Session 7: Supersonic/Hypersonic Perfect Gas Simulations

Jeff White and Mike Park

Session Overview

- How to use FUN3D to compute supersonic and hypersonic flows
  - What are the challenges
  - List of inviscid flux types, their strengths and weaknesses
  - List of inviscid flux gradient types, their strengths and weaknesses
  - Inviscid flux types and inviscid flux gradient limiters options that work the best for supersonic and hypersonic flows
  - Required practice for running adjoint based grid adaptation for sonic boom
  - Best practices for running supersonic and hypersonic flows
  - Example of a hypersonic flow application
  - What to do when things go wrong
- We will not cover:
  - Theory/details of the inviscid flux construction
  - Theory/details of the inviscid flux gradient limiters
- What should you already know
  - Physics of supersonic and hypersonic flows

What Are the Challenges?

- The inviscid terms can be discontinuous, i.e. when there are shocks
  - Strong shocks can cause difficulties in inviscid flux schemes especially near points in the flow where the dissipation vanishes. These are called entropy problems.
  - Shocks cause discontinuities that make robust implementation of higher order schemes difficult. This is called the monotonicty problem.
- The inviscid terms can be a problem when there is strong expansion
  - Strong expansions can cause difficulties such that the local conditions approach a vacuum. This is called the positivity problem.
  - Strong expansions near the sonic point where dissipation due to the u-a eigenvalues vanishes can cause difficulties . This is called the sonic rarifaction or “expansion shock” problem.
- There are a whole host of turbulence modeling challenges that are beyond the scope of this presentation
- We will concentrate on the inviscid terms in this presentation

Inviscid Flux Types

- Inviscid flux schemes fall into several categories :
  - Contact preserving, i.e. good for viscous flows
    - Flux difference splitting scheme of *flux_construction = "roe"*
      - Non positivity near vacuum conditions
      - The sonic rarefaction problem
      - The "carbuncle" problem
      - Non preservation of the total enthalpy in shocks
      - Entropy fixes exist for some but not all of these problems
    - Flux splitting schemes such as *flux_construction = "*hllc" and "ldfss" may display some limited unphysical behavior at very strong normal shocks
  - Non-contact preserving, i.e. not usually good for viscous flows
    - Flux vector split scheme, flux_construction ="vanleer", has desirable qualities
      - Positivity near vacuum conditions
      - Preservation of the total enthalpy in shocks
  - Hybrid or "blended" schemes
    - The *flux_construction = "dldfss"* scheme is a blend of two schemes
      - The vanleer scheme at shocks via a shock detector
      - The ldfss scheme near walls via a shock and boundary layer detector

Inviscid Flux Gradient Limiter Types

- Gradient limiters are available in two types:
  - Edge based : limiting is done on an edge by edge basis, *flux_limiter = "minmod"*, *"vanleer"*, *"vanalbada"* and *"smooth"*
    - They are less dissipative and they work pretty well on hex grids but they are not as robust on mixed element or tetrahedral grids.
    - They are not "freezable" and may cause convergence to get hung up by limiter cycling. They also can not be used when using the adjoint solvers
  - Stencil based : limiting is done based on the max and min reconstructed higher order edge gradients that exist over the entire control volume "stencil", *flux_limiter = "barth"*, *"hvanleer"*, *"hvanalbada"*, *"hsmooth"* and *"venkat"*
    - They are more robust but more dissipative and work on all grid types
    - They are "freezable", i.e. they can be frozen after a suitable number of iterations which sometimes will allow the solution to converge further and they must be used when solving adjoint equations
    - Limiters with the "h" prefix include a heuristic stencil based pressure limiter to increase robustness and they also automatically activate the supersonic_floors option

Calorically Perfect Supersonic flow

- *eqn_type = "cal_perf_compress"*
- Maximum Mach number in computational domain < 3.0 such that:
  - Shocks are relatively weak
  - Expansion fans are relatively weak
- Inviscid flux options suitable for these applications:
  - When solving Euler eq. i.e. *viscous_terms = "inviscid"*
    - *flux_construction = "vanleer"*, *"ldfss"* or *"hllc"*
  - When solving Navier-Stokes eq.: *viscous_terms = "laminar"* or *"turbulent"*
    - *flux_construction = "ldfss"* or *"hllc"*
- Inviscid flux gradient limiter options most suitable for these applications:
  - For applications that do not require solving the adjoint eq's.:
    - *flux_limiter = "vanleer"*, *"vanalbada"* *"hvanleer"* or *"hvanalbada"*
  - For applications that do require solving the adjoint eq's.:
    - *flux_limiter = "hvanleer" or "hvanalbada"*

Calorically Perfect Hypersonic flow

- *eqn_type = "cal_perf_compress"*
- Maximum Mach number in computational domain 3.0 -> 10.0 such that:
  - Shocks may be strong, especially when there are normal shocks
  - Expansion fans may be strong
- Inviscid flux options suitable for these applications:
  - When solving Euler eq. i.e. *viscous_terms = "inviscid"*
    - *flux_construction = "vanleer"* or *"dldfss"*
  - When solving Navier-Stokes eq.: *viscous_terms = "laminar"* or *"turbulent"*
    - *flux_construction = "dldfss"*
- Inviscid flux gradient limiter options suitable for these applications:
  - For all applications:
    - *flux_limiter = "hvanleer"* or *"hvanalbada"*

Running the Code: Best Practices

- Applications with shocks and expansions may need to be run in 2 steps. This is sometimes true for supersonic flow and almost always true for hypersonic flow.
  - Step 1 : Run solution first order while scheduling the CFL number to evolve the solution to a quasi-steady state;
    - *first_order_iterations = xxxx*, where *xxxx* is the same as the number of iterations specified by *steps = xxxx* and
    - note that *schedule_iterations = 1 yyyy should have yyyy < xxxx*
    - *schedule_cfl = 0.1 zz.00* where *zz* is a stable CFL number that is case dependent
  - Step 2 : Restart solution higher order while scheduling the CFL number to compute the final solution;
    - Read the restart file, i.e. *restart_read = "on"*
    - first_order_iterations = *0*
    - *schedule_cfl = 0.1 hh.00* where *hh* is a stable CFL number that is case dependent and will most likely be smaller than the CFL used in Step 1.

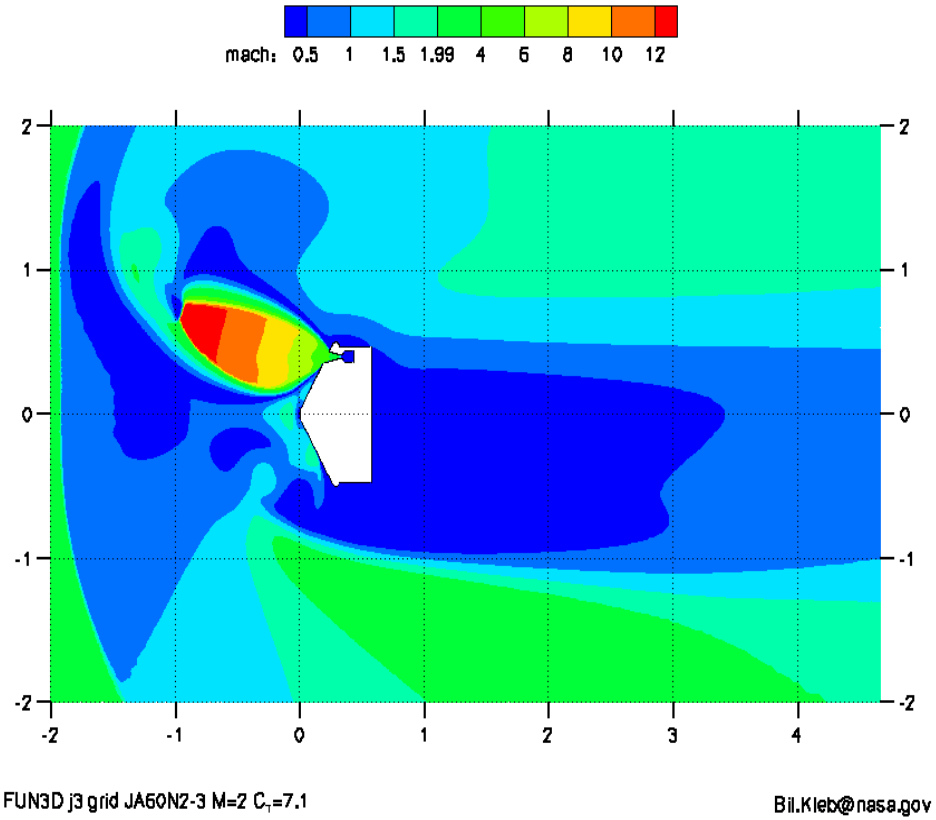
Running the Code: Sonic Boom

- Adjoint requires a frozen or differentiated limiter
- Using cut cells
  - Cut cells permit a differentiated heuristic limiter
    - *flux_limiter = "minmod"* when *–cut_cell*
  - Aft-facing steps are an issue
    - *--supersonic_floors* clips low density and pressure
    - project.cutbc *3055* allows blowing
  - *flux_construction = "vanleer"*
- Body fitted grids
  - *--freeze_limiter at_this_iteration* freezes limiter at this iteration
    - Requires a node-based "freezeable" limiter

Supersonic/Hypersonic Retro-propulsion Flow Example

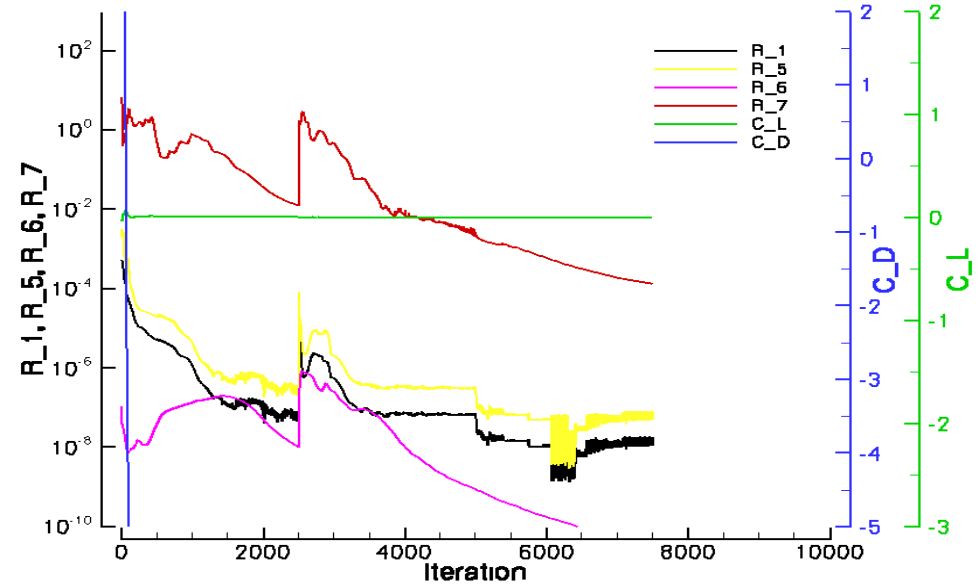
- Turbulent retro-propulsion re-entry plume flow using grid adaptation
  - Supersonic free stream (Mach = 2.0) and
  - Hypersonic plume flow (Mach = 12.0)
- Relevant namelist settings

```
&code_run_control
  steps            = 7500
  restart_read     = 'off'
/
&inviscid_flux_method
  first_order_iterations = 2500
  flux_limiter = 'hvanalbada'
  flux_construction = 'dldfss'
/
&nonlinear_solver_parameters
  schedule_iteration =   1     100
  schedule_cfl  =        0.1    10.
  schedule_cflturb   =  0.01    1.
/
```



Supersonic/Hypersonic Retro-propulsion Flow Example

- Residuals show continuity and energy eq. converged ~ 4 orders
  - Jet unsteadiness probably preventing convergence
- Lift has converged, i.e. is no longer changing
- Switch from 1st order to 2nd order scheme occurs at 2500 iterations
- The hvanalbada limiter was frozen at 5000 iterations via the command line option *--freeze_limiter 5000*



Supersonic/Hypersonic Retro-propulsion Flow Example Some Observations

- Turbulent flow has made this case easier to run because of the added dissipation caused by the eddy viscosity in the retro-propulsion jet
- If this case were laminar, it would probably be more difficult to run
  - You would need to be careful that the dldfss flux scheme does not add too much dissipation. However,
    - The careful use of feature based grid adaptation could address this
    - The proper use of output based grid adaptation would automatically address this
  - You would probably need to resort to the 2 step code running approach described earlier

What To Do When Things Go Wrong

- Try running the code 1st order before switching to 2nd order
- Try running the code 1st order longer before switching to 2nd order
- Try decreasing the CFL number
- Try decreasing the number of linear sub-iterations
- Check your grid resolution near the max. residual location
  - Under-resolved expansions can cause a lot of trouble
  - Really large grid aspect ratios near expansions can cause trouble
- Check to make sure your boundary conditions are well posed. This is especially true for internal flows

What We Learned

- A little bit about flux schemes
- A little bit about flux gradient limiters
- Which flux schemes to use for supersonic flow
- Which flux gradient limiters to use for supersonic flow
- Which flux schemes to use for hypersonic flow
- Which flux gradient limiters to use for hypersonic flow
- Some best practices
- What the convergence behavior may look like
- What to do when things go wrong