CFDWARP Mini HOWTO

Create a Control File

Create a template control file:

```
WAFL101.201267435~> ./warp -w control.wrp
```

Edit the control file:

```
WAFL101.201267435~> geany control.wrp
```

The control file uses the SOAP interpreter language, for which a manual is provided. As well, the grid generation is done through the GRIDG library [1]. Learn how to write code in SOAP and how to generate a grid through the examples given in the manuals [2].

Check Grid

You can create a grid post file for GNUPLOT this way:

```
WAFL101.201267435~> ./warp -r control.wrp -opg post.01 -pt gnuplot
```

or for tecplot this way:

```
WAFL101.201267435~> ./warp -r control.wrp -opg post.01
```

Then, verify if your grid is correct by plotting the grid with GNUPLOT or TECPLOT. See GNUPLOT HOWTO [3].

Check Initial Conditions

You can create a post file including flow properties for GNUPLOT this way:

```
WAFL101.201267435~> ./warp -r control.wrp -op post.01 -pt gnuplot
```

Alternatively, a post file can be created for TECPLOT as follows

```
WAFL101.201267435~> ./warp -r control.wrp -op post.01
```

Then, verify if your initial conditions are correct by using filled contour plots in GNUPLOT or TECPLOT.

Check Boundary Conditions

For a 2D problem, you can verify if your boundary conditions are implemented correctly through the following command:

```
WAFL101.201267435~> ./warp -r control.wrp -on 20 25 40
```

This will output to the screen the node types around the point i=20, j=25 with a bandwidth of 40 nodes.

Run a Case

Once the grid, boundary conditions, and initial conditions have been verified to be valid, you can run a case as follows:

```
WAFL101.201267435~> ./warp -r control.wrp -o data.01
```

This will iterate the case with the boundary conditions, grid, and initial conditions specified in control.wrp and output the converged solution to the data file data.01. In case of a crash, lower the time step dt (or the CFL if using dual time stepping) within the Cycle() module.

Restart a Case

If you wish to restart a case that is not yet converged, this can be done with the following:

```
WAFL101.201267435~> ./warp -r control.wrp -i data.01 -o data.02
```

This tells warp to read the data file data.01 instead of applying the initial conditions and to continue the iteration process which will be saved in the data file data.02.

Post-Process the Data File

You can create a post file post.01 from a data file data.01 for GNUPLOT:

```
WAFL101.201267435~> ./warp -r control.wrp -i data.01 -op post.01 -pt
gnuplot
```

or for PARAVIEW:

```
WAFL101.201267435~> ./warp -r control.wrp -i data.01 -op post.01 -pt
vtk
```

or for TECPLOT:

```
WAFL101.201267435~> ./warp -r control.wrp -i data.01 -op post.01
```

The latter can then be read by GNUPLOT, PARAVIEW, and TECPLOT respectively.

CFDWARP has some built-in functions within the Post() module part of the control file to help with data post-processing. For instance, using the Post() module, it is possible to find the skin friction drag or the lift coefficient (such would be tedious to obtain using GNUPLOT or other post-processing software). To run post-processing commands using the Post() module, type the following at the shell prompt:

```
WAFL101.201267435~> ./warp -r control.wrp -i data.01 -opm
```

References

[1] GRIDG: a Library of Subroutines for Generating Structured Grids. https://bernardparent.ca/viewtopic.php?f=22&t=1266

[2] SOAP: An Interpreter Language for Scientific Computing. https://bernardparent.ca/viewtopic.php?f=22&t=1267

[3] GNUPLOT HOWTO. https://bernardparent.ca/viewtopic.php?f=22&t=1264