To Do List and Open Issues

Javier Villarreal

1 Active Tasks

1.1 Configuration File

In the C++ version of the code, parameters for how to run the code were defined in various places. Fluid properties were in SimulationValues.txt, genetic algorithm options were defined in the main program (SOMA.cpp), and the file directories and other constant parameters were defined in Data.cpp and .hpp. Rather than having the simulation configuration spread out over many different places, a single file will be used to house all necessary parameters.

A search of the literature will find that the CFD General Notation System (CGNS) exists for that purpose. It could potentially lead to a significant improvement towards the code's ease of use in the future. For now, however, the configuration information will instead be created in a separate fortran module.

Update: The configuration file has been created in `config.f90`. It is not yet complete, but as with the data architecture, more parameters will be added as necessary for the code.

1.2 Initialization

The first major step in the actual code will now be to initialize all the variables. The steps to accomplish this are:

1. Allocate the arrays, dimensions should now be read in
2. Set all arrays equal to zero (**Q:** Does allocation accompish this?)
3. Set initial conditions by either:
    - Static initial conditions throughout the domain; or
    - Read in output from prior run.
4. Enforce boundary conditions
5. Calculate derivatives

The last two items on the list are not only for initialization, and are major components in running the code.

2 Future Improvements

2.1 Build Options

Improvement could be made to the makefile (or a different build software) for having different builds in separate directories for debugg and release versions. There are also some softwares that automatically figure out dependencies, so they don't have to be explicitly stated in the makefile.

2.2 Pre-processing/Data structure

The way the data is stored in the input text files and in variables within the code is suboptimal. Changes could be made both within pre-processing (i.e. the Matlab codes that generate those files) and how the variables are declared in Fortran.

2.3 OS Independence

During setup, the code sends a sh command to the system to create an output directory if it doesn't already exist. The consequence is that the code is thus Linux (and possible Mac) exclusive. It might be beneficial if the code could be independent of operating system, so it could also run on Windows. Perhaps there is some way to detect OS and use the `system` subroutine within Fortran accordingly, but this has not been looked into yet. Very low priority.

3 Completed Tasks

3.1 Build Options

The code is built with a basic Makefile. Compilation flags are hardcoded, so a `make clean` is needed anytime flags are changed.

3.2 Read in simulation data

The data files necessary to make the code run are different depending on whether the code is running a 2- or 3-dimensional problem. For 2D, geometries, the metadata files are:

- SimulationValues.txt (Mach, AOA, Re)
- Sizes.txt (# of domain, body, farfield, cloud, ghost, extrapolation, total nodes)

and the geometry data files are:

- x,y.txt (node coordinates)
- DX,DY.txt (DQ coefficients)
- EC.txt (extrapolation coefficients)
- Jd,Jb,Jf.txt (domain, body, farfield node indices)
- nxb,nyb.txt (body node unit normal vectors)
- nxf,nyf.txt (farfield node unit normal vectors)
- s11,s12,s21,s22.txt (Flow tangency matrices)

3.3 Create data architecture

The most important flow variables and their derivatives are stored as arrays. This is different from the way it was written in the C++ code, but it should allow for a more streamlines architecture and faster access of data. A slightly more in-depth explanation will be found in the documentation notes.

3.4 Ensure output directory exists

If the output directory does not exist before the code is run, previously the code would crash, as it would be unable to find it. The current solution is to include a line in Fortran during the initial data input to send a `mkdir` to the system to create the appropriate folder.

One constraint from this implementation is that the code runs exclusively on Linux (and possible Mac), since it's a sh command sent to the system.