FUN3D v12.4 Training

Session 6:
Supersonic and Hypersonic Perfect Gas Simulation

Mike Park
Session Overview

- How to use FUN3D to compute **perfect gas** supersonic and hypersonic flows (eqn_type="compressible")
  - What are the challenges and strategies
  - Inviscid flux types and inviscid flux gradient limiters options that work the best for supersonic and hypersonic flows
  - Required practice for running adjoint with gradient limiters for design and grid adaptation
- Methods to initialize supersonic and hypersonic flows
- Example of a hypersonic flow application
- What to do when things go wrong
- The focus is on high-speed flows, but the strategies discussed can be used in other flow regimes
Perfect and Generic Gas Simulation

- The input parameters described in this talk are only valid for (eqn_type="compressible")
- Generic gas input parameters are different, but the philosophy is similar
- Work is underway to merge the options where possible, but consult generic gas specific documentation for details
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 monotonicity 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 rarification or “expansion shock” problem.

• Turbulence modeling challenges compound these issues but are not the focus of this talk
Inviscid Flux Types

- Inviscid flux schemes fall into several categories:
  - Contact preserving, i.e. good for viscous flows
    - Flux difference splitting scheme of \texttt{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 (Eigenvalue smoothing) exist for some but not all of these problems
    - Flux splitting schemes such as \texttt{flux\_construction = "hllc"} and \texttt{"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, \texttt{flux\_construction = "vanleer"}, has desirable qualities
      - Positivity near vacuum conditions
      - Preservation of the total enthalpy in shocks
Inviscid Flux Types

- Inviscid flux schemes fall into several categories:
  - Hybrid or “blended” schemes
    - The $\text{flux\_construction} = \text{dldfss}$ scheme is a blend of two schemes
      - The vanleer scheme at shocks via a shock detector
      - The ldffs 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 **cannot** 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.
    - They **must** be frozen when solving adjoint equations.
    - Limiters with the “h” prefix include a heuristic stencil based pressure limiter to increase robustness.
Realizability

- Nonphysical (negative density or pressure) reconstructions are set to cell averages (first order) accompanied with a “realizability” warning.
- Nonlinear density and pressure updates are floored to a ratio of freestream with the f_allow_minimum_m namelist variable.
  - The default floor may need to be lowered if the simulation requires it.
Calorically Perfect Supersonic Flow

- 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 Euler: \textit{viscous\_terms} = "inviscid"
    - \textit{flux\_construction} = "vanleer", "ldfss", "hllc" or "roe"
  - When Navier-Stokes: \textit{viscous\_terms} = "laminar" or "turbulent"
    - \textit{flux\_construction} = "ldfss", "hllc", or "roe"
- Inviscid flux gradient limiter options most suitable for these applications:
  - \textit{flux\_limiter} = "vanleer", "vanalbada", "hvanleer", or "hvanalbada"
- For applications that require solving the adjoint:
  - \textit{flux\_construction} = "vanleer" or "roe"
  - \textit{flux\_limiter} = "hvanleer" or "hvanalbada"
Calorically Perfect Hypersonic Flow

• Maximum Mach number in computational domain > 3.0 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 Euler: viscous_terms = “inviscid”
    • flux_construction = “vanleer” or “dldfss”
  • When Navier-Stokes: viscous_terms = “laminar” or “turbulent”
    • flux_construction = “dldfss”
• Inviscid flux gradient limiter options most suitable for these applications:
  • flux_limiter = “hvanleer” or “hvanalbada”
• For applications that require solving the adjoint:
  • flux_construction = “vanleer” or “roe”
  • flux_limiter = “hvanleer” or “hvanalbada”
Nonlinear Equations

- When solving nonlinear equations (e.g., Euler, Navier-Stokes), the initial guess is critical!
- Transients can be much more challenging than the steady solution
  - Solution under and over shoots can be aggravated
  - Nonphysical states may be transited
  - Boundary conditions are less robust with nearby large gradients
  - Linear system solution scheme and nonlinear defect correction solution schemes can become unstable
Strategy

• Perform the simulation in phases
  • Initialization
  • Target solution scheme
  • Optional end game that freezes limiter for better iterative convergence.
• Initialization is the primary challenge to success for high speed, internal, and propulsion flows
Initialization Strategies

• The default initialization fills the domain with freestream flow and applies strong boundary conditions
  • Creates extremely high gradients one node off the boundary
  • Sets up a unphysical expansion on backward facing surfaces
• The goal of initialization is to improve this default flow field with one that establishes the physical mechanisms of the simulations (e.g., boundary layers, shear layers, recirculation zones)
  • Moves large gradient regions away from the boundaries and into the interior of the domain
• You have the freedom to use methods that are inaccurate as long as you later restart the solution with an appropriate method for your simulation
  • Includes changing boundary conditions, freestream conditions, etc.
Initialization Strategies

• Use first_order_iterations to create a spatially first-order solution
  • This helps the nonlinear update because there are less approximations in defect correction
• Use a more dissipative flux scheme
  • Roe with excessive Eigenvalue smoothing
    • rhs_u_eigenvalue_coef, lhs_u_eigenvalue_coef, rhs_a_eigenvalue_coef, lhs_a_eigenvalue_coef
  • “vanleer” for Navier-Stokes
• Restart from a lower Mach number or angle of attack solution
• Slow down (lower CFL number or physical time step)
  • This aids the stability of the linear solve and nonlinear updates
• Combinations of these strategies
Initialization Strategies

- Explicitly initialize with the `&flow_initialization` namelist
  - Fill plenums with subsonic high density and pressure gas
  - Place a subsonic wake behind an aft facing step
  - Surround the entire vehicle with a sphere of post shock flow conditions (subsonic high density and pressure gas)
  - May reduce the execution time by allowing the use of larger CFL numbers
Solution Scheme

• See the advantages and disadvantages of the available fluxes and limiters
• Adjust (ramp) the CFL number for the best convergence rate
• Expect the solution convergence to stall due to limiter buzz
End Game

• Optionally freeze the gradient limiter to overcome limiter buzz
  • Make sure the solution is sufficiently converged
Multiple Step Approach

• Applications with shocks and expansions may need to be run in multiple 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;
    
    • Set first_order_iterations to the same as the number of iterations specified by steps
    
    • Use schedule_iteration, schedule_cfl, and schedule_cflturb to slowly increase CFL number
  
  • 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”
    
    • Set first_order_iterations = 0
    
    • The CFL ramping of schedule_iteration, schedule_cfl, and schedule_cflturb may need to be less aggressive
Supersonic/Hypersonic Retro-propulsion Flow Example

- Turbulent retro-propulsion re-entry plume flow in one run that includes the three phases
- Relevant namelist settings

```plaintext
&code_run_control
  steps = 7500
  restart_read = 'off'
/
&viscid_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.
/
```

![Mach number contour plot](image)
Supersonic/Hypersonic Retro-propulsion Flow Example

- 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 \texttt{--freeze_limiter 5000}
- Continuity and energy equation residuals converged ~ 4 orders
  - Jet unsteadiness probably preventing further convergence
- Lift has converged, i.e. is no longer changing

![Graph showing residuals and lift convergence](image-url)
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 $\partial \varphi / \partial t$ flux scheme does not add too much dissipation by refining the grid
  - You may need to resort to a multiple step running approach or explicit initialization of the flow field
Diagnosis When Things Go Wrong

- Restart the solution and visualize just before an increase in the residual
- Create movies near the largest residual location
- Check your grid resolution near the maximum residual location
  - Under-resolved expansions can cause a lot of trouble
  - Really large grid aspect ratios near expansions can cause trouble
- Try to isolate the problem to boundary condition or interior
  - Check to make sure your boundary conditions are well posed
    - This is especially true for internal flows
Diagnosis When Things Go Wrong

• Isolate the problem to linear system or nonlinear update
  • Invoke the --monitor_linear command line option
  • Set linear_projection = .true. or change the number of linear sweeps
  • Lowering CFL number can aid linear and nonlinear stability
  • Try a different initialization strategy
What We Learned

• Recommended use cases and descriptions of flux schemes
• Recommended use cases for gradient limiters and how to freeze them
• Initialization strategies
• What the convergence behavior may look like
• What to do when things go wrong