FUN3D v13.4 Training

Session 12: Feature- and Adjoint-Based Error Estimation and Mesh Adaptation

Mike Park
Learning Goals

• Background on adaptation
• Manual step-by-step output adaptation cycle
• Describe the scripts that automate this process
Available Adaptation Modes

- Split into error-estimation/metric construction and adaptive mechanics
- Output-based adaptation for capabilities with an adjoint
- Feature-based adaptation for other flow solver capabilities
- Anisotropic metric-based triangular and tetrahedral grid adaptation with a frozen mixed element boundary layer that can be subdivided
- Experimental grid adaptation for time accurate simulations
- Controlled with the &adapt_mechanics and &adapt_metric_construction namelists
- See FUN3D user manual grid adaptation overview section and complete namelist description
Output-Based Adaptation

- Mathematically rigorous approach involving the adjoint solution that reduces estimated error in an engineering output
- Uniformly reducing discretization error is not ideal from an engineering standpoint - some errors are more important to outputs

Adapted for Drag

Adapted for Shock Propagation
Local Error and Output Adaptation

Feature based
• Flow solver/physics agnostic
• Not as robust
• Requires more manual interaction

Output (adjoint) based
• Requires adjoint solution
• More robust
• Transport of errors
• Fewer user controlled parameters
Adaptation Process

**Local error based**
- Flow Solver
- Adaptation Metric
- Grid Adaptation

**Output (adjoint) based**
- Flow Solver
- Adjoint Solver
- Adaptation Metric
- Grid Adaptation

FUN3D Training Workshop
December 11-12, 2018

http://fun3d.larc.nasa.gov
Metric Adaptation Mechanics

- Parallel node insertion, node movement, element collapse, and element swap to iteratively drive mesh to satisfy an anisotropic metric $M$
Metric

• Many methods are available in literature to construct the metric
• Most commonly used methods in FUN3D are based on a reconstructed Hessian (adapt_hessian_method) of a scalar (adapt_hessian_key), i.e. Mach number
Metric Adaptation Mechanics

- Selectable with `adapt_library in &adapt_mechanics or driven` with scripts
- FUN3D is distributed with
  - `refine/one` (mature, development stopped)
  - `refine/two` (under development, 2D, mixed elements)
- FUN3D can interact with external tools
  - BAMG (Bidimensional Anisotropic Mesh Generator)
  - In-house proprietary tools
- `refine` is also available [https://github.com/NASA/refine](https://github.com/NASA/refine)
Venditti Adaptation Metric

- Output-based size specification scales the stretching and orientation of the Mach Hessian grid metric (Venditti and Darmofal).
- This error is typically evaluated on an embedded grid (with a large memory requirement) with an interpolated solution: `adapt_error_estimation='embed'`
- `adapt_error_estimation='single'` is a single grid heuristic.

\[
e_{\kappa} = \frac{|(\hat{\lambda} - \bar{\lambda}) R(\hat{u})| + |(\hat{u} - \bar{u}) R_{\lambda}(\hat{\lambda})|}{2}
\]

\[
\frac{h_{\text{request}}}{h_{\text{current}}} = \left( \frac{e_{\text{tol}}}{\sum e_{\kappa} N e_{\kappa}} \right)^{\omega}
\]
Feature based Metric

- Implemented in the Venditti framework where the nodal error estimate is replaced with a function of a solution scalar
  - adapt_feature_scalar_key
  - adapt_feature_scalar_form
- See Bibb, et al. AIAA-2006-3679 for details and Shenoy, Smith, Park AIAAJA 2014 DOI:10.2514/1.C032195 for a recent application
Cases

- Single output-based cycle performed manually on a supersonic flat plate
- Fully scripted diamond airfoil drag adaptation in supersonic flow
Supersonic Flat Plate

- Mach 2, 1,000,000 Reynolds number, Spalart-Allmaras turbulence model
Initial Flow Solution

- Follow the design directory layout convention
- Grid and `fun3d.nml` should be in a directory named `Flow`

```
$ cd Flow
$ mpirun -np 8 nodet_mpi
```
Initial Flow Solution

- Flow solver (primal) convergence history
Initial Adjoint Solution

• Adjoint function is defined in `rubber.data`
  – Only need to set the cost function, the other design inputs no used
• This is a integral of pressure along a line
  – Target off-body pressures required for sonic boom prediction

Components of func 1: boundary id (0=all)/name/value/weight/target/power
  0 boom_targ  0.000000000000000  1.0  0.00000 1.000

...
Initial Adjoint Solution

- The **boom_targ** function requires an additional namelist in *fun3d.nml*

```python
&sonic_boom
  x_lower_bound = 0.0
  x_upper_bound = 1.0
  nsignals = 1
  y_ray(1) = 0.05
  z_ray(1) = 0.1
/
```

Initial Adjoint Solution

• Initial `fun3d.nml` adjoint solver parameters

```plaintext
&code_run_control
  steps          = 200
  stopping_tolerance = 1.0e-13
  restart_read    = "off"
/
```

Typically run less adjoint iterations
Initial Adjoint Solution

- Follow the design directory layout convention
- Grid and `fun3d.nml` should be in a directory named `Flow`
- The file `rubber.data` should be in the directory above
- Adjoint solver should be run in a directory named `Adjoint`

```
$ cd Adjoint
$ mpirun -np 8 dual_mpi --outer_loop_krylov
```
Initial Adjoint Solution

- Adjoint solver (dual) convergence history
Output-Based Adaptation

• Output-based adaptation fun3d.nml parameters

```plaintext
&adapt_mechanics
  adapt_project = 'box02'  New project name
  adapt_freezebl = 0.001  Frozen boundary layer
/
```

Adapted Grid

Frozen Grid

Original Grid
Output-Based Adaptation

- Planar geometry is specified to refine/one with faux_geom
- Place in the same directory that the adaptation is executed (Adjoint)

<table>
<thead>
<tr>
<th>Number</th>
<th>Plane</th>
<th>Normal</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xplane</td>
<td>-1.00000000000000000000000</td>
<td>2.00000000000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>yplane</td>
<td>0.00000000000000000000000</td>
<td>0.10000000000000000000000</td>
</tr>
<tr>
<td>5</td>
<td>zplane</td>
<td>0.00000000000000000000000</td>
<td>0.00000000000000000000000</td>
</tr>
<tr>
<td>6</td>
<td>zplane</td>
<td>0.00000000000000000000000</td>
<td>0.8813629407814508</td>
</tr>
<tr>
<td>7</td>
<td>zplane</td>
<td>0.8813629407814508</td>
<td></td>
</tr>
</tbody>
</table>
Initial Adjoint Solution

- Follow the design directory layout convention
- Grid and `fun3d.nml` should be in a directory named `Flow`
- The file `rubber.data` should be in the directory above
- Adjoint grid adaptation should be run in a directory named `Adjoint`

```
$ cd Adjoint
$ mpirun -np 8 dual_mpi --rad --adapt

--rad = Residual Adjoint Dot-product
--adapt = Activates grid adaptation
```
Adapted Flow Solution

• **Initial fun3d.nml grid and flow conditions**

```plaintext
&project
  project_rootname = "box02"
/
&raw_grid
  grid_format = "aflr3"
  data_format = 'stream'
/
&code_run_control
  steps = 1000
  stopping_tolerance = 1.0e-13
  restart_read = "on"
/```

New project name

New grids are always AFLR3 (ugrid) stream format

The solution is interpolated
Adapted Flow Solution

- Follow the design directory layout convention
- Grid and `fun3d.nml` should be in a directory named `Flow`

```bash
$ cd Flow
$ mpirun -np 8 nodet_mpi
```
Adapted Flow Solution

- Flow solver (primal) convergence history

![Graph showing flow solver convergence history](image-url)
Adapted and Original Flat Plate Grid
F3D script

• Domain specific language written in Ruby
• Simple syntax for driving adaptation with the power of a scripting language if needed
• Input file case_specifics is scanned for updates during adaptation allowing for computational steering
• All input files are expected to be in the current directory and are also scanned for updates
  – Files are copied to Flow and Adjoint as needed
• Can generate rubber.data with $ f3d function cd
• Subcommands to start, stop, and examine adaptation in progress
• Discussed in Grid Adaptation section of the user manual
Drag-Adapted Diamond Airfoil

- Mach 2.0, inviscid flow, extremely coarse initial BAMG grid
**F3D input case_specifics example**

- Keyword value pairs to add command line options, adjust namelist settings, and specify outer adaptation cycle iterations

```plaintext
root_project 'diamond'

number_of_processors 8

adj_cl "--outer_loop_krylov"

rad_nl["adapt_complexity"] = 200*(1.5**iteration)

all_nl['data_format']="stream" if (iteration>1)

first_iteration 1
last_iteration 10
```
Namelist Setup

- **Initial** `fun3d.nml` grid and flow conditions

```plaintext
&adapt_mechanics
  adapt_library = 'refine/two'
  adapt_project = 'diamond02'
/
&adapt_metric_construction
  adapt_hessian_method = 'grad'
  adapt_hessian_average_on_bound = .true.
  adapt_twod = .true.
  adapt_statistics = 'average'
  adapt_max_anisotropy = 10.0
  adapt_complexity = 1000
  adapt_gradation = 1.5
  adapt_current_h_method = 'implied'
/
```

refine version 2 mechanics
F3D script

- Run with no subcommands for help

$ f3d

usage: f3d <command>

<table>
<thead>
<tr>
<th>&lt;command&gt;</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Start adaptation</td>
</tr>
<tr>
<td>view</td>
<td>Echo a single snapshot of stdout</td>
</tr>
<tr>
<td>watch</td>
<td>Watch the result of view</td>
</tr>
<tr>
<td>shutdown</td>
<td>Kill all running fun3d and ruby processes</td>
</tr>
<tr>
<td>clean</td>
<td>Remove output and sub directories</td>
</tr>
<tr>
<td>function [name]</td>
<td>write rubber.data with cost function [name]</td>
</tr>
</tbody>
</table>
F3D script

- To begin and watch progress

```bash
$ f3d start
$ f3d watch
```
F3D script

- Copies `fun3d.nml` into Flow directory and modifies it to set `project_rootname`, `restart_read`, and other options with the `nl_flo`, `nl_adj`, `nl_rad` hashes
- Backup copies of `fun3d.nml` are saved as `[project]_flow_fun3d.nml`, `[project]_dual_fun3d.nml`, and `[project]_rad_fun3d.nml`
- Backup copies of standard screen output are saved as `[project]_flow_out`, `[project]_dual_out`, and `[project]_rad_out`
Drag-Adapted Diamond Airfoil
Drag-Adapted Diamond Airfoil

- Mach 2.0, inviscid flow
What Can Go Wrong?

• Flow solver did not produce a project.forces file on completion
  – Indicate a setup problem (first iteration)
  – Previous grid adaptation failed (error estimation, grid mechanics)
  – Flow solver crashed or diverged
• Examine flow_out for more details

/u/mpark/fun3d/opt/bin/f3d:149:in `readlines': No such file or directory - Flow/diamond07.forces (Errno::ENOENT)
  from /u/mpark/fun3d/opt/bin/f3d:149:in `read_forces'
  from /u/mpark/fun3d/opt/bin/f3d:121:in `flo'
  from /u/mpark/fun3d/opt/bin/f3d:224:in `iteration_steps'
  from /u/mpark/fun3d/opt/bin/f3d:233:in `iterate'
  from /u/mpark/fun3d/opt/bin/f3d:310
What Can Go Wrong?

• Adjoint solver setup (particularly `rubber.data`)
Evolving Process

- Lightning talk
- Continuing development of refine grid mechanics (for CAD models)
- Implementation of error estimation techniques