FUN3D v12.4 Training
Session 17:
Rotorcraft Simulations

Bob Biedron
Session Scope

• What this will cover
  – Overview of actuator-disc models for rotorcraft
  – Overview of setup for overset, articulated-blade rotorcraft simulations
    • Rigid Blades
    • Elastic Blades / Loose Coupling to Rotorcraft Comprehensive Codes
• What will not be covered
  – Rotorcraft Comprehensive Code set up and operation
  – All the many critical setup details
• What should you already know
  – Basic time-accurate and dynamic-mesh solver operation and control
  – Rudimentary rotorcraft aeromechanics (collective, cyclic…)
Introduction

• Background
  – FUN3D can model a rotor with varying levels of fidelity/complexity
    • As an actuator disk - when only the overall rotor influence is needed
    • As rotating, articulated-blade system (cyclic pitch, flap, lead-lag), with or without aeroelastic effects - if detailed airloads are needed
      – Trim and aeroelastic effects require coupling with a rotorcraft “comprehensive” code
    • As a steady-state problem for rigid, isolated, fixed-pitch blades in a rotating noninertial frame (not covered here)

• Compatibility
  – Coupled to the CAMRAD II and RCAS comprehensive codes

• Status
  – Coded for multiple rotors, but largely untested
  – Far less experience / testing with RCAS than with CAMRAD II
Time-Averaged Actuator-Disk Simulations (1/2)

- Actuator disk method utilizes momentum/energy source terms to represent the influence of the disk (pressure jump)
  - Original implementation by Dave O’Brien (GIT Ph.D. Thesis)
  - HI-ARMS implementation (SMEMRD) by Dave O’Brien ARMDEC adds trim and ability to use C81 airfoil tables (*Not covered* )
- Simplifies grid generation – disk is embedded in computational grid (note some refinement in the vicinity of actuator surface needed for accuracy)
- Any number of actuator disks can be modeled
- Different disk loading models available
  - **RotorType** = 1 actuator disk
    - **LoadType** = 1 constant (specified thrust coefficient $C_T$)
    - **LoadType** = 2 linearly increasing to blade tip (specified $C_T$)
    - **LoadType** = 3 blade element based (computed $C_T$)
  - **RotorType** = 2 actuator blades (time-accurate) *Not Functional*
Time-Averaged Actuator-Disk Simulations (2/2)

• Actuator disk implementation runs orthogonal to the standard steady-state flow solver process (compressible and incompressible)
  – Standard input grid formats for the volume grids
  – Standard solver input deck (fun3d.nml)
  – Standard output is available (project.forces, project_hist.tec, project_tec_boundary.plt)
  – Want similar solution convergence as a standard steady-state case

• Actuator disk model is activated in the command line by
  mpirun nodet_mpi --rotor
  – Rotor input deck file (rotor.input) is required in the local directory
  – rotor.input contains disk geometry and loading specifications
  – The disk geometry and loading are output in plot3d format in files source_grid_iteration#.p3d and source_data_iteration#.p3d
Incompressible Robin/Actuator Disk

Advance Ratio $= 0.051$ ($V_{\text{inf}}/V_{\text{tip}}$)
Thrust coefficient $C_T = 0.0064$
Angle of attack $= 0$ deg
Shaft angle $= 0$ deg
### rotor.input File

- **Constant/linear loading needs only a subset of the data in the file**

```plaintext
# Rotors    Uinf/Uref    Write    Soln    Force Ref    Moment Ref
           1        1.000      1500    0.001117    0.001297

--- Main Rotor -----------------------------------------------

<table>
<thead>
<tr>
<th>Rotor Type</th>
<th>Load Type</th>
<th># Radial</th>
<th># Normal</th>
<th>Tip Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1.000</td>
<td>180</td>
</tr>
</tbody>
</table>

X0_rotor    Y0_rotor    Z0_rotor    phi1    phi2    phi3
          0.696    0.0    0.322    0.00    -0.0    0.00

Utip/Uref    ThrustCoff    PowerCoff    psi0    PitchHing/R    DirRot
            19.61    0.0064    -1.00    0.0    0.0    0.0

# Blades    TipRadius    RootRadius    BladeChord    FlapHinge/R    LagHinge/R
            4    0.861    0.207    0.066    0.051    0.051

LiftSlope    alpha, L=0    cd0    cd1    cd2
            0.0    0.00    0.002    0.00    0.00

CL_max    CL_min    CD_max    CD_min    Swirl
            0.00    0.00    0.00    0.00    0.0

Theta0    ThetaTwist    Theta1s    Theta1c    Pitch-Flap
            0.0    0.00    0.0    0.0    0.00

# FlapHar    Beta0    Beta1s    Beta1c
            0    0.0    0.0    0.0

Beta2s    Beta2c    Beta3s    Beta3c
            0.0    0.0    0.0    0.0

# LagHar    Delta0    Delta1s    Delta1c
            0    0.0    0.0    0.0

Delta2s    Delta2c    Delta3s    Delta3c
            0.0    0.0    0.0    0.0
```

**Key:**
- Required for constant loading
- Required for blade element
- Not implemented (all must have a values)

- **Note Vref=Vtip is bad choice for incompressible flow - suggest using rotor induced velocity**
Incompressible Robin/Actuator Disk
Articulated-Blade Simulations

- Relies on the use of overset grids; blades may be rigid or elastic
- Elastic-blade cases (or *trimmed* rigid-blade cases) must be coupled to a rotorcraft Computational Structural Dynamics (CSD, aka comprehensive) code such as CAMRAD or RCAS
  - The CSD code provides trim solution in addition to blade deformations
  - The interface to the CSD code is through standard OVERFLOW `rotor_N.onerev.txt` and `motion.txt` type files
  - Interface codes for CAMRAD are maintained and distributed by Doug Boyd, NASA Langley (d.d.boyd@nasa.gov)
  - RCAS coupling does not require any interface codes
  - FUN3D has several postprocessing utility codes tailored to CAMRAD
- This is about as complicated as it gets with the FUN3D flow solver
  - There are *many* small details that must be done correctly; we don’t have time to cover them all here
  - Novice users of FUN3D will want to start with simpler problems
CFD/CSD – Loose (Periodic) Coupling

**Coupling Process**

1. **Rotorcraft CSD Code**
   - Iter $i = 0$: $F/M_0 = F/M_{L_0}$
   - Iter $i > 0$: $F/M_i = F/M_{L_i} + \Delta F/M_{L_i}$
   - $\Delta F/M_{L_i} = \Delta F/M_{L_{i-2}} + (F/M_{\text{CFD}_{i-1}} - F/M_{L_{i-1}})$

2. Obtain Trimmed Control Settings For Rotor F/M

3. **c/4 Motion - disp & rot**

4. **Move Grid + CFD Soln**

5. **F/M: $C_{lw}$, $C_M$, $C_{Chord}$**

6. **F/M along c/4**

7. **F/M and Trim Converged?**
   - **no**
   - **yes**

8. **Done**

---

**motion.txt and rotor_onerev.txt files common to FUN3D and OVERFLOW**

**CFD/CSD loose coupling implemented via shell script with error checking**
**dci_gen Preprocessor (1/8)**

- A rudimentary code to simplify rotorcraft setup (/utils/Rotocraft/dci_gen)
  - Uses libSUGGAR++ routines
  - Takes a single blade grid and a single fuselage / background grid (extending to far field) and assembles them into an N-bladed rotorcraft
  - Creates the SUGGAR++ XML file (`Input.xml_0`) needed by FUN3D
  - Generates, using libSUGGAR++ calls, the initial \( (t = 0) \) dci file and composite grid needed by FUN3D
  - Generates the composite-grid “mapbc” files needed by FUN3D
  - Component grids *must* be oriented as shown on following slide
    - Blade must have any “as-built” twist incorporated
    - If grids do not initially meet the orientation criteria, can use SUGGAR++ to rotate them *before* using **dci_gen**
dci_gen Preprocessor (2/8)

HART II Component Grids

Correct Axes Orientation For Fuselage Component Grid (location of origin not critical)

Correct Axes Orientation For Blade Component Grid (x-axis should correspond to blade feathering/pitch axis)
dcigen Preprocessor (3/8)

HART II Composite Grid

Composite 4-Bladed Rotorcraft (Surface) Grid
# rotor.input File

- Articulated rotors need only a subset of the data (manual defines variables)

<table>
<thead>
<tr>
<th># Rotors</th>
<th>Uinf/Uref</th>
<th>Write Soln</th>
<th>Force Ref</th>
<th>Moment Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.245</td>
<td>1500</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

*** Main Rotor ================================

<table>
<thead>
<tr>
<th>Rotor Type</th>
<th>Load Type</th>
<th># Radial</th>
<th># Normal</th>
<th>Tip Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0_rotor</td>
<td>Y0_rotor</td>
<td>Z0_rotor</td>
<td>phi1</td>
<td>phi2</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uref</th>
<th>ThrustCoff</th>
<th>PowerCoff</th>
<th>psi0</th>
<th>PitchHinge</th>
<th>DirRot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.0064</td>
<td>-1.00</td>
<td>0.0</td>
<td>0.0466</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Blades</th>
<th>TipRadius</th>
<th>RootRadius</th>
<th>BladeChord</th>
<th>FlapHinge</th>
<th>LagHinge</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>26.8330</td>
<td>2.6666</td>
<td>1.741</td>
<td>0.0466</td>
<td>0.0466</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LiftSlope</th>
<th>alpha, L=0</th>
<th>cd0</th>
<th>cd1</th>
<th>cd2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.28</td>
<td>0.00</td>
<td>0.002</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CL_max</th>
<th>CL_min</th>
<th>CD_max</th>
<th>CD_min</th>
<th>Swirl</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.50</td>
<td>-1.50</td>
<td>1.50</td>
<td>-1.50</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theta0</th>
<th>ThetaTwist</th>
<th>Thetals</th>
<th>Thetalc</th>
<th>Pitch-Flap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.00</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># FlapHar</th>
<th>Beta0</th>
<th>Betals</th>
<th>Betalc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beta2s</th>
<th>Beta2c</th>
<th>Beta3s</th>
<th>Beta3c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># LagHar</th>
<th>Delta0</th>
<th>Delta1s</th>
<th>Delta1c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Delta2s</th>
<th>Delta2c</th>
<th>Delta3s</th>
<th>Delta3c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Key:**
- Required for rigid and elastic
- Required for untrimmed rigid
- Unused (must have a value)
Nondimensional Input (1/2)

- Typically define the flow reference state for rotors based on the tip speed; thus in `rotor.input`, set \( \frac{U_{\text{tip}}}{U_{\text{ref}}} = 1.0 \) (data line 4)
- This way, \( \frac{U_{\text{inf}}}{U_{\text{ref}}} \) (data line 1) is equivalent to \( \frac{U_{\text{inf}}}{U_{\text{tip}}} \), which is the Advance Ratio, and is usually specified or easily obtained
- Since the reference state corresponds to the tip, the `mach_number` in the `fun3d.nml` file should be the tip Mach number, and the `reynolds_number` should be the tip Reynolds number
- Nondimensional rotation rate: not input directly, but it is output to the screen; you might want to explicitly calculate it up front as a later check:

\[
\Omega^* = \frac{U_{\text{tip}}^*}{R^*} \quad (\text{rad/s}, \quad R^* \text{ the rotor radius})
\]

and recall
\[
\Omega = \frac{\Omega^* (L_{\text{ref}}^*/L_{\text{ref}})}{a_{\text{ref}}^*} \quad (\text{compressible})
\]
so with \( a_{\text{ref}}^* = \frac{U_{\text{ref}}^*}{M_{\text{ref}}} \) and taking \( L_{\text{ref}}^* = R^* \)

\[
\Omega = M_{\text{ref}} \left( \frac{U_{\text{tip}}^*}{U_{\text{ref}}^*} \right) / R \quad (\text{compressible})
\]

\[
\Omega = \frac{U_{\text{tip}}^*}{U_{\text{ref}}^*} / R \quad (\text{incompressible})
\]
Nondimensional Input (2/2)

• Nondimensional time step:
  
  time for one rev: \( T^* = \frac{2\pi}{\Omega^*} = \frac{2\pi R^*}{U_{tip}^*} \) (s)

  and recall \( t = t^* a_{ref}^* \left( \frac{L_{ref}^*}{L_{ref}^*} \right) \) (compressible)

  so with \( L_{ref}^* = R^* \) we have

  \( T = a_{ref}^* \left( \frac{R}{R^*} \right) \frac{2\pi R^*}{U_{tip}^*} = \frac{2\pi R}{(M_{ref} U_{tip}^* / U_{ref}^*)} \) (nondim time / rev)

  For N steps per rotor revolution:

  \( \Delta t = \frac{2\pi R}{(NM_{ref} U_{tip}^* / U_{ref}^*)} \) (compressible)

  \( \Delta t = \frac{2\pi R}{(NU_{tip}^* / U_{ref}^*)} \) (incompressible)

• Note: the azimuthal change per time step is output to the screen in the Rotor info section. Make sure this is consistent, to a high degree of precision (say at least 4 digits), with your choice of N steps per rev – you want the blade to end up very close to 360 deg. after multiple revs!

• Formulas above are general, but recall we usually have ref = tip, at least for compressible flow
CAMRAD Considerations

• User must set up basic CAMRAD II scripts; the `RUN_LOOSE COUPLING` script provided with FUN3D requires 3 distinct, but related CAMRAD scripts
  – `basename_ref.scr`
    • Used to generate the reference motion data used by CAMRAD
    • Set this file to use rigid blades; zero collective/cyclic; no trim
  – `basename_0.scr`
    • Used for coupling/trim cycle “0”
    • Set up for elastic blades with trim; use CAMRAD aerodynamics exclusively (no delta airloads input); simplest aero model will suffice
  – `basename_n.scr`
    • Used for all subsequent coupling/trim cycles
    • Set up for elastic blades with trim; use same simple CAMRAD aerodynamics but now with delta airloads input
Untrimmed Rigid-Blade Simulations

• Overview of the basic steps

1. Prepare rotor blade and fuselage grids, with proper axis orientation
2. Set up the `rotor.input` file based on desired flight conditions
3. Run the `dci_gen` utility to create a composite mesh and initial dci data
4. Set up `fun3d.nml` and `moving_body.input` files
5. Optionally set up the `&slice_data` namelist in the `fun3d.nml` file
6. Run the solver with the following command line options (in addition to any other appropriate ones, like `--temporal_err_control`)
   ```
   --moving_grid --overset --overset_rotor --dci_on_the_fly
   --dci_period 360 --reuse_existing_dci
   ```
   If optional step 5 is used, add the following (N as desired, typically 1)
   ```
   --slice_freq N --output_comprehensive_loads
   ```
7. Number of time steps required is case dependent – usually at least 3 revs
Trimmed, Elastic-Blade Simulations

• Overview of the basic steps; steps 1-4 are the same as for the untrimmed rigid-blade case; use of CAMRAD is assumed

5. Set up the &slice_data namelist; not optional

6. Set up the 3 CAMRAD run-script templates

7. Set up the RUN_LOOSE COUPLING run script (a c-shell script geared to PBS environments); user-set data is near the top – sections 1 and 2

8. Set up the fun3d.nml_initial and fun3d.nml_restart files used by the run script; typically set the time steps in the initial file to cover 2 revs, and 2/N_{blade} revs in restart version

9. If using the run script make sure all items it needs are in place; script checks for missing items, but it gets old having to keep restarting because you forgot something!

10. Number of coupling cycles required for trim can vary, but 8-10 is typical for low-moderate thrust levels; high thrust cases near thrust boundary may require 10-15; user judges acceptable convergence
RUN_LOOSE_COUPLING Directory Tree

Run Directory
Script executed here
FUN3D runs here

CAMRAD

Reference

Trim_0

Trim_1

Principal solver files archived here at end of each trim cycle

Script creates all subdirectories

CAMRAD and interface codes run here and output stored here for each trim cycle

FUN3D Training Workshop
March 24-25, 2014
Postprocessing

Sample Plots Possible Via process_rotor_airloads.f90 Output
The End