FUN3D v14.0 Training
Linearized Frequency Domain

Kevin Jacobson

Learning Goals

What we will cover:

- Static aeroelastic analysis with Stabilized Finite Element (SFE) solver in FUN3D
- Linearized Frequency Domain (LFD) analysis with SFE

What we will not cover:

- Using Generalized Aerodynamics Forces from LFD for aeroelastic prediction

What you should already be familiar with:

- Aeroelastic material covered in the 13.4 training (December 2018 Workshop) and the updates in the 14.0 training on aeroelasticity
- Details of steady SFE analysis from the 14.0 training on SFE

Recap from SFE training

- Much of the infrastructure for the finite-volume solver also drives SFE
  - Mesh partitioning
  - Mesh motion including the internal modal aeroelastic solver
  - Solution sampling
- `sfe.cfg` is an additional namelist-like file for SFE-specific inputs
  - Smoother inputs add local dissipation for cases with local supersonic flow
  - Linear solver settings
  - Indices start at 0

Static Aeroelastic Analysis with SFE (1/2)

The general steps of static aeroelastic analysis and SFE steady analysis apply:

- Certain **`fun3d.nml`** parameters can be overridden by equivalent inputs in the **`sfe.cfg`** file
  - This lets SFE run in steady mode while using the unsteady modal solver

- As covered in the SFE training, FUN3D will use the SFE solution to compute and report loads based on both FV and FE integration of the forces
  - In version 14.0, the loads applied in the internal modal solver will use the FV force integration during static aeroelastic coupling

Static Aeroelastic Analysis with SFE (2/2)

There are two ways to compute static aeroelastic solutions with SFE:

1. Time-step coupling:
   - Couple each steady SFE solver iteration to the modal solver which is running with a large time step and critical damping
   - Similar to the way static aeroelastic solutions are computed with the FV solver
2. Full-solver coupling:
   - Fully converge the steady SFE solver with fixed modal displacements, compute new modal displacements based on the modal forces, and repeat
   - Looser coupling but the SFE nonlinear controller does not have to chase a moving target

SFE Static Aeroelastic – Time-Step Coupling

1. Set up the fun3d.nml and moving_body.input as if you were doing a static aeroelastic simulation with the finite-volume solver:
   - **`time_accuracy = “2ndorderOPT”`**, with a large nondimensional time step
   - modal properties in **`&aeroelastic_modal_data`** with **`damp = 0.999`**
   - **`restart_read = “on_nohistorykept”`**
2. Set the **`flow_solver = “sfe”`** in the fun3d.nml
3. Start from the sfe.cfg file used for the steady analysis. Add the following entry:

```
time_accuracy = 0 ! Steady analysis
```

4. Use the command line argument: **`--aeroelastic_internal`**

Langley Research Center

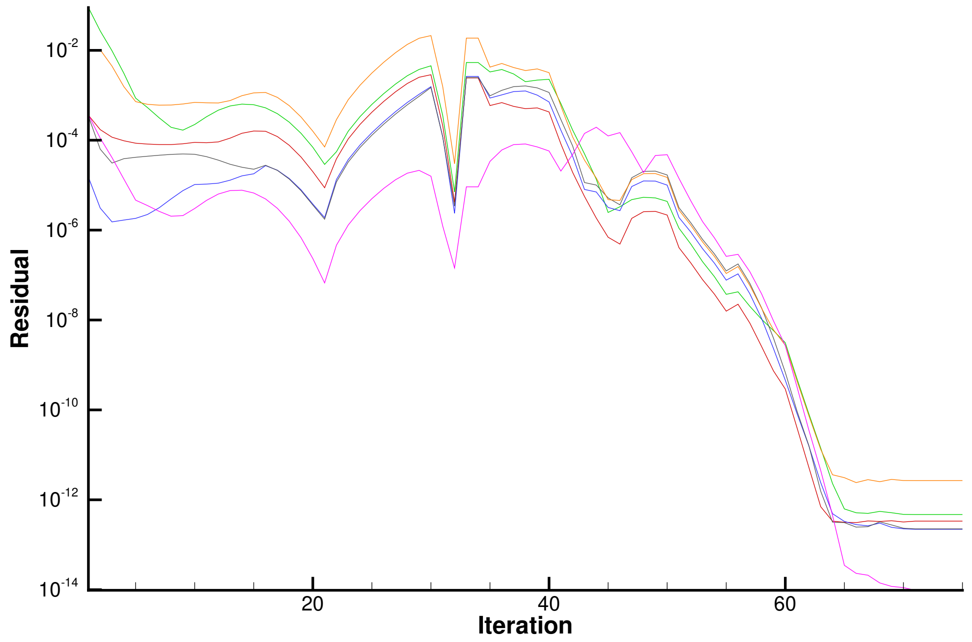
Tutorial Case: BSCW

- The Benchmark Supercritical Wing (BSCW) is one of the Aeroelastic Prediction Workshop (AePW) cases.
  - The tutorial cases here will step through Case 2 of the 2nd AePW: flutter prediction at Mach=0.74, AoA=0.0
    - Steady SFE analysis -> static aeroelastic SFE analysis -> LFD
    - Plunge and linearized pitch structural degrees of freedom
- The steady analysis is repeated from the SFE training:
  - fun3d.nml

```
 &governing_equations
     prandtlnumber_molecular = 0.755
     flow_solver = "sfe"
  /
```

  - sfe.cfg

```
    smoothing = .true.
    number_of_smoothers = 2
    smoother_type(1) = ramped
```



Tutorial Case: Static Aeroelastic BSCW – Time-Step Coupling (1/4)

fun3d.nml:

```
&global
   moving_grid = .true.
/
&governing_equations
   prandtlnumber_molecular = 0.755
   flow_solver = "sfe"
/
&nonlinear_solver_parameters
   time_accuracy = "2ndorderOPT"
   time_step_nondim =     100.0000000000000
   subiterations =                1
/
&code_run_control
   steps = 200
   stopping_tolerance =   1.000000000000000E-035
   restart_read = "on_nohistorykept"
/
```

Tutorial Case: Static Aeroelastic BSCW – Time-Step Coupling (2/4)

moving_body.input:

```
&body_definitions
  n_moving_bodies = 1
  body_name(1) = 'wing'
  n_defining_bndry(1) =  1
  defining_bndry(1,1) = 1
  motion_driver(1) = 'aeroelastic'
  mesh_movement(1) = 'deform'
/

&aeroelastic_modal_data
  nmode(1)   = 2
  uinf    = 4508.4
  grefl   = 1.0
  qinf    = 1.1722
  freq(1,1) = 20.92
  freq(2,1) = 32.67
  gmass(1,1)   = 1.0
  gmass(2,1)   = 1.0
  damp(1,1)   = 0.99999
  damp(2,1)   = 0.99999

  use_modal_deform = .true.
  modal_ref_amp(1,1) = 0.005
  modal_ref_amp(2,1) = 0.005
/
```

Tutorial Case: Static Aeroelastic BSCW – Time-Step Coupling (3/3)

sfe.cfg:

```
time_accuracy = 0
smoothing = .true.
cfl_init = 1e5
```

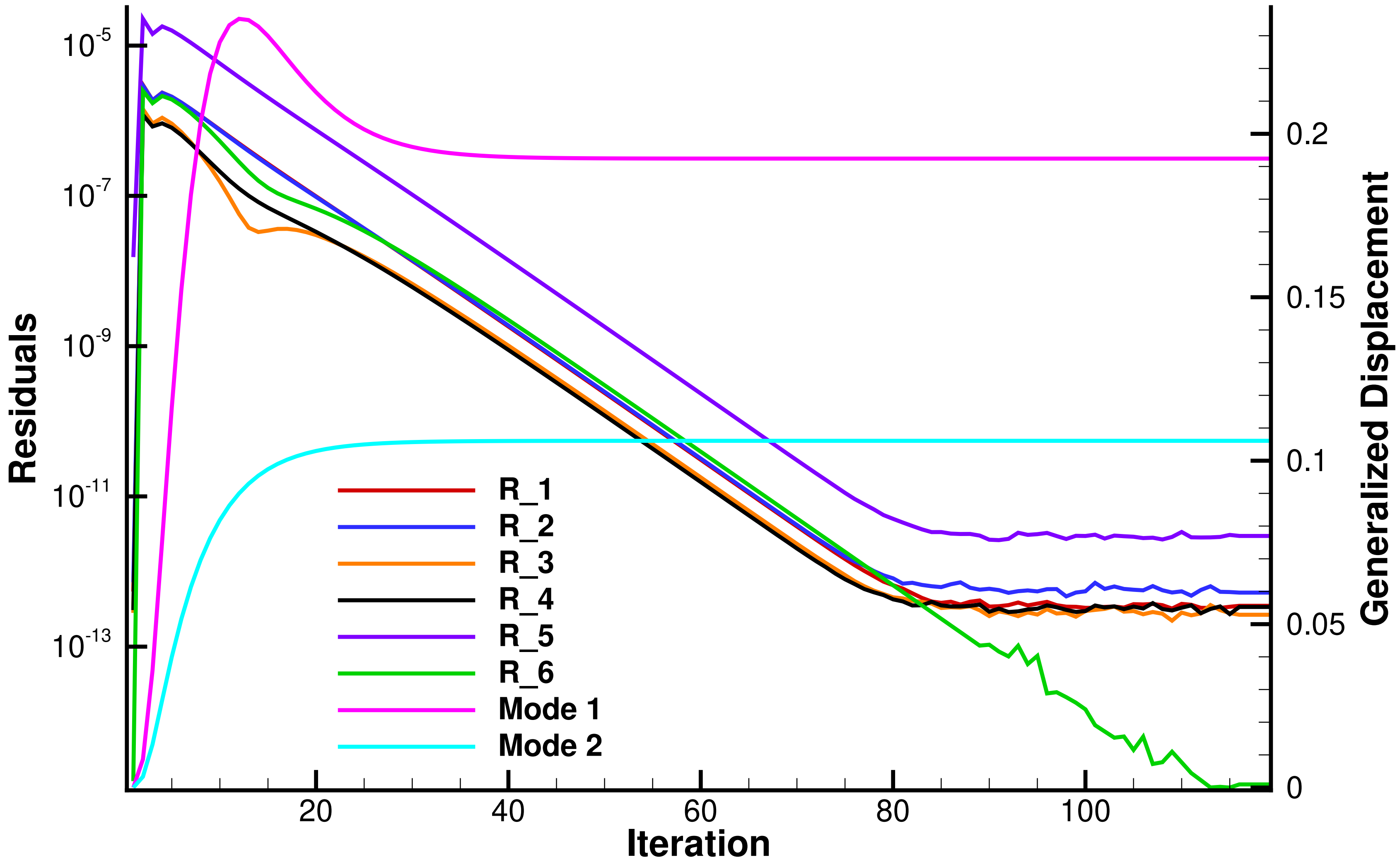
Additional input files:

- bscw_coarse_mixed_nc.flow from the steady analysis
- bscw_coarse_mixed_nc_body1_mode*.dat

Command

- `mpirun nodet_mpi --gamma 1.136 --aeroelastic_internal`

Tutorial Case: Static Aeroelastic BSCW – Time-Step Coupling (4/4)



SFE Static Aeroelastic – Full-Solver Coupling

- Full-solver coupling inputs are the same as the time step coupling version except using **`gdisp0`** and **`moddfl=-1`** to set the modal displacements and hold them fixed during the simulation
- Write a driver script that does the following in a loop:

1. Run FUN3D with **`moddfl=-1`**
2. Compute new modal displacements from the modal forces in the aehist files and the modal properties, gdisp0 = modal_force / modal_stiffness
   - You can add underrelaxation here for stability
3. Write new modal displacements to the **`gdisp0`** entry of moving_body.input

- Example Python script provided in tutorial

Tutorial Case: Static BSCW – Full-Solver Coupling (1/2)

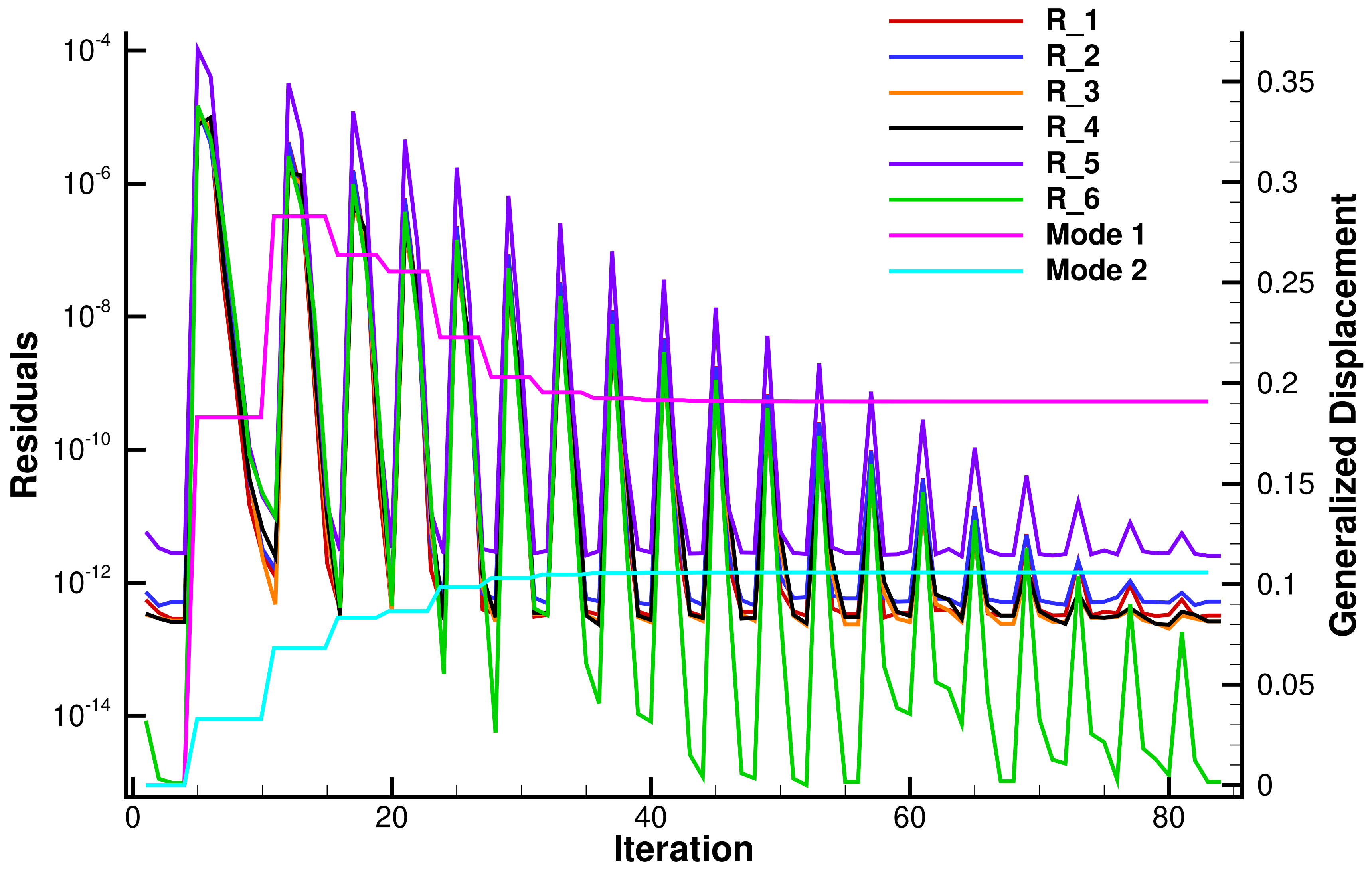
moving_body.input:

```
&body_definitions
    n_moving_bodies = 1
    body_name(1) = 'bscw_coarse_mixed_nc'
    n_defining_bndry(1) = 1
    defining_bndry(1,1) = 1
    motion_driver(1) = 'aeroelastic'
    mesh_movement(1) = 'deform'
/

&aeroelastic_modal_data
    nmode(1) = 2
    uinf = 4508.4
    grefl = 1.0
    qinf = 1.1722
    freq(1:2,1) = 20.92, 32.67
    gmass(1:2,1) = 1.0, 1.0
    moddfl(1:2,1) = -1, -1
    gdisp0(1:2,1) = 0, 0
    use_modal_deform = .true.
    modal_ref_amp(1:2,1) = 0.005, 0.005
/
```

Langley Research Center

Tutorial Case: Static BSCW – Full-Solver Coupling (2/2)



LFD Background

The LFD method is the exact linearization of the flow residual about a steady flow field

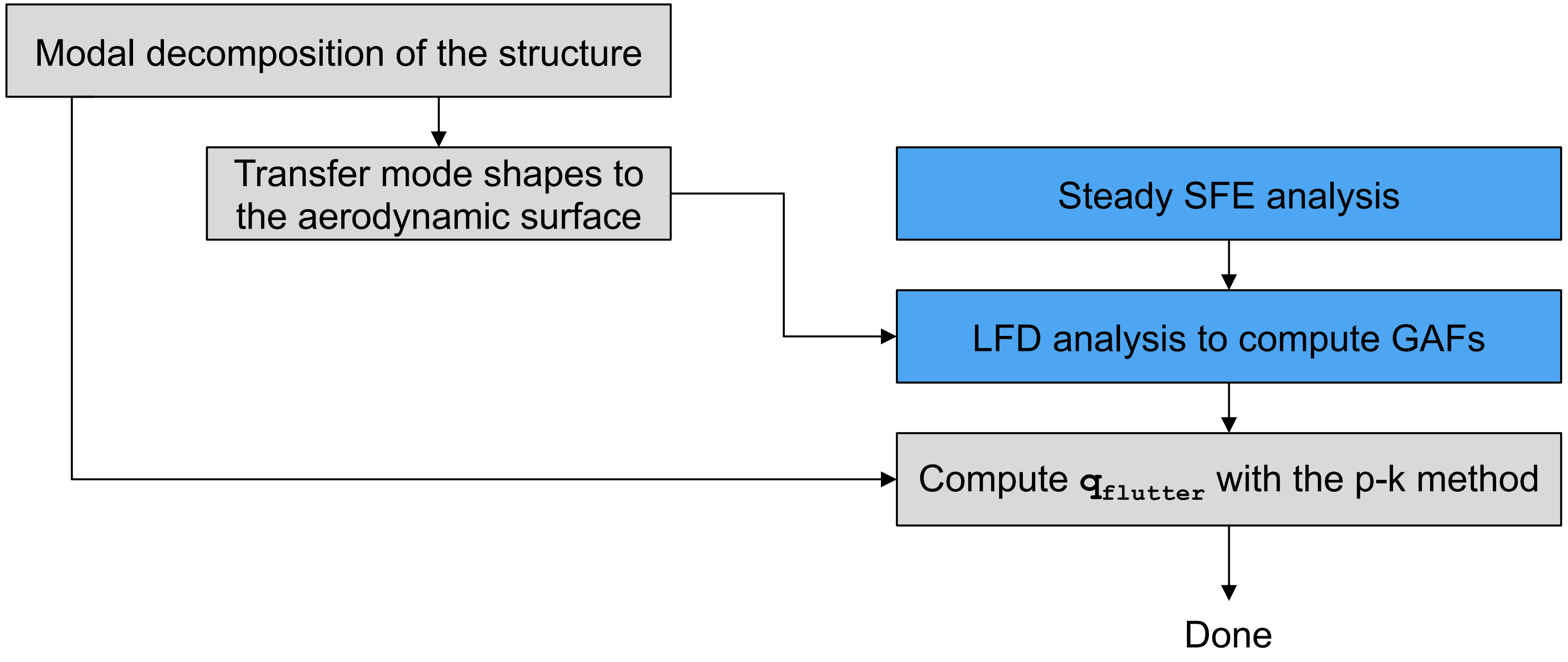
- The linearized flow response is computed due to harmonic input of the mesh motion at a prescribed frequency

$$\left(i\omega \mathbf{M}\big|_{\mathbf{x}_{G0},\mathbf{q}_0} + \frac{\partial \mathbf{R}}{\partial \mathbf{q}}\bigg|_{\mathbf{x}_{G0},\mathbf{q}_0} \right) \hat{\mathbf{q}}_j = - \frac{\partial \mathbf{R}}{\partial \mathbf{x}_G}\bigg|_{\mathbf{x}_{G0},\mathbf{q}_0} \hat{\mathbf{x}}_{G,j} - i\omega \frac{\partial \mathbf{R}}{\partial \dot{\mathbf{x}}_G}\bigg|_{\mathbf{x}_{G0},\mathbf{q}_0} \hat{\mathbf{x}}_{G,j}$$

- Solve this complex-valued linear problem for each structural mode and a range of frequencies
- Generalized Aerodynamic Forces (GAFs) are computed from the linearized mesh motion and flow response
- GAFs can model aerodynamics in a subsequent aeroelastic analysis like a p-k flutter solver

LFD Flutter Analysis Process (1/4)

Linearized about the rigid (jig) shape:

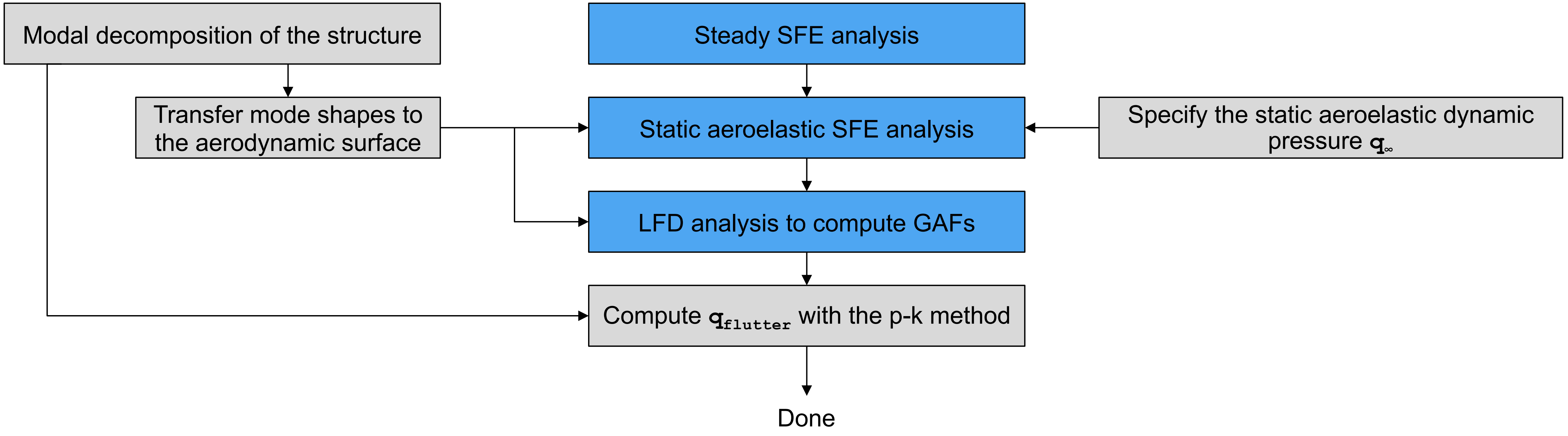

Modal decomposition of the structure
Transfer mode shapes to the aerodynamic surface
Steady SFE analysis
LFD analysis to compute GAFs
Compute $q_{flutter}$ with the p-k method
Done

🟦 FUN3D Analysis

Langley Research Center

LFD Flutter Analysis Process (2/4)

Standard process with static aeroelastic analysis:

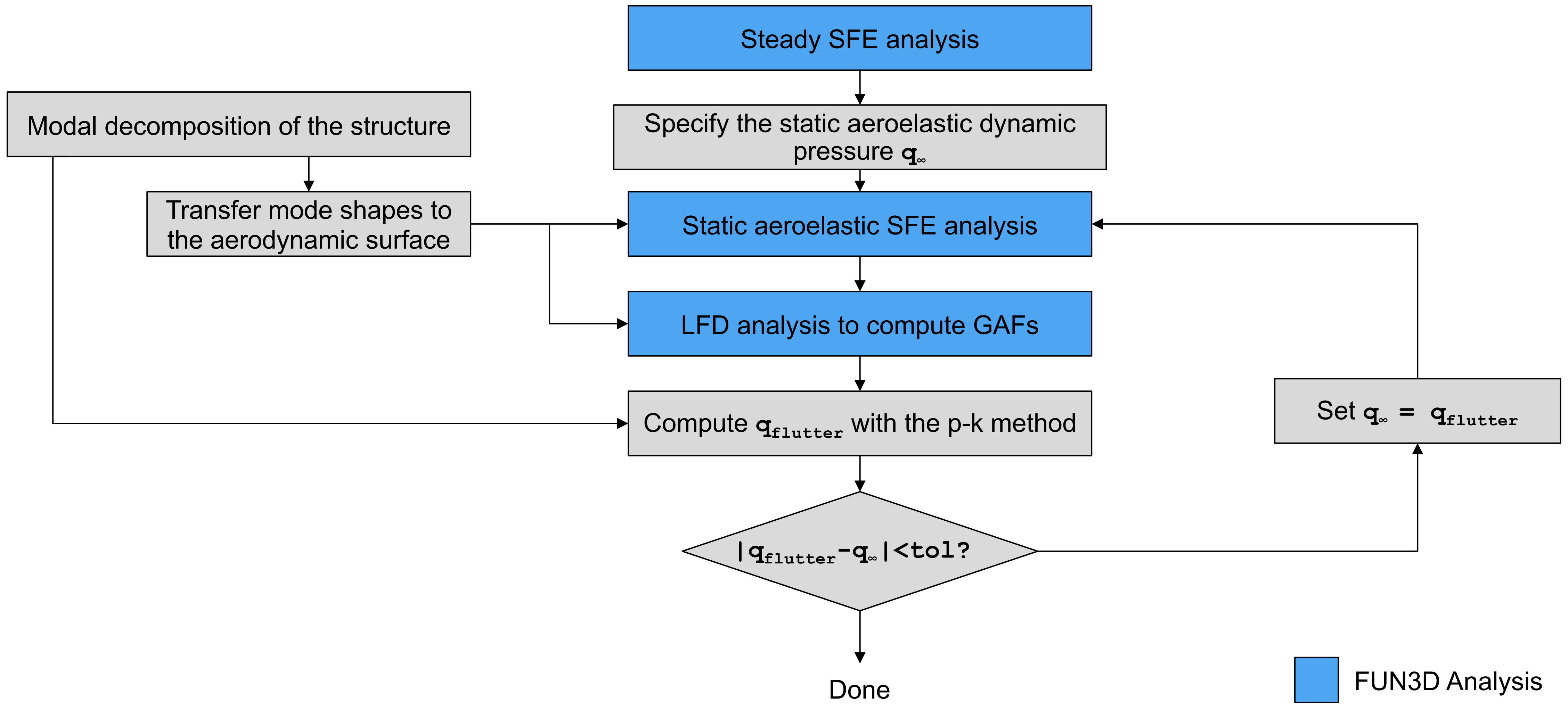

Modal decomposition of the structure
Transfer mode shapes to the aerodynamic surface
Steady SFE analysis
Static aeroelastic SFE analysis
Specify the static aeroelastic dynamic pressure q_∞
LFD analysis to compute GAFs
Compute $q_{flutter}$ with the p-k method
Done

🟦 FUN3D Analysis

Langley Research Center

LFD Flutter Analysis Process (3/4)

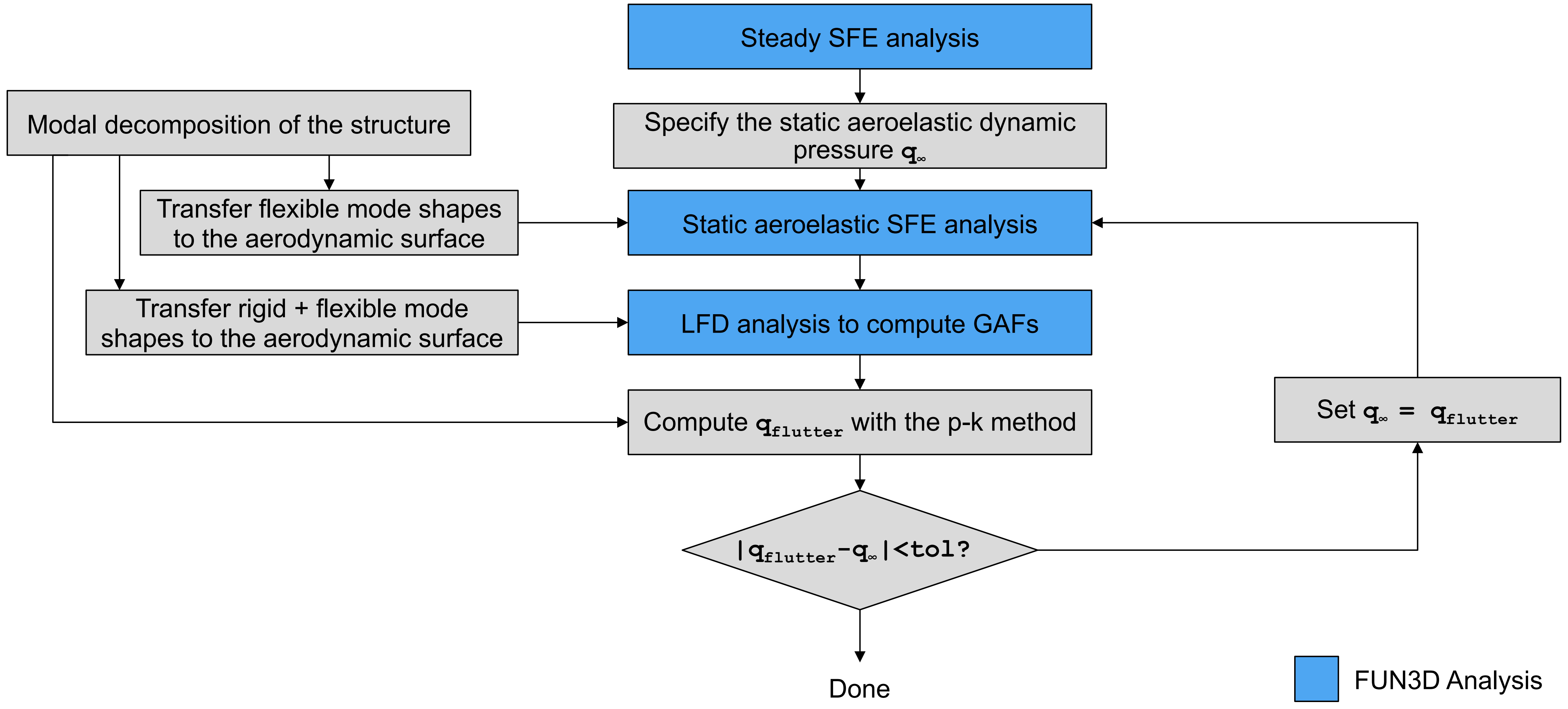
For highly nonlinear conditions, GAFs may be a strong function of q_∞:



Steady SFE analysis
Modal decomposition of the structure
Specify the static aeroelastic dynamic pressure q_∞
Transfer mode shapes to the aerodynamic surface
Static aeroelastic SFE analysis
LFD analysis to compute GAFs
Compute $q_{flutter}$ with the p-k method
Set $q_\infty = q_{flutter}$
$|q_{flutter} - q_\infty| < tol?$
Done
FUN3D Analysis

LFD Flutter Analysis Process (4/4)

The mode shapes at the LFD stage do not need to match the static aeroelastic modes:


Steady SFE analysis
Specify the static aeroelastic dynamic pressure q_∞
Modal decomposition of the structure
Transfer flexible mode shapes to the aerodynamic surface
Static aeroelastic SFE analysis
Transfer rigid + flexible mode shapes to the aerodynamic surface
LFD analysis to compute GAFs
Compute $q_{flutter}$ with the p-k method
Set $q_\infty = q_{flutter}$
$|q_{flutter} - q_\infty| < tol?$
Done
FUN3D Analysis

Complex mode FUN3D

LFD analysis requires a complex FUN3D executable. You must configure FUN3D to make the executable:

1. `../configure --enable-full-precision --enable-complex {other options}`
2. `make –j complex`
3. `make install`

- This will create and install both **`nodet_mpi`** and **`complex_nodet_mpi`**

Things to consider with the complex executable:

- Run the steady and static analyses with the real-valued executable because it will be faster. The complex executable can read real-valued restart files (*.flow) for LFD
- GFortran requires you to explicitly specify complex numbers in the namelist files. If you use a real-valued namelist, it will return an error:
  - `Probable incomplete read of namelist: &reference_physical_properties`
  - Need to change all float entries from `mach_number = 0.7` to `mach_number = (0.7, 0.0)`
  - Intel's ifort compiler does not require this conversion. It can directly read the floats.

LFD Inputs – moving_body.input (1/2)

Start from the static aeroelastic fun3d.nml and moving_body.input files

`&aeroelastic_modal_data`

- Must use **`use_modal_deform = .true.`** for LFD
    - The modal mesh deformation is how FUN3D computes the mesh perturbations for the RHS of the LFD linear problem
- **`lfd_write_mode_files = .true.`** – whether to write LFD mode shapes to files for reuse or recompute at each frequency. Writing modes generates many files to store the volume mode shapes but saves duplicate calls to the mesh deformation
- **`lfd_use_existing_mode_files = .false.`** – whether to read existing LFD mode shapes to files. The previous run must have used the same number of MPI ranks.

LFD Inputs – moving_body.input (2/2)

`&aeroelastic_modal_data`

- **`lfd_nfreq = 0`** – The number of frequencies at which GAFs will be computed
  - Typically, 10-15 frequencies based on reduced frequency, e.g., k = [0.0, 0.01, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0]
- **`lfd_freq(1) = 0.0`** – Frequencies at which to compute GAFs
  - Frequencies need to be in same units as structural frequencies, **`freq`** [typically rad/s],
  - To convert from reduced frequency: $\omega = k*U_\infty/b$
- **`lfd_nmodes_perturbed = -1`** – The number of modes to perturb for the LFD RHS.
  - Default of -1 will perturb each mode sequentially.
- **`lfd_modes_perturbed(1) = 1`** – array of mode numbers to perturb for LFD RHS. Do not need to specify if **`lfd_nmodes_perturbed = -1`**

LFD Inputs – fun3d.nml

In the fun3d.nml:

```
&global
   moving_grid = .true.
/
&governing_equations
   flow_solver = "sfe"
/
&nonlinear_solver_parameters
  time_accuracy = "lfd"
/
```

- For **&code_run_control**,
  - set **steps =** (number of perturbed modes) x (number of LFD frequencies) specified in the moving_body.input. Each "step" is an LFD linear system for a structural mode being perturbed at a frequency
  - **restart_read = "on_nohistorykept"**

LFD Inputs – sfe.cfg

- Parameters that affect the flow equations like **`smoothing`** or **`weak_bc`** need to be the same as the static aeroelastic simulation
  - If you used a ramped smoother to aid convergence, it should be turned off
- LFD linear problems can be difficult to converge
  - Typically need to increase the total number of linear solver search directions, **`max_matvecs = 1500`**
- On LFD problems where the linear solver stalls, add q-ordering with low prune width to stabilize the linear solver
  - **`q_ordering = 1`**
  - **`prune_width = 8.0`**
- **`level_of_fill`** can also be increased to improve linear solver convergence
  - Will require more memory to store the preconditioner

Running LFD

The LFD solver requires running the complex FUN3D executable:

- `mpirun complex_nodet_mpi --aeroelastic_internal`

The (number_of_modes_perturbed) x (number_of_lfd_frequencies) LFD linear systems are independent of each other

- FUN3D uses a loop of the number of modes as the inner loop, i.e. will loop over all the modes before moving to the next frequency
  - The LHS of the linear system is not a function of the perturbed mode, so we can reuse the LHS for a given frequency
- Given enough computational resources, you can split your LFD linear systems among different FUN3D analyses for simultaneous evaluation
  - Combine the GAFs into a single set as a post-processing step

LFD Screen Output

```
Begin Mesh Movement, Time Step 8   (Current Step 8)

 Recomputing distance function:
   Wall spacing: 0.472E-04 min, 0.940E-04 max, 0.912E-04 avg

Iter     8 LFD freq =     4 perturbed mode =     2
  linear matvecs =    1500   final res = (1.49777e-06,-3.76931e-07)   rate = (6.68285e-07,-1.68182e-07)

 Begin Mesh Movement, Time Step 9   (Current Step 9)

 Recomputing distance function:
   Wall spacing: 0.472E-04 min, 0.940E-04 max, 0.912E-04 avg

Iter     9 LFD freq =     5 perturbed mode =     1
  linear matvecs =    1500   final res = (1.29336e-08,4.64008e-08)   rate = (4.45022e-08,1.59657e-07)
```

LFD detailed output in {project}_sfe.out

- Useful to see if linear solver has stalled, still converging but needs a larger number of matvecs
- Actual residual orders of magnitude larger than GMRES estimated residual can indicate instability in the linear solver. Try q-ordering with lower prune width

```
 LFD solve for freq = 4 mode = 1
 Wall clock time for LFD RHS =    4.8246006484e+01
 Wall clock time for LFD LHS via operator overloaded operations using expression templates =    4.8190145409e+01
     0 Number of zero or negative diagonals  =         0         0         0         0         0         0
      0 max preconditioner application growth =  (1.5009381434e+02,9.3006267686e-02) rank =    80
Search direction      1 residual = (2.5336772656e-01,0.0000000000e+00) rate = (1.0000000000e+00,0.0000000000e+00)
Search direction     10 residual = (1.1633926634e-01,0.0000000000e+00) rate = (4.5917160767e-01,0.0000000000e+00)
Search direction     20 residual = (1.0496543091e-01,0.0000000000e+00) rate = (4.1428098336e-01,0.0000000000e+00)
Search direction     30 residual = (8.9977322670e-02,0.0000000000e+00) rate = (3.5512542932e-01,0.0000000000e+00)
Search direction     40 residual = (5.8538557591e-02,0.0000000000e+00) rate = (2.3104188677e-01,0.0000000000e+00)
Search direction     50 residual = (4.2648395371e-02,0.0000000000e+00) rate = (1.6832607669e-01,0.0000000000e+00)
Search direction     60 residual = (2.5748253132e-02,0.0000000000e+00) rate = (1.0162404455e-01,0.0000000000e+00)
Search direction     70 residual = (1.5696444610e-02,0.0000000000e+00) rate = (6.1951239107e-02,0.0000000000e+00)
Search direction     80 residual = (1.3025385356e-02,0.0000000000e+00) rate = (5.1409015398e-02,0.0000000000e+00)
Search direction     90 residual = (1.0098664624e-02,0.0000000000e+00) rate = (3.9857738634e-02,0.0000000000e+00)
...
Search direction    1460 residual = (6.2992445625e-08,0.0000000000e+00) rate = (2.4862063720e-07,0.0000000000e+00)
Search direction    1470 residual = (5.6486633968e-08,0.0000000000e+00) rate = (2.2294328775e-07,0.0000000000e+00)
Search direction    1480 residual = (5.2300589866e-08,0.0000000000e+00) rate = (2.0642167247e-07,0.0000000000e+00)
Search direction    1490 residual = (4.7875596947e-08,0.0000000000e+00) rate = (1.8895696621e-07,0.0000000000e+00)
Search direction    1500 residual = (4.3297264838e-08,0.0000000000e+00) rate = (1.7088705584e-07,0.0000000000e+00)
    0 Final Search direction    1500 residual = (4.3297264838e-08,0.0000000000e+00) rate = (3.3225392978e-08,1.5404109083e-07)
actual residual = (8.4182422831e-09,3.9029040981e-08) actual rate = (3.3225392978e-08,1.5404109083e-07)
```

LFD Outputs – *{project}_gafs.dat*

After each linear system, the corresponding GAFs are computed and written to a row of gafs.dat

- First two columns identify the LFD frequency and perturbed mode
- The rest of the row is pairs of real and imaginary parts of generalized force in each mode (2*number_of_modes columns of data), even if perturbing a subset of modes

```
ifreq | imode perturb | Re(GAF(jmode)) | Im(GAF(jmode))
1 1    3.0999872068e-02   -1.1369866212e+00    1.2186358720e-02    6.2447771062e-01
1 2   -6.1986908277e+02    1.0515572973e+00    3.4540543706e+02   -1.5947672522e+00
2 1   -3.7478600296e-02   -5.6799794263e+00    7.4814482836e-02    3.1195417455e+00
2 2   -6.1938917402e+02    5.1403140418e+00    3.4512785490e+02   -7.9088476717e+00   ...
3 1   -2.4478177499e-01   -1.1334011338e+01    2.6688166259e-01    6.2240859425e+00
3 2   -6.1815876420e+02    9.9227313047e+00    3.4441085581e+02   -1.5623174455e+01
4 1   -3.6364003147e-01   -1.3584741176e+01    3.7805284052e-01    7.4594977510e+00
```

Real and imag. parts of modal force in mode 1

Real and imag. parts of modal force in mode 2

Reading the GAFs

- The GAFs are written to the file as they are computed. The final is blocks of GAFs per frequencies and each block is the transpose of a typical GAFs matrix
- The following Python function will read the GAFs file into a complex-valued 3D array where the first index is the frequency index.

```
import numpy as np

def read_fun3d_lfd_gafs(gafs_filename):
    file_data = np.loadtxt(gafs_filename, skiprows=1)
    nmodes = (file_data.shape[1] - 2) // 2
    nfreq = file_data.shape[0] // nmodes
    gafs = file_data[:,2:].reshape(nfreq, nmodes, 2*nmodes)
    return (gafs[:,:,::2] + 1j * gafs[:,:,1::2]).transpose(0,2,1)
```

Other LFD Considerations

- LFD mode does not move the mesh from the `{project_rootname}.flow` file
  - You can remove or add modes (e.g., rigid body modes) that were not in the steady analysis
- Lack of machine precision convergence of the steady analysis makes it more difficult to solve the LFD linear problem
  - While it is possible to use a finite-volume solution as the background flow for LFD, it is not recommended because the finite-volume solution will not satisfy the SFE residual
- Recommend starting with smaller meshes
  - SFE's lower dissipation in means that means you typically can get the same accuracy as finite-volume on smaller meshes
  - Complex variables + large preconditioners can require significantly more memory than a typical steady analysis

Tutorial Case: BSCW LFD (1/4)

- The BSCW is known to flutter at low reduced frequency, so the LFD frequencies are based on the structural frequencies.

moving_body.input:

```
&aeroelastic_modal_data
  nmode(1)   = 2
  uinf    = 4508.4
  qinf    = 1.1722
  freq(1:2,1) = 20.92, 32.67
  gmass(1:2,1)  = 1.0, 1.0
  use_modal_deform = .true.
  modal_ref_amp(1:2,1) = 0.05, 0.05
  lfd_nfreq = 15
  lfd_freq(1) = 1.0
  lfd_freq(2) = 5.0
  lfd_freq(3) = 10.0
  lfd_freq(4) = 12.0
  lfd_freq(5) = 15.0
  lfd_freq(6) = 18.0
  lfd_freq(7) = 21.0
  lfd_freq(8) = 24.0
  lfd_freq(9) = 27.0
  lfd_freq(10) = 30.0
  lfd_freq(11) = 33.0
  lfd_freq(12) = 36.0
  lfd_freq(13) = 39.0
  lfd_freq(14) = 42.0
  lfd_freq(15) = 45.0
/
```

Tutorial Case: BSCW LFD (2/4)

fun3d.nml:

```
&global
   moving_grid = .true.
/
&governing_equations
   eqn_type = "compressible"
   viscous_terms = "turbulent"
   prandtlnumber_molecular = 0.755
   flow_solver = "sfe"
/
&reference_physical_properties
   temperature_units = "Kelvin"
   mach_number =        0.74
   reynolds_number =  278125.0
   temperature =    304.911111
   angle_of_attack = 0.0
/
&nonlinear_solver_parameters
   time_accuracy = "lfd"
/
&code_run_control
   steps = 30 ! 2 modes * 15 frequencies
   restart_read = "on_nohistorykept"
/
```

Tutorial Case: BSCW LFD (3/4)

sfe.cfg:

```
smoothing = .true.
max_matvecs = 1500
```

Run FUN3D:

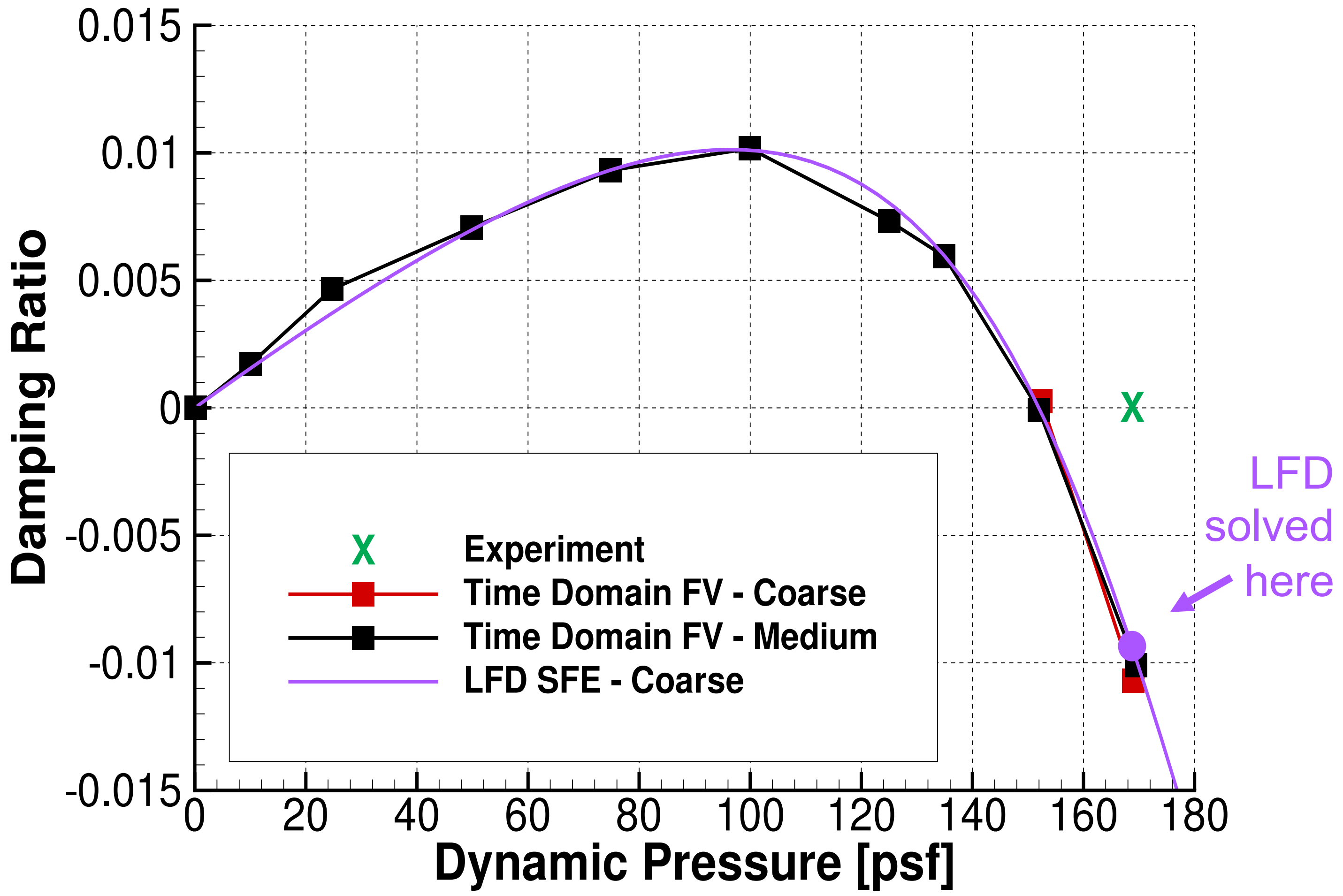
```
mpirun complex_nodet_mpi --gamma 1.136 --aeroelastic_internal
```

This LFD analysis takes about ~3.5 hours on 400 Skylake cores

Langley Research Center

Tutorial Case: BSCW LFD (4/4)

- Feed GAFs into p-k flutter solver (only evaluated GAFs at experiment dynamic pressure)



Visualizing the linearized pressure coefficient

- As with the SFE adjoint, the LFD solution cannot be visualized with the typical FUN3D sampling namelists
  1. In sfe.cfg, set **`write_solb = .true.`** – will write the linearized solution to a solb file: **`[project_rootname]_lfd_[step].solb`**
     - The solb file contains the real part of the linearized state, then the imaginary part.
       - Recall SFE's 5th state is temperature
  2. Run **`ref visualize [mesh file] [project_rootname]_lfd_[step].solb [output].{dat,plt,tec}`**
  3. Combine the linearized state with the steady state to compute linearized pressure coefficient.

$$c_p' = \frac{2}{\gamma M_{ref}^2}\left(\rho' T + \rho T'\right)$$

What We Learned

- Static aeroelastic analysis with SFE
  - Timestep coupling
  - Solver coupling
- LFD analysis with SFE

LFD Reference Paper:

- “Flutter Analysis with Stabilized Finite Elements based on the Linearized Frequency-domain Approach” K.E. Jacobson, B.K. Stanford, S.L. Wood, W.K. Anderson, AIAA SciTech Forum, 2020.

Public Community Questions: fun3d-users@lists.nasa.gov
Private/Proprietary Questions: fun3d-support@lists.nasa.gov