FUN3D v12.7 Training

Session 15:
Adjoint-Based Design for Unsteady Flows

Eric Nielsen

Learning Goals

• The challenges of unsteady adjoint-based design
• Additional inputs for unsteady design
• Example problem: Maximize L/D for a pitching wing
• Application examples

What we will not cover
• Extensive details on setting up the most general problems
The Challenges of Unsteady Adjoint-Based Design

Sheer Expense

- The adjoint approach still provides all of the sensitivities at the same cost as analysis, and the 20x estimate still applies for the expense of an optimization
- But every simulation is now an unsteady problem
- Where the steady adjoint solver linearized about a single solution (the steady-state), the unsteady adjoint solver must essentially do this at every physical time step

Big Data

- Since the adjoint must be integrated backwards in time, this implies that we have the forward solution available at every time plane
  - Brute force it: Store the entire forward solution
  - Recompute it: Store the forward solution periodically and recompute intermediate time steps as needed
  - Approximate it: Store the forward solution periodically and interpolate intermediate time planes somehow
The Challenges of Unsteady Adjoint-Based Design

**Big Data**

*In FUN3D, we store all of the forward data to disk*

- The amount of data adds up fast – consider an example:
  - 50,000,000 grid points and 10,000 physical time steps
  - Using a 1-equation turbulence model (6 unknowns per grid point)
  - Dynamic grids (3 additional unknowns per grid point)
    → 50,000,000 x 10,000 x (6+3) x 8 bytes = 36 Terabytes
- So far, this amount of data has not been prohibitively large for our resources, but it is a lot (and we need to go bigger)
  - Will need to tackle this in the long-term
- So far, the challenge has been efficiently getting the data to/from the disk at every single time step

---

The Challenges of Unsteady Adjoint-Based Design

**Big Data**

- Conventional approaches used to write restart files are prohibitively expensive
- System should have a parallel file system
- FUN3D uses parallel, asynchronous, unformatted direct access read/writes from every rank
  - Flow solver is writing the previous time plane while the current time step is computing
  - Adjoint solver is pre-fetching earlier time planes while the current time step is computing
- This strategy performs well for the problems we have run, but is not infinitely scalable
The Challenges of Unsteady Adjoint-Based Design

Extensive Linearizations

- If dynamic grids are involved, all of the unsteady metrics and mesh motion/deformations must be differentiated at each time step
- If overset dynamic grids are involved, the relationship between the component grids must also be differentiated at each time step – both motion and interpolants
- If another disciplinary model impacts the CFD model, then that other discipline must also be differentiated, as well as the coupling procedure between the two

The Challenges of Unsteady Adjoint-Based Design

The Chaos Problem

Wish to compute sensitivities of infinite time averages for chaotic flows (DES, HRLES, LES...)

- Theory exists that states these sensitivities are well-defined and bounded

Why does conventional approach not work?

For chaotic flows:
- The finite time average approaches the infinite time average
- The sensitivity for a finite time average does not approach the sensitivity for the infinite time average

Chaotic shedding for 0012

$M_\infty=0.1 \quad Re=10,000 \quad \alpha=20^\circ$

Adjoint solution grows exponentially in reverse time
The Challenges of Unsteady Adjoint-Based Design

The Chaos Problem

- Least-Squares Shadowing (LSS) method proposed by Wang (MIT) and Blonigan (MIT; former LaRC student)
  - Key assumption is ergodicity of the simulation: long time averages are essentially independent of the initial conditions
  - Also assumes existence of a shadowing trajectory
- The LSS formulation involves a linearly-constrained least squares optimization problem which results in a set of optimality equations
- The LSS adjoint equations are a globally coupled system in space-time
- To date, work at MIT has focused on solutions of this system for academic dynamical systems containing O(1) state variables
- Close collaboration between LaRC and MIT is exploring the extension to CFD systems: enormous computational challenge for even the smallest of problems

---

The Challenges of Unsteady Adjoint-Based Design

The Chaos Problem

Shedding NACA 0012

- $M_{\infty}=0.1$  
- $Re=10,000$  
- $\alpha=20^\circ$
- 102,940 grid points

- Goal is to compute an AOA sensitivity that would allow us to maximize the time-averaged lift over final 1,000 time steps
The Challenges of Unsteady Adjoint-Based Design

The Chaos Problem

- FUN3D used to output data for use in LSS solver
  - Nonlinear residual vectors; Jacobians of residual, objective function
  - For this tiny problem, this is 1.1 TB of raw data

- Dimension of the resulting LSS matrix problem:
  102,940 grid points x 5 DOFs
  x 2,000 time planes = 1.03 billion

- Stand-alone LSS solver has been developed where decomposition is performed in time with a single time plane per core

- Global GMRES solver used with a local ILU(0) preconditioner for each time plane

Just tip of the iceberg – desired simulations are $10^6$ larger!
Desired matrix dimension = $10^9 \times 10^6 = 10^{15}$

Additional Inputs For Unsteady Design

Design Variables

- All design variables available for steady flows are also available for unsteady flows
- Design variables for a body may now also include FUN3D’s rigid motion parameters
- Also have infrastructure for other variables such as boundary condition parameters (e.g., blowing/suction rates), pilot inputs (collective, cyclics) for rotor trimming, etc
Additional Inputs For Unsteady Design

Custom Kinematics

• Design of custom kinematics: users may provide their own routine with a time-dependent $T(D)$ matrix governing an individual body's motion
  - Written in complex-variable form, FUN3D will determine its Jacobians automatically

```fortran
!-------------------------------- USER_SUPPLIED_T ---------------------------------80
! Provides route for user to supply a custom T matrix as a function of time
! and design variables. Complex-valued variables enable automated jacobian
! evaluation.
!
!=============================================================================80
subroutine user_supplied_t(ndv,current_time,dvs,t,xcg,ycg,zcg)
  use kinddefs, only : dp
  integer, intent(in) :: ndv
  complex(dp), intent(in) :: current_time
  complex(dp), intent(out) :: xcg, ycg, zcg
  complex(dp), dimension(ndv), intent(in) :: dvs
  complex(dp), dimension(4,4), intent(out) :: t
  continue
end subroutine user_supplied_t
```

Additional Inputs For Unsteady Design

Objective/Constraint Functions

• The unsteady implementation supports two forms of objective/constraint functions
• The first is based on an integral of the functional form $f$ introduced for steady flows:

$$ f_i = \sum_{n=N_i^1}^{N_i^2} f_i^n \Delta t $$

• The second form is similar, but is based on time-averaged quantities:

$$ f_i = \left[ \left( \frac{1}{(N_i^2 - N_i^1 + 1)} \sum_{n=N_i^1}^{N_i^2} C_i^n \right) - C_i^0 \right] \Delta t $$
Additional Inputs For Unsteady Design

Objective/Constraint Functions

- The sign of the cost function/constraint input toggles between the two unsteady function forms
  - Positive sign indicates form #1, negative sign indicates form #2
- In addition to the inputs required for steady simulations, the user must now also provide the time interval over which to accumulate the cost function

```
Number of composite functions for design problem statement
1

Cost function [f] or constraint [g]
1
If constraint, lower and upper bounds
0.0 0.0
Number of components for function 1
1
Physical timestep interval where function is defined
1
Composite function weight, target, and power
1.0 0.0 1.0
Components of function 1: boundary id (0=all)/name/value/weight/target/power
0 clcd 0.000000000000000 1.000 20.00000 2.000
```

Maximize Time-Averaged L/D for a Pitching Wing

- FUN3D’s design driver and the optimization packages themselves don’t distinguish between steady and unsteady CFD problems – they just see \( f \) and \( \nabla f \)
- The problem setup is very similar to steady design cases; will only highlight the differences here
Maximize Time-Averaged L/D for a Pitching Wing

**command_line.options**

```
2
2 flow
'--moving_grid'
'--timedep_adj_frozen'
2 adjoint
'--moving_grid'
'--timedep_adj_frozen'
```

- Tell the solvers that it is a moving grid case
- Also specify that we want to do a time-dependent adjoint
  - This kicks in the I/O mechanisms, among other things

---

Maximize Time-Averaged L/D for a Pitching Wing

**moving_body.input**

```
body_definitions
  n_moving_bodies = 1,
  body_name(1) = 'domain',

  parent_name(1) = '',
  n_defining_bndry(1) = -1,
  defining_bndry(1,1) = 1,
  mesh_movement(1) = 'rigid',
  x_m(1) = 0.25,
  y_m(1) = 0.0,
  z_m(1) = 0.0,
  move_m(1) = 1
```

```c
! number of bodies in motion
! name must be in quotes
! shortcut to specify all solid surfaces
! index 1: boundary number 2: body number: use any number for shortcut
! 'forced', '6dof', 'file', 'aeroelastic'
! 'rigid', 'deform'
! x-coordinate of moment center
! y-coordinate of moment center
! z-coordinate of moment center
! move mom. cntr with body/grid: 0=no, 1=yes
```

```c
! rotation type: 1=constant rate 2=sinusoidal
! reduced rotation frequency
! max rotational displacement
! x-coordinate of rotation origin
! y-coordinate of rotation origin
! z-coordinate of rotation origin
! unit vector x-component along rotation axis
! unit vector y-component along rotation axis
! unit vector z-component along rotation axis
```

- Body names must match those specified in rubber.data
Maximize Time-Averaged L/D for a Pitching Wing

rubber.data

---------------------------------------------------------------
Design Variable Information
---------------------------------------------------------------
Global design variables (Mach number / angle of attack)

<table>
<thead>
<tr>
<th>Index</th>
<th>Active</th>
<th>Value</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mach</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+01</td>
</tr>
<tr>
<td>AOA</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of bodies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Rigid motion design variables for 'domain'

<table>
<thead>
<tr>
<th>Var</th>
<th>Active</th>
<th>Value</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotRate</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>RotFreq</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>TrnVecy</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>TrnVecz</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
</tbody>
</table>

Parameterization Scheme (Massoud=1 Bandaids=2 Sculptor=4)

<table>
<thead>
<tr>
<th>Number of shape variables for 'domain'</th>
</tr>
</thead>
<tbody>
<tr>
<td>166</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Active</th>
<th>Value</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>164</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>165</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
<tr>
<td>166</td>
<td>0</td>
<td>0.000000000000000E+00</td>
<td>0.000000000000000E+00</td>
<td>0.500000000000000E+01</td>
</tr>
</tbody>
</table>

Maximize Time-Averaged L/D for a Pitching Wing

rubber.data

Number of composite functions for design problem statement

| 1 |

Number of composite functions for design problem statement

| 1 |

Cost function (1) or constraint (2)

| 1 |

If constraint, lower and upper bounds

| 0.0 0.0 |

Number of components for function

| 1 |

Physical timestep interval where function is defined

| 51 100 |

Composite function weight, target, and power

| 1.0 0.0 1.0 |

Components of function

<table>
<thead>
<tr>
<th>1: boundary id (0=all)/name/value/target/power</th>
</tr>
</thead>
<tbody>
<tr>
<td>b clcd 0.000000000000000 1.000 20.00000 2.00000</td>
</tr>
</tbody>
</table>

• Negative sign on function/constraint selection indicates time-averaging form is to be used
• Time step interval for function is also specified

\[ f = \left( \frac{1}{50} \sum_{n=51}^{100} (L/D)^n \right) - 20 \Delta t \]
Maximize Time-Averaged L/D for a Pitching Wing

- The optimization is executed just as in the steady flow case
- Here, the time-averaged value of L/D has been raised from its nominal baseline value of 0 to an optimized value of 6.8

Unsteady Design Applications

- This capability is very advanced and can require extensive problem setup for more general, complex applications
- Willing to work closely with someone interested in using it, but fire-hosing you with the intimate details at this point is probably not productive
- Instead, consider some of these prior applications to perhaps spur some ideas on future uses...
F-15 Configuration
Modify Shape to Maximize L/D Subject to Prescribed Oscillations

Flow Solution

Adjoint Solution

Active Flow Control Study

- Objective: Maximize lift using all available parameters
- Design variables include
  - External wing shape
  - Jet blowing parameters
  - Jet incidence and location
  - Relative location of slat/main/flap
- Scaling study also performed for very frequent massively parallel I/O
- Designs performed using 2,048 cores for ~5 days per run
- Mean value of lift increased by 27%
Active Flow Control Study

Flapping Wing Shape & Kinematics

http://fun3d.larc.nasa.gov
UH-60 Black Hawk
Maximize Lift Subject to Trimming Constraints

- Design variables include blade shape and collective/cyclics
- Three unsteady adjoints computed simultaneously (lift, long/lat moments)

\[
\theta = \theta_c + \theta_{c_1} \cos \psi + \theta_{c_2} \sin \psi
\]

- Adjoint shows sensitivity of objective function to local disturbances in space and time
- May also be used to perform rigorous error estimation and mesh adaptation
  - Traditional feature-based techniques do not identify such regions

http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

27
List of Key Input/Output Files

Input
• Same as for steady flows, plus
  • moving_body.input

Output
• Same as for steady flows

What We Learned

• Challenges involved with adjoint-based unsteady design
• Additional inputs required for unsteady design
• Simple design example for pitching wing
• Previous applications

Many aspects of this capability are “researchy” and applications of it would benefit from close collaboration