand* time-step loading a) and using the time-step prefetcher b), on the iteration rate of the simulation.



a) b)

Fig. 11 History of recorded bank angle during the LES a) and the analytic b) wake vortex encounters, respectively.



a) b)

Fig. 12 History of accelerations recorded at the aircraft center of gravity during a simulated flight in the proximity of a wake vortex after 150 s of decay. The lateral accelerations a) and the vertical accelerations b) are shown, respectively.

at the aircraft center of gravity show cyclic behavior with a period of approximately 5 s. In Fig. 12b, vertical accelerations at the center of gravity are plotted while the aircraft was held centrally above the two vortex cores at a sufficient distance so that the autopilot could maintain altitude and heading. At this location one would expect the aircraft to experience periodic vertical acceleration changes because of the influence of the rising and falling nature of the vortices, as shown in Fig. 3c. These cyclic acceleration effects are clearly noticeable, again with an approximate period of 5 s, which coincides with the period that the encountering aircraft travels the length of an individual data tile. The noise observed on the plots in Fig. 12 is mainly attributed to the turbulent nature of the LES data at this point in the decay of the wake, although these effects may also be introduced by the data resolution. These results demonstrate the ability of the model to simulate flight through LES simulations of decaying wake vortices in real time. Such a tool should provide insight in any future analysis of flight through decaying wake vortices, especially in the investigation of reduced aircraft approach separation distances.

IV. Conclusions

The approach of using LES computational data as a basis for wake vortex models in a flight simulator has been demonstrated to be achievable and practical. Two main problems have been overcome: 1) simulating an accurate wake vortex decay using LES, and 2) integrating this large dataset in the flight simulator. The LES simulation was able to replicate short- and long-wave instabilities observed in real life and in other published results. Integrating this data successfully into a flight simulator depends on the number of data points, the number of aircraft strips, the spatial data organization method, and the available computational resources. The simulated flight test showed that the technique compared consistently with the analytic wake vortex model before the wake instabilities develop; however, detailed validation of the model has yet to be undertaken. Although the model involves a significantly more complicated software implementation than an analytic wake model, it offers significant benefits and potential for wake encounters in the unstable phase of the wake vortices and for future modification using different forms of datasets. For example, it would be possible to create and use new datasets simulating vortices generated by different aircraft or datasets accounting for atmospheric stratification and for ground effect. While the aircraft/wake interaction model is based on strip theory, the wake modeling has been substantially improved, using the LES data as a step towards a more realistic real-time model. This method of generating real-time vortices provides a useful tool for investigating simulated encounters with wake vortices and also a means to simulate encounters with other atmospheric phenomena that can be modeled with computational techniques.

Acknowledgment

This work is funded by the U.K. Engineering and Physical Science Research Council under Grant GR/R84047/01, and their support is gratefully acknowledged.

References

[1] "In-Flight Separation of Vertical Stabilizer, American Airlines Flight 587, Airbus Industrie A300-605R, N14053, Belle Harbor, New York, 12 Nov. 2001," National Transportation Safety Board, Aircraft Accident Report NTSB/AAR-04/04, Oct. 2004.

[2] Spalart, P. R., "Airplane Trailing Vortices," *Annual Review of Fluid Mechanics*, Vol. 30, Jan. 1998, pp. 107–138.

[3] Jacquin, L., "On Trailing Vortices: A Short Review," *International Journal of Heat and Fluid Flow*, Vol. 26, No. 6, 2005, pp. 843–854.

[4] Gerz, T., Holzäpfel, F., and Darracq, D., "Commercial Aircraft Wake Vortices," *Progress in Aerospace Sciences*, Vol. 38, No. 3, 2002, pp. 181–208.

[5] Sammonds, R. I., Stinnett, G. W., Jr., and Larson, W. E., "Criteria Relating Wake Vortex Encounter Hazard to Aircraft Response," *Journal of Aircraft*, Vol. 14, No. 10, 1977, pp. 981–987.

[6] Vicroy, D. D., and Nguyen, T., "A Numerical Simulation Study to

Develop an Acceptable Wake Encounter Boundary for a B737-100 Airplane," AIAA Paper 96-3372, July 1996.
[7] Stewart, E. C., "A Piloted Simulation Study of Wake Turbulence on Final Approach," AIAA Paper 98-4339, Aug. 1998.
[8] Sedin, Y. C.-J., Grasjo, I., Kullberg, E., and Larsson, R., "A Model for Simulation of Flight Passages Through Trailing Tip Vortices," ICAS Paper 2002-7.9.3, Sept. 2002.
[9] Crow, S. C., "Stability Theory for a Pair of Trailing Vortices," *AIAA Journal*, Vol. 8, No. 12, 1970, pp. 2172–2179.
[10] Loucel, R. E., and Crouch, J. D., "Flight-Simulator Study of Airplane Encounters with Perturbed Trailing Vortices," AIAA Paper 2004-1074, Jan. 2004.
[11] Proctor, F. H., Hamilton, D. W., Rutishauser, D. K., and Switzer, G. F., "Meteorology and Wake Vortex Influence on American Airlines FL-587 Accident," NASA TM-2004-213018, April 2004.
[12] Holzäpfel, F., Gerz, T., Frech, M., and Dörnbrack, A., "Wake Vortices in Convective Boundary Layer and Their Influence on Following Aircraft," *Journal of Aircraft*, Vol. 37, No. 6, 2000, pp. 1001–1007.
[13] Woodfield, A. A., and Tomlinson, B. N., "Ship Airwakes—A New Generic Model for Piloted Simulation," *AGARD FVP Symposium*, CP-577, Paper 10, May 1995.
[14] Polsky, S., and Naylor, S., "CVN Airwake Modeling and Integration: Initial Steps in the Creation and Implementation of a Virtual Burble for F-18 Carrier Landing Simulations," AIAA Paper 2005-6298, Aug. 2005.
[15] Advani, S. K., and Wilkinson, C. H., "Dynamic Interface Modelling and Simulation—A Unique Challenge," *Royal Aeronautical Society Conference*, The Challenge of Realistic Rotorcraft Simulation, Royal Aeronautical Society, London, U.K., Nov. 2001.
[16] Stumpf, E., "Numerical Study of Four-Vortex Aircraft Wakes and Layout of Corresponding High-Lift Configurations," AIAA Paper 2004-1067, Jan. 2004.
[17] Holzäpfel, F., Gerz, T., and Baumann, R., "The Turbulent Decay of Trailing Vortex Pairs in Stably Stratified Environments," *Aerospace Science and Technology*, Vol. 5, No. 2, 2001, pp. 95–108.
[18] Switzer, G. F., and Proctor, F. H., "Numerical Study of Wake Vortex Behavior in Turbulent Domains with Ambient Stratification," AIAA Paper 2000-0755, Jan. 2000.
[19] Moet, H., Darracq, D., Laporte, F., and Corjon, A., "Investigation of Ambient Turbulence Effects on Vortex Evolution using LES," AIAA Paper 2000-0756, Jan. 2000.
[20] Moet, H., "Simulation Numérique du Comportement des Tourbillons de Sillage dans l'Atmosphère," Ph.D. Dissertation, Institut National Polytechnique, Toulouse, France, 2003.
[21] Laporte, F., "Simulation Numérique Appliquée à la Caractérisation et aux Instabilités des Tourbillons de Sillage d'Avions de Transport," Ph.D. Dissertation, Institut National Polytechnique, Toulouse, France, 2002.
[22] Le Moigne, A., and Qin, N., "LES with Numerical Dissipation for Aircraft Wake Vortices," AIAA Paper 2006-1258, Jan. 2006.
[23] Boris, J. P., Grinstein, F. F., Oran, E. S., and Kolbe, R. L., "New Insights into Large Eddy Simulation," *Fluid Dynamics Research*, Vol. 10, No. 4–6, 1992, pp. 199–228.
[24] Leweke, T., and Williamson, C. H. K., "Cooperative Elliptic Instability of a Vortex Pair," *Journal of Fluid Mechanics*, Vol. 360, April 1998, pp. 85–119.
[25] Muraki, S., "Volume Data and Wavelet Transforms," *Computer Graphics and Applications*, Vol. 13, No. 4, July 1993, pp. 50–56.
[26] Cignoni, P., Costanza, D., Montani, C., Rocchini, C., and Scopigno, R., "Simplification of Tetrahedral Meshes with Accurate Error Evaluation," *Proceedings of IEEE Visualization 2000*, IEEE, Piscataway, NJ, 2000, pp. 85–92.
[27] Fuchs, H., Kedem, Z., and Naylor, B., "On Visible Surface Generation by A Priori Tree Structures," *Computer Graphics*, Vol. 14, No. 3, 1980, pp. 124–133.
[28] Samet, H., "The Quadtree and Related Hierarchical Data Structures," *ACM Computing Surveys*, Vol. 16, No. 2, 1984, pp. 187–260.
[29] Bentley, J. L., "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, Vol. 18, No. 9, 1975, pp. 509–517.
[30] Gaede, V., and Gunther, O., "Multidimensional Access Methods," *ACM Computing Surveys*, Vol. 30, No. 2, 1998, pp. 170–231.
[31] Spence, G. T., Allerton, D. J., Le Moigne, A., and Qin, N., "Real-Time Model of Wake Vortices Based on Large Eddy Simulation Datasets," AIAA Paper 2005-6205, Aug. 2005.
[32] Bentley, J. L., "Multidimensional Binary Search Trees in Database Applications," *IEEE Transactions on Software Engineering*, Vol. 5, No. 4, 1979, pp. 333–340.
[33] Cox, M., and Ellsworth, D., "Application-Controlled Demand Paging for Out-of-Core Visualization," *Proceedings of IEEE Visualization 1997*, IEEE, Piscataway, NJ, 1997, pp. 235–244.
[34] Ueng, S. K., Sikorski, C., and Ma, K. L., "Out-of-Core Streamline Visualization on Large Unstructured Meshes," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 3, No. 4, 1997, pp. 370–379.
[35] Lane, D. A., "UFAT—A Particle Tracer for Time-Dependent Flow Fields," *Proceedings of IEEE Visualization 1994*, IEEE, Piscataway, NJ, 1994, pp. 257–264.
[36] Silberschatz, A., Galvin, P. B., and Gagne, G., *Operating System Concepts*, 6th ed., Wiley, Hoboken, NJ, 2002.
[37] Allerton, D. A., "A Distributed Approach to the Design of a Real-time Engineering Flight Simulator," *Proceedings of the 21st ICAS Congress*, ICAS, Stockholm, Sept. 1998.
[38] Reimer, H. M., and Vicroy, D. D., "A Preliminary Study of a Wake Vortex Encounter Hazard Boundary for a B737-100 Airplane," NASA TM 110223, April 1996.
[39] Aliaga, D., Cohen, J., Wilson, A., Baker, E., Zhang, H., Erikson, C., Hoff, K., Hudson, T., Stuerzlinger, W., Bastos, R., Whitton, M., Brooks, F., and Manocha, D., "MMR: An Interactive Massive Model Rendering System Using Geometric And Image-Based Acceleration," *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, Association for Computing Machinery, New York, 1999, pp. 199–206.
[40] Wilson, A., Manocha, D., and Lin, M. C., "Representation and Interactive Manipulation of Massive CAD Databases," *Lecture Notes in Computer Science*, No. 1737, Springer, Berlin/Heidelberg, 1999, pp. 268–285.
[41] Sarpkaya, T., Robins, R. E., and Delisi, D. P., "Wake-Vortex Eddy-Dissipation Model Predictions Compared with Observations," *Journal of Aircraft*, Vol. 38, No. 4, 2001, pp. 687–692.