Matthew Honeychuck

Aerospace Engineering

ENGR 395

Fall 2012

National Aeronautics and Space Administration (NASA)

Johnson Space Center

Houston, TX

mch5169@psu.edu

EG3: My Second Co-op Rotation at NASA's Johnson Space Center

*Note: As of this writing, I still have four weeks left in my work rotation. As such, I am still working on some of my projects.

For my first co-op tour at NASA's Johnson Space Center (JSC), I was assigned to work in the Mission Operations Directorate. However, one of the great things about NASA's co-op program is that, after the first assignment, students have the opportunity to choose where they want to work within JSC. So this fall, I decided to go work in Engineering. The Engineering Directorate is responsible for the design, development, and testing that supports human spaceflight programs at JSC, including the Space Shuttle (historically), International Space Station, and Orion/Multi Purpose Crew Vehicle programs. Within the directorate, Engineering is divided into divisions that are divided into branches. I chose to work in EG3, the Applied Aeroscience and Computational Fluid Dynamics (CFD) branch, which falls under the Aeroscience and Flight Mechanics division.

EG3 provides expertise in four main areas: Aerodynamics, Aerothermodynamics, Decelerator Systems, and Rarefied Gas Dynamics (RGD). The Aerodynamics group is responsible for determining the airflow around a spacecraft, and how the vehicle will respond to that flow. To do this, they utilize high-fidelity computer simulations, wind tunnel tests, and, ultimately, flight tests. The Aerothermodynamics group uses these same analysis tools, but they are more focused on heat transfer and thermal response of the vehicle during flight. Next, the Decelerator Systems team is responsible for the design, testing, and analysis of the parachutes that NASA uses to guide vehicles safely to the ground. Lastly, the Rarefied Gas Dynamics (RGD) group deals with low-density, non-continuum flows. Inside Earth's atmosphere, air can be considered a continuum, meaning it is a continuous mass rather than a collection of discrete particles. However, as you move farther away from Earth's surface, the air molecules become farther apart, and the continuum approximation is no longer valid. The engineers in RGD are responsible for simulating and analyzing flows in this non-continuum gas regime. During my

time in EG3, I was able to work on three projects that spanned the Aerodynamics and RGD disciplines.

My first project was to create a grid of the rotocapsule vehicle, and then use that geometry to perform CFD simulations. The rotocapsule is a concept that the Engineering Directorate has been developing for the past few years. After reentry into Earth's atmosphere, most U.S. spacecraft deploy parachutes to slow down their descent before a water landing. The rotocapsule, on the other hand, would deploy three rotor blades and use autorotation to slow the vehicle enough for a landing on solid ground. Autorotation is a state of flight in which air is drawn up through a rotor system and causes the blades to spin up (this is the concept that allows helicopters to land safely even in the event of total engine failure.) The rotocapsule would then use that stored energy to decelerate and guide itself safely to the ground.

Figure 1: Artist's rendering of the rotocapsule

As with any spacecraft, the rotocapsule team is interested in the aerodynamics of the capsule during its descent. One method for aerodynamic analysis is CFD simulations. CFD uses numerical methods to solve fluid problems and analyze interactions between the fluid and solid surfaces. There are many different flow solvers available, but NASA has developed its own software package called Overflow, which solves the Navier-Stokes equations for 3D compressible flows. Before a user can run a CFD simulation, however, he or she must first define the solid body with a grid. A grid is a collection of data points that the flow solver uses to generate a solution. The gridding process often begins with a CAD (computer-aided design) file. From there, grid generation software can be used to discretize the solid body into a surface mesh.

Before I arrived in the branch, another EG3 team member had already created grids for the body and first blade segment of the rotocapsule. However, as you can see in the Figure 1, each blade actually has two more segments that telescope out from the first. My job was to create the grids for these final two segments and then run some Overflow cases with the new geometry. To create the grids, I used another NASA-developed software package called the Chimera Grid Tools (CGT). The CGT has a graphical user interface that allows a user to create grids manually. However, one can automate the grid generation process by writing scripts in Tool Command Language (TCL). The CGT developers also created a directory called SCRIPTLIB that contains a collection of common TCL scripts used for grid generation and manipulation. Using the SCRIPTLIB functions in TCL scripts, I was able to successfully create grids for the missing segments of the blades. In the next couple of weeks, I will begin to run Overflow simulations with my grids and analyze the results.

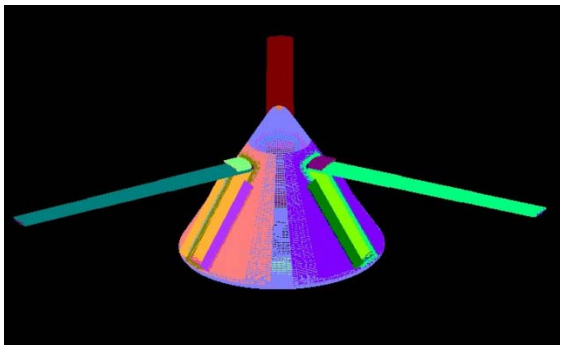

Figure 2: Completed rotocapsule grid

Whereas the rotocapsule project was completed entirely on my computer, my second project this semester was much more hands-on. My job was to outfit a remote controlled (RC) airplane with sensors connected to a microcontroller, in order to gather air data and other information about the aircraft during flight. The overall goal of the project is to develop a low cost platform for flight tests. While full scale vehicle flight testing produces the most accurate aerodynamic data, it is also extremely expensive. By using an RC plane, the EG3 can practice and refine their flight test techniques on a vehicle that is inexpensive to fly. Specifically, the branch wants to develop their ability to produce flight extracted aerodynamics. Flight extracted

aero can be explained in a basic sense by examining Newton's second law, F = ma. We already know the mass of the aircraft, m. If we can measure the accelerations, a, and any external forces acting on the plane, we can use Newton's second law to back out the rest of the aerodynamic forces. Again, this is a simplified explanation, but the basic concept is the same.

Before I got to EG3, another intern, Megan Heard, kicked off this project by ordering the aircraft, remote control system, batteries, and initial sensors that she thought she would need. She also purchased a microcontroller board called Arduino to gather the required data from each of the sensors. The Arduino is an open-source electronics platform that can receive input from several different sources and be programmed by easy-to-use software that is a slight variation of C++. In order to extract the aerodynamics, Megan determined that she needed to program the Arduino to record the data seen in column 1 of Figure 3. In column 2, you can see the sensor that the she planned to use to measure that quantity. My only change to her original plan was to use the RC signals to determine the control surface positions, instead of potentiometers.

Data Requirement	**Sensor/Measurement Method**
Position	GPS
Accelerations and rotational rates	Inertial Measurement Unit (IMU)
Altitude	Barometric Pressure Sensor
Positions of control surfaces	Read RC Signals from Radio Receiver
Angle of attack/ angle of sideslip	5-Hole Pitot Probe
Dynamic Pressure	Differential Pressure Sensor
Mach Number	Static Pressure Sensor

Figure 3: Plan for gathering the data required for flight extracted aero

Since I am pretty inexperienced with electronics, this project involved a lot of Google searching and perusing of Internet electronics blogs. Fortunately, there is a large community of Arduino users online, and many of my questions had already been asked and answered by other users. As of this writing, I am still working on configuring all the sensors and attaching the Arduino itself to the airplane. In the next few weeks, I should be able to fly the plane and demonstrate the functionality of the sensors, even if it will take some fine tuning to actually develop the flight-extracted aero.

My third project this semester involved using a software code called DAC to run simulations with visiting vehicles to the International Space Station (ISS). DAC stands for **D**irect Simulation Monte Carlo (DSMC) **A**nalysis **C**ode, and it is a program that the RGD team uses to model flows in a low-density gas environment. The process for running DAC is similar to CFD in that it all starts with a surface grid of the vehicle you want to analyze. Then the software discretizes the surrounding flow field volume into cells, the sizes of which are determined from user inputs. Next, DAC begins introducing simulated molecules into the flow field domain and tracks these particles in space and time. Collisions with the solid surface are determined directly; however, collisions between molecules are determined based on the probabilities. The software will pick two neighboring molecules in the flow field and calculate the probability that they will collide. If the collision is accepted, the software calculates the new values of momentum and energy for each particle. DAC calculates the probabilities of these collisions thousands of times for every time step in the simulation. At specified intervals, the code randomly samples the density, velocity components, and squares of the velocity components of the particles within each flow field cell to determine the flow properties. If the solution is not yet at steady-state, these properties are thrown out. If a steady state-solution has

been reached, DAC averages these properties over a specified number of time steps, and those averages represent the flow field solution.

My DAC project arose from the RGD group's desire to have a database of grids for all the visiting vehicles to ISS. One of my coworkers had already created the geometries; my job was simply to make sure they ran successfully in DAC. That way, if the team was ever asked to do some emergency analysis, they would know that they have working grids for any vehicle. For instance, early in my tour, the Japanese resupply vehicle HTV3 performed an emergency abort and fired its main engines very close to ISS. The Russian Space Agency was worried that some of the plumes from HTV's engines could have damaged the Soyuz vehicle that was docked to ISS close by. It turned out that the Soyuz was fine, but it is for situations such as this that the RGD team wanted working grids for all visiting vehicles.

In the end, I ran ten different vehicle geometries in DAC, including the ATV (European Space Agency), Progress (Russian Space Agency), Dragon (SpaceX), and Cygnus (Orbital). Some of those vehicles had both docked and undocked configurations. I ran each grid twice: once with the velocity in the x-direction, and once with velocity components equal in all

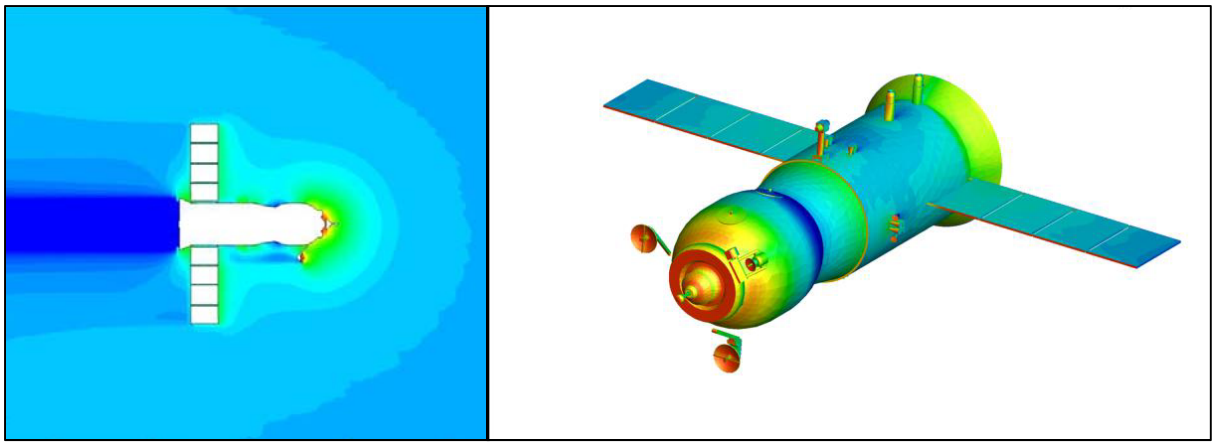

Figure 4: Visualized results after running the Progress vehicle geometry in DAC. The left picture shows the density of molecules in the flow field around the vehicle. The right picture shows the pressure on the surface of the vehicle.

directions. That way, the molecules in the flow were colliding with the vehicles in different locations and from different angles. When the simulations completed, I used DAC post processing scripts to visualize my results and make sure they made sense. For instance, Figure 4 shows sample results from a DAC run I completed with the Progress vehicle. In the left picture, I checked for things such as a well-defined wake and whether there was a higher density of particles upstream from protuberances. In the right side, I checked to make sure there were "shadows" behind protuberances and that surfaces normal to the flow direction have the higher pressures and temperatures. In the end, my testing of the grids was very successful; all of the geometries produced valid results and no errors.

As my tour comes to an end, I will leave EG3 with several skills that I will continue to use throughout my career in aerospace. First and foremost, I have gained valuable experience with the Linux operating system. I had used Linux in a limited capacity prior to this tour, but I used it on a daily basis in EG3 and as a result I am much more proficient at navigating the file system and executing shell commands. I also got a lot of programming practice, using both the TCL and Arduino languages. While experimenting with the Arduino board, I learned a lot about circuits, electronics hardware, and communication standards. Last but not least, I have increased my knowledge of aerodynamics and RGD. Though I have likely only scratched the surface of either field, becoming familiar with some of the terminology and understanding the basics behind running both types of simulations will prove useful both in the classroom back at Penn State and in future employment.

In addition to being a good technical experience, my tour in EG3 was a lot of fun! There is a camaraderie in the branch and in the division that I had not really experienced prior to working here. For instance, over the course of my tour, the division has been hosting the EG

Olympics, including such games as ping-pong, Wii boxing, and a scavenger hunt. They also organized a division picnic and there will be a Christmas party in December. I think small events like these go a long way in creating a work environment that feels more like a family. In addition to being exceptionally intelligent, everyone I have interacted with was friendly and happy to answer any questions that I had. Specifically, I want to thank my branch chief Jay Lebeau, deputy branch chief Nikki Williams, and mentors Darby Vicker and Katie Boyles for all of their help this semester. Without their guidance, I would have been lost several times throughout my tour. All in all, my experience at NASA's Johnson Space center continues to be an enjoyable one, and I look forward to coming back next summer.