Improvements of Automation for Mesh Generation of a Canadian CFD Capability for Store Release Predictions

F. Fortin, T. Zoric and A. Benmeddour
Aerodynamics Laboratory, Institute for Aerospace Research (IAR)
National Research Council (NRC), Ottawa
Ontario, Canada, K1A 0R6

Email:francois.fortin@nrc-cnrc.gc.ca

ABSTRACT

Following a request by the Canadian Department of National Defence, the Institute for Aerospace Research of the National Research Council of Canada has developed a Computational Fluid Dynamics quasi-steady approach for store release prediction. Upon completion, this software was transferred to industry (Bombardier Aerospace) for use by Store Release Engineers, and, during evaluation, improvements were suggested. These included the elimination of user intervention during the computations, which arose due to problems with the mesh motion module of the software. To satisfy this requirement, automation of the mesh generation was seen as a major requirement. While implementing this automation, a Graphical User Interface was also developed, which extended the software's capability by allowing interactive definition of the different aircraft loadings in carriage position. Following the loading description, script procedures call the mesh generation software to obtain an initial mesh, eliminating the necessity to use the mesh generation software interactively, through its GUI interface. Automated remeshing can then be carried out with the stores at any position, either in carriage or at some position throughout the trajectory. In addition to automatic mesh generation, further improvements include the automatic computation of the intersection curves between the objects loaded on the A/C.

1.0 INTRODUCTION

The safe separation envelope of a store released from an aircraft is crucial in the store certification process [1]. In order to predict store trajectories, accurate predictions of the aerodynamic loads acting on the stores are required. For most store release configurations, such predictions are required for the first 0.5 seconds of the trajectory, when the store is in the interference flow field generated by the parent aircraft and there remains a risk of the aircraft/store collision. Simulating such problems continues to be challenging, as the aircraft geometries are becoming more complex and more stores are carried simultaneously, increasing mutual interference. The safe separation region must also cover larger areas of the flight envelope of the releasing aircraft and higher release velocities.

Two traditional approaches to store clearance problems is to carry out wind tunnel and flight tests [2]. The requirement to obtain sufficient data, for the certification of a given store/aircraft configuration, results in many flight and wind tunnel tests. Due to time constraints, shrinking budgets and rising wind tunnel costs, Computational Fluid Dynamics (CFD) has become a complementary tool to help the stores release engineers fulfil their certification processes in a reasonable time and within a reasonable budget [3-4].

Paper presented at the RTO AVT Symposium on "Functional and Mechanical Integration of Weapons and Land and Air Vehicles", held in Williamsburg, VA, USA, 7-9 June 2004, and published in RTO-MP-AVT-108.

Following an initiative prompted, in the main, by the Canadian Department of National Defence (DND), in 1994, the Institute for Aerospace Research (IAR), at the National Research Council Canada (NRC), undertook the initiative to develop a Canadian CFD capability for the numerical simulation of external stores carriage and release. As a result, an Euler quasi-steady CFD approach to predict the trajectories of stores released from the CF-18 aircraft [5] was developed. This approach consists of coupling, in a quasi-steady mode, a 3D unstructured Euler solver, a 6-DOF Store Separation Model and an unstructured mesh motion module. The developed CFD approach has been validated through participation in international efforts organized to demonstrate the capability of CFD methods as an integral component to the overall store certification process [5-6].

Following evaluation of this procedure by Bombardier Aerospace Store Release Engineers (SRE) for a flight-tested CF-18/stores configuration, some shortcomings of this technique were highlighted [7]. A suggestion by the SRE's was that the mesh generation process should be simplified and that global remeshing, when required, should be performed automatically. It was considered that this would provide a major improvement in the efficiency of running the software by reducing requirements for mesh generation process around complex configurations, which have always been and remain the bottleneck in any CFD computations. Commercial mesh generation software, even with the improvements in the graphical user interface, still require a lot of user expertise to be used efficiently. Eliminating the necessity for the SRE's to become proficient in mesh generation would allow them to concentrate fully on the physics of the stores clearance.

A procedure for automated generation of an initial mesh was therefore required. For the specific application of store release, the mesh generation process around an aircraft carrying different stores is a repetitive process, involving repositioning of various stores at different displacements and attitudes. This is done without the need to change store geometry through additional CAD effort. With unstructured mesh discretisation, as used in the NRC/IAR quasi-steady approach, the implementation of an automated procedure was relatively easy, as no topological constraints (bloc topologies) exist. To help the user, a Graphical User Interface (GUI) software was developed, which allows interactive positioning of the various stores at their respective positions and orientations. Once the loading is specified, execution of a single script file, calling the mesh generation software, automatically generates a mesh for the specified stores positions.

This automation was added to the store trajectory computations. In order to take into account the motion of the released store and its surrounding mesh, IAR selected an unstructured moving mesh approach. Due to the appearance of negative volume elements as the trajectory progresses, however, this approach sometimes fails and requires user intervention to remesh the whole domain. This could take anywhere between four to eight hours depending on the user's experience and know-how. If the mesh failure happened overnight, the trajectory calculation would not carry on until the next day. To remedy all these problems, a complete global remeshing is carried out automatically, using the same approach described previously to generate an initial mesh, without invoking the GUI. The store positions are extracted automatically from the output files describing the store trajectory.

This paper describes the aspects of the automated remeshing procedures, as well as some part of the GUI developed to specify the loading of a releasing aircraft. An extension of the method to automatically evaluate intersection of different objects is also presented. This would be required to handle the cases where stores are in direct contact with the releasing aircraft, such as the AIM-7, which is carried in semi-recessed position.

2.0 NRC/IAR QUASI-STEADY STEADY CFD APPROACH

The NRC/IAR quasi-steady CFD approach assumes that the steady state flow fields computed at the various positions and orientations of the store can provide a reasonable estimate of the store's aerodynamic loads. There is thus no time history taken into account for the flow field. This is an acceptable approximation, as the store release is a low frequency process. The approach consists of three modules, which can be used separately. It is summarised in the following steps:

A) For a given store position, compute the steady inviscid flow field solution with the inviscid flow solver (**FJ3SOLV**). Output the store aerodynamic coefficients in global axes.

B) Provide as input the store aerodynamic forces and moments to the six-degrees-of freedom Store Separation Model code (**6-DOF SSM**). Transfer the loads in store axes. Compute the new store CG position and orientation (Euler angles) for one time step Δt.

C) Move the store to its new position with the surrounding mesh, using a spring analogy technique (**MESH MOTION**).

These three steps are repeated until the store moves sufficiently far from the releasing aircraft or the store hits the aircraft. For subsequent store positions (back to step A), the previous numerical solution is used as an estimate. Occasionally, however, the mesh cannot distort to the next trajectory point, while maintaining positive volume elements throughout the computational domain. This is due to the basic mesh motion algorithm, which assumes that the edges connecting the vertices of the mesh are equivalent to a series of springs of a given stiffness. The mesh motion is carried out by minimizing the potential energy of the system. This does not prevent the overlapping of the edges, which creates negative volume elements. As the steady-state flow solver **FJ3SOLV** is unable to run with negative volume elements, global remeshing is then required, resulting in the penalties previously discussed.

The details of the quasi-steady approach have already been presented in [5]. Only a review is presented here. The unstructured inviscid flow solver, **FJ3SOLV,** is based on a finite volume formulation. The convective fluxes are computed using Jameson's cell-centered formulation with the standard explicit addition of second and fourth order artificial viscosity [8]. The steady-state solution is obtained using an explicit 4-stage scheme. Standard acceleration techniques, such as local time stepping, implicit residual smoothing and enthalpy damping, are used to speed up the convergence of the scheme.

The **6-DOF SSM** store separation model is a simplified version of a code developed by Bombardier [9]. The trajectory of the store is computed using the standard equations of motion, while the releasing aircraft is considered to be a single rigid point, without elastic deformation of any part of its structure. The ejector forces are imparted to the store, by assuming that there is a single point of application and continued contact between the ejector pistons and the surface of the store. As the approach used is quasi-steady, the effects of the aerodynamic damping coefficients (Clp, Cmq and Cnr) are added to the total store aerodynamic moments using estimated theoretical values. A time step of 0.02 sec. was found to be adequate [6] to provide a good engineering accuracy for the trajectory predictions.

3.0 GLOBAL REMESHING AUTOMATION

The global remeshing automation consists to automatically remesh a set of different objects after positioning them at different positions and orientations. No change in the CAD representation of the individual objects is necessary. In other words, the initial spatial representation of the various objects are brought to another final position and orientation, with use of the basic transformation matrices of translation and rotation. By bounding the domain with an outer shell representing the far field extent, a 3D computational domain can therefore be meshed.

The transformations can be applied either on the surface meshes representing the various objects, or on their surface CAD representations. The first option would imply that the mesh vertices on the surfaces would be exactly recovered, during the volume mesh generation process. The mesh generation software will usually move the surface vertices in order to obtain a better quality mesh. If the CAD representation is not available, some of the characteristics of the surfaces may be lost.

The CAD representations of objects in ICEM CFD, a mesh generation software used at IAR, are output in files called TETIN (TETra INput). They supplement the CAD representations through NURBS (Non-Uniform Rational Bi-Splines) and the mesh settings, which define the resolution of the surface meshes of the objects. These TETIN files can be obtained from either the DDN (Design Drafting and Numerics) meshing interface facility (DDN-TETIN) or from an appropriate ICEM-CAD interface for geometries generated with CAD software other than DDN.

To automate the remeshing, independent TETIN files were concatenated into a single TETIN file, representing specific aircraft loadings (whole configuration). This final resulting TETIN file can be meshed with ICEM CFD. This was originally accomplished through the GUI interface of ICEM CFD. However, in the automated version, these are performed using script files run by ICEM CFD in batch mode. These script files can call all the functions of ICEM CFD that would be otherwise executed interactively. Other commands can be added in these script files using TCL (Tool Command Language), which improve the capabilities and the portability of these files. Parts of this technique have been published in reference [10]. The whole configuration is saved back to a database and can be reused as a starting point for a new loading of the A/C.

ICEM CFD uses an octree technique to progressively refine the mesh by taking into account the mesh settings defined for the various surface meshes of the objects. As the effects of these settings depend on the relative positions of the adjacent objects, ICEM CFD will vary the refinement for different object positions. The surface meshes on the independent objects will thus be slightly different depending on the relative positions of the objects. However, the vertices projected on the surface will always be exactly projected on the surfaces due to the presence of the CAD representation of the objects.

The positioning of the various objects is initially specified in ASCII file format. Once defined, the script file would then generate a mesh for the store in carriage position. The store trajectory computation could then be started. However, failure of the **MESH MOTION** module still remained, resulting in a stopping of the process. As the position of all the moving objects are known in some reference axis system, the remeshing script file can then be reused by reading this information to regenerate a new 3D computational mesh. Once done, the quasi-steady approach continues the computations. The integration of a global remeshing procedure, within the trajectory calculation modules, eliminated the user intervention. This is only required at the beginning (carriage position definition) and end of the computations, in order to post-process and analyze the data.

3.1 Global remeshing application

The global remeshing procedure was applied to the generic wing/pylon/finned store configuration [11]. This test case consists of a clipped delta wing with a 45° leading edge sweep angle carrying a generic finned-store with an ogive-cylinder-ogive body shape with four fins. The free stream conditions are given in Table 1 while Figure 1 represents the geometry with the store in carriage position showing iso-Mach contours. Figure 2 shows the computed store orientations (Euler angles) and positions compared with experimental values (EXP) while Figure 3 presents a comparison of the computed aerodynamic coefficients acting on the store, in stores axis, with the experimental values (EXP). The computations were carried out for 13 time steps on a SGI ORIGIN 2000. Four appearances of negative volume elements, resulting from **MESH MOTION**, were fixed automatically with the automated remeshing approach. A reasonably good agreement was obtained without any user intervention during the trajectory computation process

	W-P/FS
Mach	0.95
AOA (deg)	0.0
Dive angle(deg)	0.0
Pressure altitude(ft)	26000.0

Table 1. Freestream release conditions

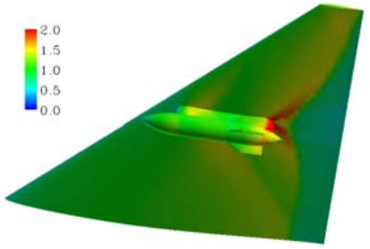


Figure 1: Wing/pylon/finned store with iso-Mach contours

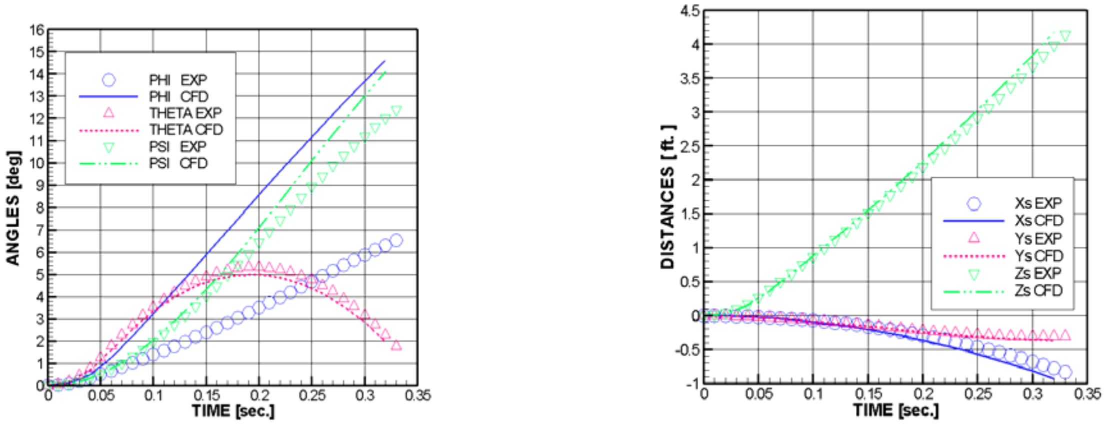


Figure 2: Store CG attitudes

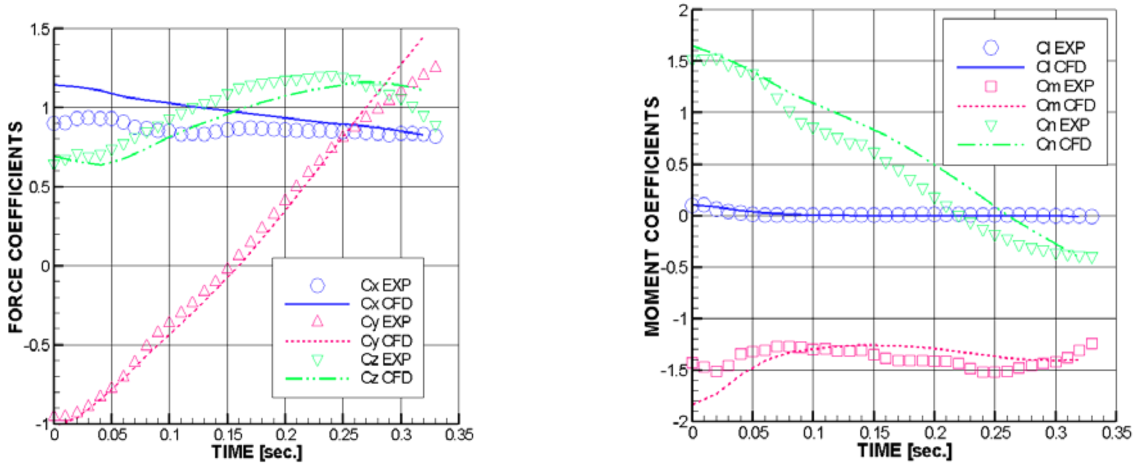


Figure 3: Store aerodynamic loads

4.0 GUI DESCRIPTION

The object positioning process has been simplified for efficient use of the quasi-steady approach as an engineering tool. The initial procedure, which consisted of input global object positions in ASCII files, was too cumbersome and error prone. These files are currently created through the GUI interface. This interface allows interactive positioning of the objects and, at the same time, provides a 3D visual representation of the resulting configuration. The 3D display provides a visual check of objects positioning. The various objects (stores, pylons and fuel tanks) that can be carried by the releasing aircraft are accessible through a dynamic database and allow interactive building of all the possible loadings on a given aircraft. Positioning of the objects is carried out through a sequence of mouse and keyboard operations with visual references. The final loading file is then included in a script file, which runs ICEM CFD to get an initial mesh with the stores in carriage position.

The environment chosen for the GUI is Linux. The platform chosen is a high-end PC, which offers a very cost effective alternative to the SGI systems. In order to ease the coupling of the GUI with the CFD codes, Fortran 90, with dynamic memory allocation, was selected. The tool chosen for the creation of windows, dialogs and menus, is Winteracter [http://www.winteracter.com/]. The creation of these basic graphic functions is done in a simple way, through the use of a series of calls to a set of basic functions. For successful interactivity of the interface, it was also necessary to enable the user to select, using the mouse, objects on the screen. To enable this, Winteracter provides the API OpenGL, which has become a standard in the graphics industry. In the following paragraphs and for simplicity, the keywords, dialog names, as well as the options available used by the GUI, will be highlighted in **Bold**.

Once started, the GUI consists of three different windows that are always present during the various operations (Figure 4). The top (**Menu Window**) and the bottom (**Info)** windows are for selection and generic information, while the middle window (**OpenGL Graphics Window**) provides a graphic display. **Menu Window** provides a list of the various basic menus available. **OpenGL Graphics Window** displays, using OpenGL**,** the 3D views of the various objects representing the aircraft loading. The objects can be interactively selected, using the mouse inside this window, minimizing the need to use the keyboard. Different options are available to help remove ambiguity or confusion during the selection process. These include various camera positions as well as colouring of the objects. The last window (**Info)** displays messages describing the actions expected by the user as well as the actual process being carried out (object selection, translation, rotation, etc).

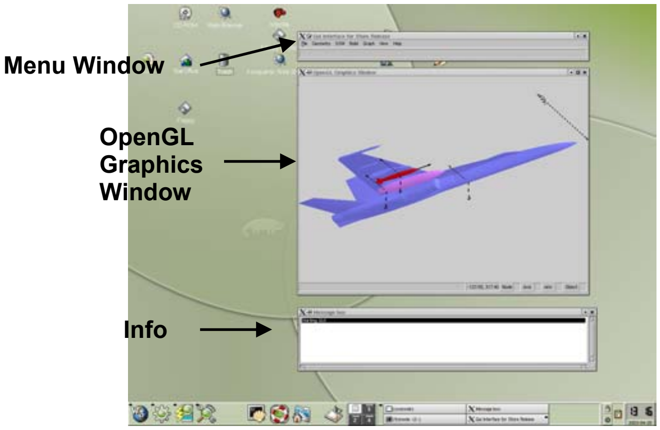


Figure 4: Windows of the GUI

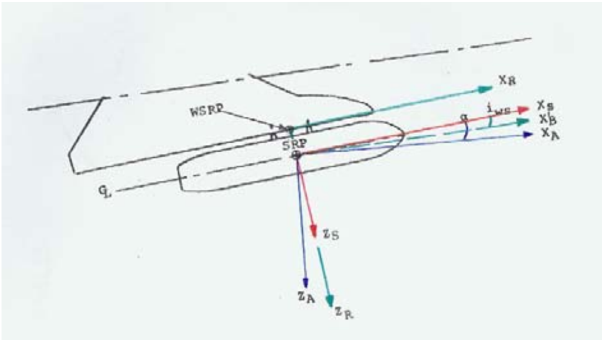


Figure 5: Reference points and axes systems [1]

4.1 Entities manipulation

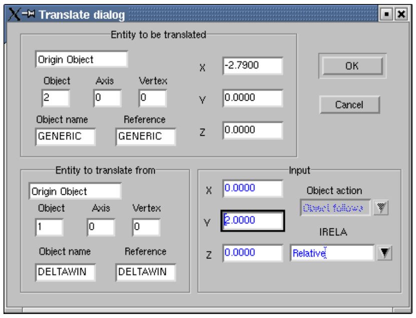
The main reason for the GUI development was to help in the object positioning process. If the object's final position and orientation are known in a global axis system, the confusion is minimized. However, the objects position are often defined relative to an already existing point or set of axes. This is the case, for example, of stores carried on Vertical Ejection Rack (VER) (Figure 5). In this case, the store reference point (SRP) is defined relative to the Weapon Store Release Point (WSRP), in an axis system that is fixed to the VER (Xr,Zr). This set of axes is pitched down 3 degrees relative to the global axis system of the aircraft. Evaluation of the final attitude in the global coordinates is therefore a possible source of error. The visual aids provided help to reduce the uncertainty in the selection of the attitudes.

The GUI facilitates the object manipulation by using three basic entity types. These are the **Vertex**, **Axis** and **Object** type. The **Axis** type contains the **Vertex** type, while the **Object** type contains both the **Vertex** type and the **Axis** type. The entities **Vertex** and **Axis** are used to provide some reference point in space and/or on the geometries, which helps in the positioning of the **Object**s. All these entity types have options to help minimize confusion in the selection process.

A **Vertex** is defined only by its position. Its position can be either in a global axis system (**Free vertex**) or relative to an **Axis** system or **Object**. An **Axis** type is defined by its position and its orientation that can be either global (**Free Axis**) or local when defined relative to an **Object**. An **Object** type is defined by its position and orientation in global coordinates. A main **Axis** is used to visually represent the position of the **Object**. A set of local **Axes** can be connected to the **Object** to allow relative positioning. The position and orientation of these local **Axes** are given, relative to the main **Axis** of the **Object**. The different local **Axes** could correspond to the various Weapon Store Release Points (WSRP) of the various stations, as shown on Figure 5, axis system Xr,Zr.

Manipulation of the various entity types is carried out by first selecting what is the entity to be moved and then the entity to move from. Object positioning is then carried out in the **Axis** system of the entity to move from. Two dialogs are used to implement either pure translation (**Translate dialog**), or pure rotation (**Rotate dialog**).

Translate dialog (Figure 6) applies pure translations from an entity type (1) that is the **Entity to be translated**, relative to another entity type (2) that is the **Entity to translate from**. The translation can be either global in the **Axis** system of entity 2, or in relative displacement with respect to entity 2 from entity 1, or from entity 2 (**IRELA)**. When a local **Axis** of an **Object** is being moved, the **Object action** option specifies if the **Object** will follow the translation or whether the **Object** will stay fixed in space. In this case, the relative attitude of this local **Axis** is recomputed from the main **Axis** of the **Object**.



Entity to be translated

Origin Object

Object 2 Axis 0 Vertex 0

Object name GENERIC Reference GENERIC

X -2.7900
Y 0.0000
Z 0.0000

Entity to translate from

Origin Object

Object 1 Axis 0 Vertex 0

Object name DELTAWIN Reference DELTAWIN

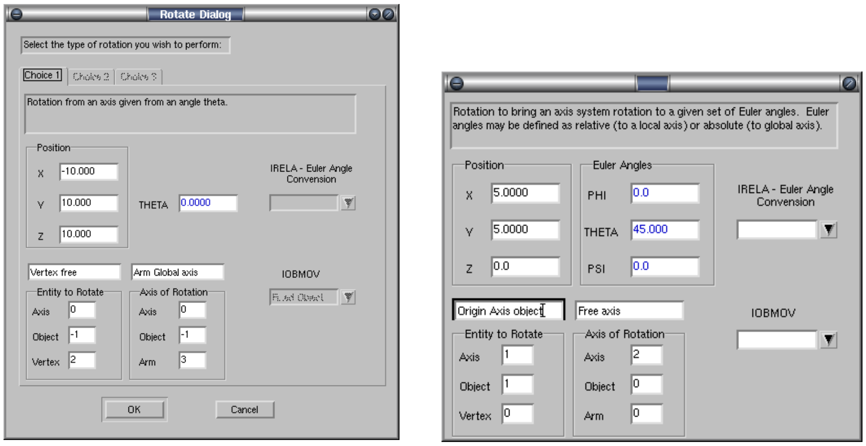
Input

X 0.0000
Y 2.0000
Z 0.0000

Object action Object follows

IRELA Relative

Figure 6: Translate dialog



Select the type of rotation you wish to perform:

Rotation from an axis given from an angle theta.

Position

X -10.000
Y 10.000
Z 10.000

THETA 0.0000

IRELA - Euler Angle Conversion

Vertex free

Arm Global axis

Entity to Rotate

Axis 0
Object -1
Vertex 2

Axis of Rotation

Axis 0
Object -1
Arm 3

IOBMOV Fixed Object

Rotation to bring an axis system rotation to a given set of Euler angles. Euler angles may be defined as relative (to a local axis) or absolute (to global axis).

Position

X 5.0000
Y 5.0000
Z 0.0

Euler Angles

PHI 0.0
THETA 45.000
PSI 0.0

IRELA - Euler Angle Conversion

Origin Axis object

Free axis

IOBMOV

Entity to Rotate

Axis 1
Object 1
Vertex 0

Axis of Rotation

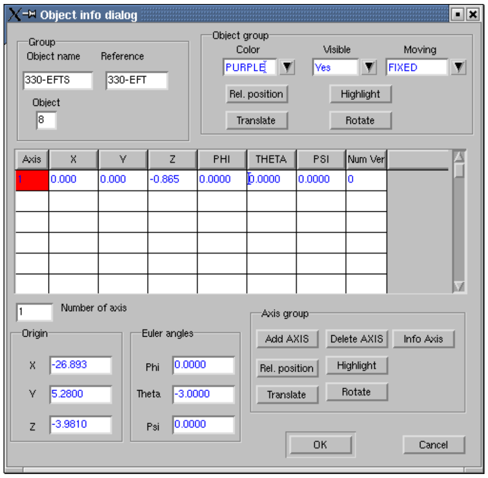
Axis 2
Object 0
Arm 0

Figure 7: Rotate dialog a) Rotation around arm of an axis,
b) Rotation to a specified orientation.

Figures 7a and 7b represent the **Rotate dialog.** The rotations are defined for an **Axis** or an **Object.** Figure 7a is the dialog for a rotation around the arm of an **Axis** of the second entity while Figure 7b allows positioning by specifying the Euler angles around the global **Axis** system, or relative to the local **Axis** system of the second entity. A last possibility (not presented) involves taking the orientation of the second entity. As in the **Translate** dialog, when the first entity is the local **Axis** of an **Object**, the user needs to decide whether the **Object** rotates or stays fixed, using the **Object action**.

Access to a specific entity type (**Vertex**, **Axis** or **Object**) is allowed to change their characteristics. It is possible to modify their positions (**Translate** or **Rotate),** find their **Relative positioning, Highlight** them to easily locate them on the screen, as well as to **Add** or **Delete** sub-type entities. For example, when in the **Axis** menu, it is possible to **Add** or **Delete** vertices. As soon as a modification is done inside the dialog, the resulting effects are updated inside the **OpenGL Graphics Window.** This combination of analog and digital representations of the effects of the actions is crucial for friendly interaction.

Figure 8 represents the **Object info dialog.** At this level**,** the basic information, such as position and orientation, are available and can be changed. The list of the sub type entity connected to the basic entity, **Axis**, is then provided. If a modification on one of the local **Axes** is desired, the **Axis Info Dialog** is then displayed (Figure 9). The basic information on the **Axis** is then provided, as well as a list of its sub-type entity (**Vertex**).



Object name: 330-EFTS
Reference: 330-EFT
Object: 8
Color: PURPLE
Visible: Yes
Moving: FIXED

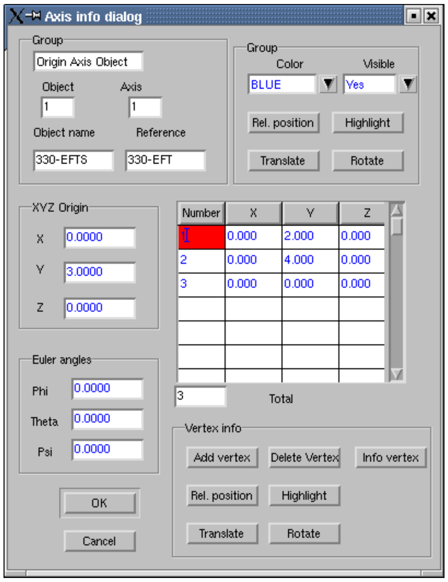
Axis	X	Y	Z	PHI	THETA	PSI	Num Ver
1	0.000	0.000	-0.865	0.0000	0.0000	0.0000	0

Number of axis: 1

Origin
X: -26.893
Y: 5.2800
Z: -3.9810

Euler angles
Phi: 0.0000
Theta: -3.0000
Psi: 0.0000

Figure 8: Object info dialog



Group: Origin Axis Object
Object: 1
Axis: 1
Object name: 330-EFTS
Reference: 330-EFT
Color: BLUE
Visible: Yes

XYZ Origin
X: 0.0000
Y: 3.0000
Z: 0.0000

Euler angles
Phi: 0.0000
Theta: 0.0000
Psi: 0.0000

Number	X	Y	Z
1	0.000	2.000	0.000
2	0.000	4.000	0.000
3	0.000	0.000	0.000

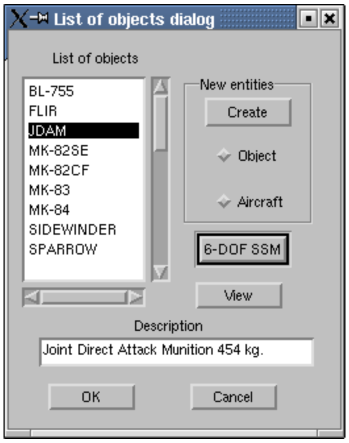
Total: 3

Figure 9: Axis info dialog

4.2 Aircraft loading dialogs

A specific aircraft loading is started by invoking the **Aircraft** environment from the **Geometry** menu on **Menu Window.** Two dialogs are available to assist the user. One of them, **List of objects dialog** (Figure 10)**,** gives access to the database of the various **Objects** that can be used as building blocks for the definition of a new aircraft loading. The database consists basically of the various loads, stores, pylons and fuel tanks that can be carried by the releasing aircraft. A **Description** box briefly describes the highlighted selection entity of the database.

The database is dynamic, i.e. it can be increased when needed through the option **Create.** The creation of a new object opens the **Store characteristics dialog** (Figure 11)**.** Here, the **Object** characteristics required for integration in the **6-DOF SSM** of a rigid body (characteristic length **Diameter,** reference area **Sref, Mass, Inertia** matrix and damping derivative coefficients **Clp, Cmq** and **Cnr)** are provided. It is also possible to set the object to a **Moving** or a **Fixed Object**. A **View** option opens another window in Winteracter format, which allows viewing the object in 3D. This 3D view, which uses Winteracter tool, is more limited than OpenGL. When an object of a given type appears more than once, it is possible to give it a different name (**Store name**), in order to separate it from the other objects. The generic name of the object (**Reference**) must, however, be kept. This renaming is required, for example, when defining a CF-18 with a load of four MK-82 stores. The flow solver must know what are the respective surface meshes of the released objects in order to compute the local loads acting on one specific store. During the concatenation of the various TETIN files, the family names of the objects appearing more than once will also be modified.

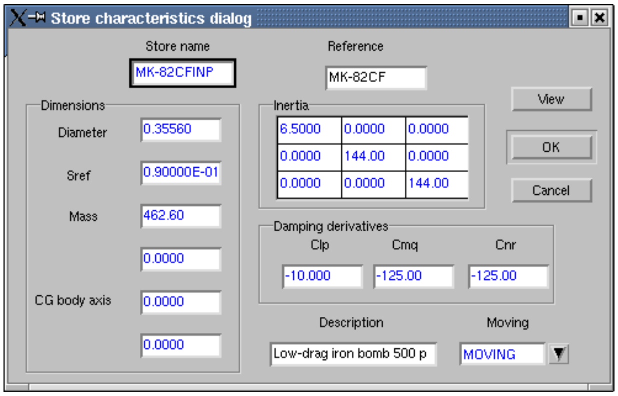


List of objects

- BL-755
- FLIR
- JDAM
- MK-82SE
- MK-82CF
- MK-83
- MK-84
- SIDEWINDER
- SPARROW

Description: Joint Direct Attack Munition 454 kg.

Figure 10: List of objects dialog



Store name: MK-82CFINP

Reference: MK-82CF

Dimensions

Diameter: 0.35560

Sref: 0.90000E-01

Mass: 462.60

0.0000

CG body axis: 0.0000

0.0000

Inertia

6.5000	0.0000	0.0000
0.0000	144.00	0.0000
0.0000	0.0000	144.00

Damping derivatives

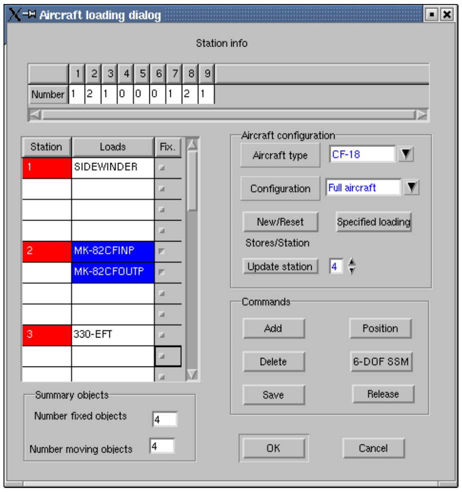
Clp	Cmq	Cnr
-10.000	-125.00	-125.00

Description: Low-drag iron bomb 500 p

Moving: MOVING

Figure 11: Store characteristics dialog

The second dialog is called **Aircraft loading dialog** (Figure 12). At the beginning, no loading is defined. Unless an existing loading is provided (**Specified Loading**), the loading must be specified. The **Aircraft Type** can be selected, as well as the **Configuration** of the aircraft (full aircraft, port or starboard side). Using the mouse, the user can sequentially select the **Object** to be **Add**ed to the various stations of the aircraft from the **List of Objects dialog**. As soon as an **Object** is added, a faceted representation of the object is immediately drawn in the **OpenGL Graphics Window**. An **Object** can be displayed either in **Solid** or in **Wire frame** mode. A push-button allows the user to select whether the added **Object** is a fixed or a moving **Object**. As the **Object** is selected from the generic database, its characteristics for the **6DOF SSM,** as well as its **Position,** can be modified. The resulting loading can be **Save**d for later work. It is also possible to **Delete** some loads from a previously defined loading and then **Add** the new components.



Station info

	1	2	3	4	5	6	7	8	9
Number	1	2	1	0	0	0	1	2	1

Station	Loads	Fix.
1	SIDEWINDER	
2	MK-82CFINP	
	MK-82CFOUTP	
3	330-EFT	

Aircraft type: CF-18

Configuration: Full aircraft

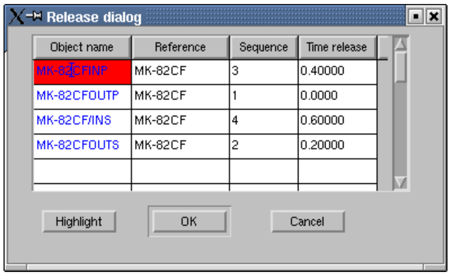
Stores/Station: 4

Summary objects

Number fixed objects: 4

Number moving objects: 4

Figure 12: Aircraft loading dialog



Object name	Reference	Sequence	Time release
MK-82CFINP	MK-82CF	3	0.40000
MK-82CFOUTP	MK-82CF	1	0.0000
MK-82CF/INS	MK-82CF	4	0.60000
MK-82CFOUTS	MK-82CF	2	0.20000

Figure 13: Release dialog

After loading definition, the release **Sequence** and **Time release** of the dropped stores can be specified by calling the **Release dialog** (Figure 13), using the **Release** button of the **Aircraft loading dialog**. This information will be used in the **6-DOF SSM** code.

4.3 Loading example

Some loadings, specified on the CF-18 aircraft are presented in Figure 14. The various objects used are the CF-18 with pylons, a 330 EFT (External Fuel Tank) and a MK-83 general-purpose bomb. In all cases, the various positioning were straightforward.

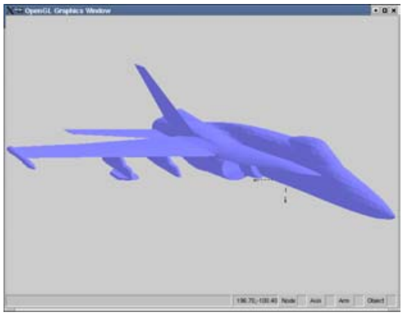


Figure 14a): CF-18 configuration,

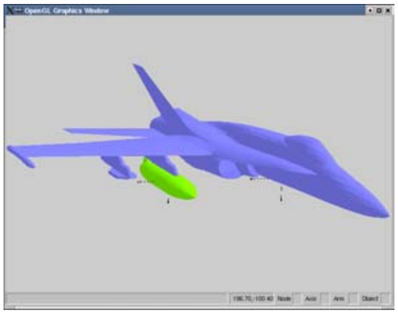


Figure 14b): CF-18 + 330 EFT,

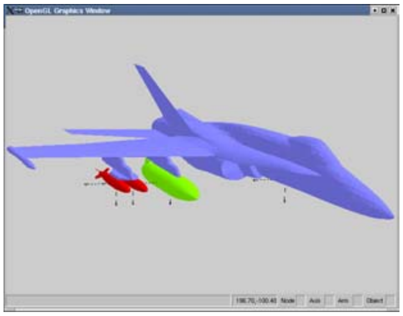


Figure 14c) :CF18 + 330 EFT + 2 MK-83

5.0 INTERSECTION PROCESS

The capabilities of the global remeshing process have been extended to handle the cases where different objects come into direct contact. This would be the case for stores carried in semi-recessed position under the belly of the fuselage, for example the AIM-7 missile and FLIR pods. It would be possible to augment a clean CF-18 with objects like the wing pylons, the VER, MK-82, MK-83 and the MK-84 stores. When required, these positioned objects would have their intersections computed automatically. This approach could also be used in conceptual aircraft design, for optimisation of the basic shapes.

The extension has been added in the script file. The intersection sequence is made of a series of basic steps. The first step compares the various families in the TETIN files to check for the existence of intersections. The families correspond to the symbolic names representing the surfaces in the TETIN files. The comparison must be made between the families of one basic TETIN file with the families of all the other TETIN files.

For each family's pair, ICEM CFD creates an intersection curve. If the two families are too far apart, so that no intersection exists, ICEM CFD generates an error message. If this process fails, another family's pair is selected for evaluation. If there is no error message, an intersection is found and a new intersection curve is created. In order to adequately represent the new intersection curve in the meshing process, the element sizes of both families are compared and the smallest value is assigned to the curve.

During the procedure's test phase, it was found that adding points on the intersection curve was required. This is due to the fact that, if the intersection curve deviates by large angles, the absence of points will result in a poor mesh representation at the interface. Therefore, an ICEM CFD procedure was called to add extra points in order to improve the mesh quality and better capture the angles.

Figures 15 to 20 represent a simplified generic geometry demonstrating the concept. The geometry is made of eight components: a cone for the nose, a seat, a lower and upper canopy, a rear fuselage, a left and right wing as well as a vertical fin. Figure 15a shows the various objects as an exploded view with Figure 15b giving the full assembly. Intersections of the wings and the fin with the fuselage were found automatically without difficulty. As the upper canopy, the lower canopy, the nose and the rear fuselage are in direct contact, the space bounded by these components was not meshed. This included the seat, which is hidden. The mesh can be seen on Figure 16a, which represents the surface mesh on the nose and the lower canopy. Note that the base of the cone as well as the upper plane of the lower canopy is not meshed. Figure 16b gives the full cone canopy assembly and clearly shows a bounded empty volume.

Figures 17 and 18 demonstrate the ease of building more complex geometries from basic shapes. On figure 17, two vertical fins were added at 135 degrees to the left and right of the vertical fin. Figure 18 is the same geometry than figure 17, but with the addition of another left and right wing at 90 degrees.

Figure 19 represents the fully assembled geometry, but with the seat positioned at different locations. For clarity, the upper canopy was not added. This demonstrates that it would be very easy to simulate, in a quasi-steady mode, the ejection of a pilot. Figure 20 represents the fully assembled geometry with a single ejection seat and the upper canopy positioned at different locations. The upper canopy collides with the vertical fin. The intersection, after collision, was still automatically computed.

6.0 CONCLUSIONS

The quasi-steady approach has been fully automated. User intervention is only required in the definition of the test-case (loading and release conditions). The problem definition is carried out through a GUI approach, which gives a visual confirmation of correct positioning of objects. This approach is valid for an inviscid tetrahedral mesh. For meshes suitable for Navier-Stokes computations, the technology (the mesh generation software) is still not mature enough to make this process fully automatic. In the near future, however, it is conceivable that this approach could be extended to viscous simulations.

Although the remeshing was carried out using TETIN files in a format specific to ICEM CFD for the present study, the technique can be extended to other format representations. The only requirements would be for the mesh generation software's capability to receive, as input, the resulting file containing all the concatenated objects and be able to run in batch mode.

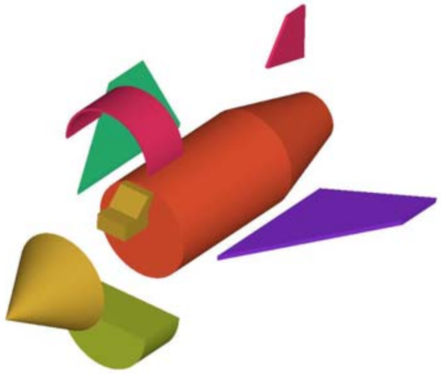


Figure 15a: Breakdown of aircraft 8 components

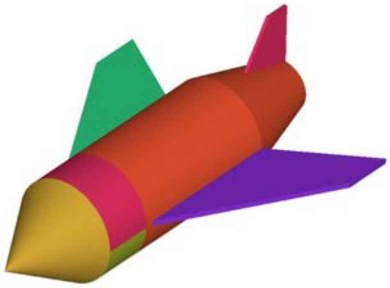


Figure 15b: Initial position with canopy closed

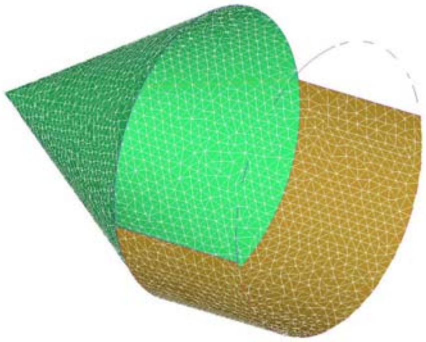


Figure 16a: Surface mesh on nose and lower canopy

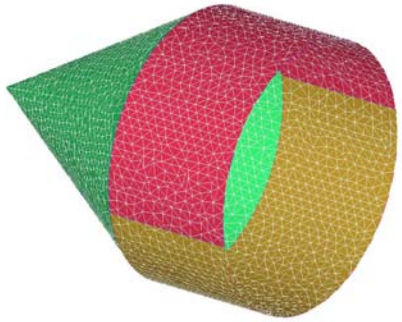


Figure 16b: Figure 16a + upper canopy

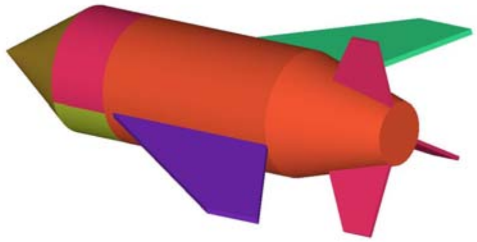


Figure 17: Basic geometry (Figure 15b) + 2 fins

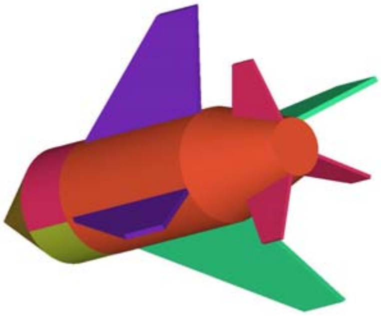


Figure 18: Geometry (Figure 17) + 2 wings

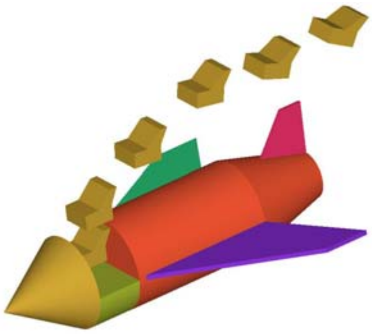


Figure 19: Basic geometry with multiple seats

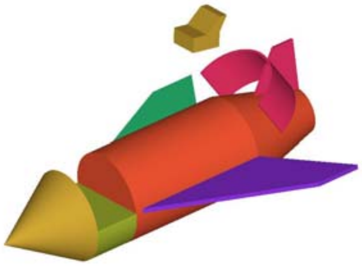


Figure 20: Basic geometry with seat and upper canopy moved

Work is in progress, which aims to extend the approach by defining a new entity type, called **Super-object.** A Super-object will consist of a series of simple objects, which are defined relative to a master object. The positioning would be carried out with the master. An example of a Super Object would be a VER with two MK-83 stores.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the Canadian Department of National Defence (DND) for the financial support.

Bibliography

[1] Aircraft/ Store Separation course, Lecture Notes, Volume 1, 1993.

[2] J.B. Carman Jr., Store Separation Testing Techniques at the Arnold Engineering Development Center, ARO, Arnold Engineering Development Center, Arnold AFB, TN, AEDC-TR-79-1, Vol. I, Aug. 1980.

[3] J. H. Fox, T. L. Donogan and R.H. Nichols, “Computed Euler Flow for a Transonic Aircraft with Store”, Journal of Aircraft, Vol. 28, pp. 389-396, 1991.

[4] A. Cenkho and M. Lutton, “A CFD Application to Store Separation-Status Report”, The Aeronautical Journal, Vol. 104, pp. 459-466, October 2000.

[5] F. Fortin, A. Benmeddour and D.J. Jones, “Application of the Canadian Code to the F/A-18C JDAM Separation”, AIAA Paper 99-0127, 37th AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV, USA, January 11-14, 1999.

[6] D.J. Jones, A. Benmeddour and F. Fortin, “CFD techniques for store carriage and release”, The Technical Cooperative Program, Key Task 2-14, Minutes of the 3rd Meeting, Ottawa, Sep. 1998.

[7] T. Amirlatifi and N. Stathopoulos, “FJ3SOLV evaluation for Store Separation Application”, Report MAU-261-220, Bombardier Aerospace, Sep. 2002.

[8] A. Jameson, W. Schmidt and E. Turkel, “Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Stepping Schemes”, AIAA 81-1259.

[9] Canadair SSM code, RAU-261-158, August 1995.

[10] A. Tahi, F. Fortin and A. Benmeddour, “Modular and Automated Unstructured Grid Generation Using ICEM CFD”, Proceedings of the 8th CASI Aerodynamics Symposium, Toronto, Ontario, Canada, April 29-May 2, 2001.

[11] E Heim, “CFD Wing/Pylon/Finned-Store Mutual Interference Wind Tunnel Experiments”, AEDC-TSR-91-p4, February 1991.