|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | | In reply refer to | | | XXXXXXXX | |
| To | | Hossny El-Sharif, Edmond Fields, Simulation, GNC, Aero | | | From | | Travis Vetter | |
| Subject | | Summary Aero Database Class need and solution. | | | Date | | August 27, 2010 | |
| Copies | |  | | | Ref | |  | |

# Summary:

Every aircraft program at NGC requires multiple complex aerodynamic buildup methods. The methods use to build up these models for each type of analysis is currently is very time consuming, prone to error, and hard to verify. A proposed solution has been in the larva stage of development between GNC, Aero, and programs since late 2008. This solution will certainly improve the simplicity of data base function calls, enable consistent data passing between GNC & Aero, improve documentation of data sources, and reduce the risk of errors in the data base buildups. Besides its near term benefits, it has the potential to dramatically streamline the data flow from Tunnel test to data base buildup, to Stability and Control analysis, to 6DoF simulation. The AeroDB class effort has broad and strong support from GNC and Aero departments, as well as IRAD,CRAD and Program efforts.

# Background:

Over the last 5 years GNC and Aero have improved the process by which aero data is passed between the groups. Aero now releases data bases as MatLab structures and a MS Word Document which describes how those tables are to be assembled into an aircraft model. However, from there, the process is cumbersome and error prone. On the GNC side, must be repeated at least twice, once for use in scripts, and once for use in simulation. These implementations, which vary greatly by Program/user must be significantly modified with each release/update of an aero database. Each time, each version must be verified against other scripts and buildups to prevent errors. Recent program experience has shown that even the most diligent of efforts can find errors well into the development process.

Starting in 2008, capitalizing on new MatLab features, and standard object oriented software practices, conceptual development, and some company investment was made to dramatically improve the Aero, Aero->GNC, and GNC efficiency of Aero model buildups. The "AeroDb Class" is a custom MatLab software class which would have common methods of storing, accessing and analyzing aerodynamic data. Continued program pull, many discussions between various groups and users, and some initial prototyping have created a feasible starting point for dedicated development.

# Solution:

A Matlab class in an object, similar to a 'Double' or 'float' which contains in it, the additional data and methods to interact with the user and other, pre-existing or future objects. So the "AeroTbl" object constrains an aerodynamic table. Including the independent data, dependent data as well as interpolation methods, derivative methods, and meta-data associated with that table. This allows the user (or another program) to efficiently access the data the same way each time, and enables the caller to know more about the data it is getting back. This object could also be read and used to generate Simulink objects which accurately recreate the data in the simulation environment. These objects have already been prototyped and work very well. They are even backwards compatible with past structure objects.

Similarly, an "AeroFcn" could be created which allows the use of generic curves to approximate data early in the development cycle to have the same calling conventions as table data. To the data base creator, this allows for the creation of reasonable placeholders early, and detailed data as it becomes available. To the user, the difference is completely transparent.

This object concept extends to a parent object, and "AeroDB" object which contains a collection of Aero Elements, (AeroTables, Aero Functions, Aero Spline, etc). This AeroDB object would include data about how those elements are assembled and calculated. Enabling a fast and generic interface to the database transparent to the general user, but highly tailorable to the designer.

This is a classic application of object oriented concepts to a long standing engineering problem. The benefits of encapsulation in this was are quickly apparent. Each user and program is calling the data the same way, while the thin nature of the solution allows complete flexibility. The simulation, analysis, and development are all acting on a common framework. Verification performed once is achieved everywhere. Updates are propagated easily. Tools and methods which use the objects are instantly applicable across all databases.

This solution is not a "do all Aero tool", nor is it intended to be, (that would be impossible!). it is only a common language for assembling data bits into data buildups. In fact, its name misleads as we are already discussing the use of this tool for propulsion data as well as other applications. But simply forming this common language provides the framework for communication, commonality, process and development to improve dramatically across these efforts.

# Next Steps

While functioning prototypes exist, rigorous development is required to make the AeroDB class functional. The class requires its core capabilities to be implemented before it can be transitioned to the program tasks. Funding and commitment is required at this time. The tasks proposed for funding are:

1) Cleanup of the prototype code (easy)

2) Development and coding of the required basic methods/calls (easy)

3) Development and coding of the derivative call methods (easy)

- At this point the class can be used efficiently -

4) Development and coding of the parent AeroDB Object (difficult)

5) Development and coding of the AeroElement -> Simulink conversion tool

6) Development and coding of the AeroDB -> Simulink Conversion tool

Given the tasks ahead, $20k is requested to achieve at least tasks 1-3. The exact efficiency of the program is highly dependent on personnel available. The best choices could dig well into tasks 4-6, however, company priorities and commitments affect level of progress.

The Aero group is involved, and has also promised some level of funding for integration, collaboration, development, and any required integration necessary to capitalize on this capability. The level of matching / details will require addition planning and detail effort.

This task was initially envisioned as its own program, however, strong synergies exist with current Common Simulation Architecture (CSAsim) efforts. As a plus-up to that task, it could be easily rolled in to the example simulation and be ready for LEMV use. While acceptance into that simulation requires additional levels of documentation and testing, those requirements are in the best practice and interest of the task. This concept is supported by all involved, and could also reduce the management/paperwork overhead associated with achieving this capability.

I can provide a brief demo of the prototype code, and any additional information on request.

Sincerely,

Travis Vetter

travis.vetter@ngc.com

C: 310-613-5911

P: 310-240-0004