Tau-Code: Examples for Unsteady Simulations

Revision: 1.1

January 21, 2003

Contents

1 Hints for unsteady simulations

This user-guide is aimed at describing the usage of the unsteady features of the TAU code. It is far away from being complete, and not all features possible are described. The authors try to explain the usage of the code by some basic examples like a pitching oscillation. The main parameters of these examples will be highlighted and some of the numerical results are plotted, like lift over time. All input files necessary to test the examples on your own (or as a template for your own numerical experiments) are contained in the same tar-file as this user-guide.

1.1 Example-file locations

As mentioned above, all input files needed are available in the tar-file `TAU-unsteady-example.tar.gz`. If you want to play with the example files, create a directory, e.g. TAU-unsteady-example, move the tar file to this directory, and change to this directory. Then simply type

```
gunzip TAU-unsteady-example.tar.gz
tar xvf TAU-unsteady-example.tar
```

You will find a Makefile, a small program `makeconv.c`, this document (`user-guide-unsteady.ps`) and a subdirectory `NACA0012`. The program `makeconv.c` is a primitive C-program which creates some convergence output files in tecplot format using the TAU-standard output or log-files. Please note that `makeconv.c` is not part of the TAU-distribution (and not supported). If you want to use the Makefile, you have to adapt the path to the TAU binaries, for example

```
TAU_BIN=/beaaix1a/TAU_DIR/taubin_release.2003.1.0/bin_SGI_64_MPICH
```

The examples (and the Makefile) have been tested with the binaries of the TAU-release 2003.1.0 for the SGI_64_MPICH platform.

1.2 Pitching oscillation

As a first example we want to compute the pitching oscillation of a NACA0012 airfoil. From an experiment we know the Mach number (0.755), the reduced frequency (k = 0.1628) and the amplitude (2.51 degrees) of the oscillation, as well as the temperature in the wind tunnel (278.75 K). The profile is oscillating around its quarter point, the mean angle of attack is 2.0 degrees.

$$\alpha = 2.0^o + 2.51^o \sin(\omega * t)$$

where the angular velocity ω is part of the definition of the similarity parameter reduced frequency k:

$$k = \frac{\omega * l_k}{q_\infty}$$

l_k is the characteristic length used for calculating the reduced frequency. Usually the chord length is taken. q_∞ can be calculated using the Mach number. In our case the velocity

$$q_\infty = Ma_\infty a_\infty = Ma_\infty \sqrt{\gamma R T_\infty}.$$

With $R = 287 J/(kg * K)$ we get

$$q_\infty = 252.67[m/s].$$

This value is used by the TAU-code to calculate the physical timestep size. We come to that later. To explain the usage of the code, we will compute the flow in different ways with two different (but similar) meshes. Figure 2 shows the surface of the NACA0012 meshes (grid-1 and grid-2). grid-2 is created by scaling grid-1 by a factor of 10 and translating it 2 grid units in x and 1 grid unit in z direction. In the following we assume that 1 grid unit is equivalent to 1 [m]. Figure 1 shows the symmetry plane of grid-1. grid-1 and grid-2 can be found in the directory `NACA0012`. The names of the primary meshes are `naca.grid` and `naca-small.grid`.

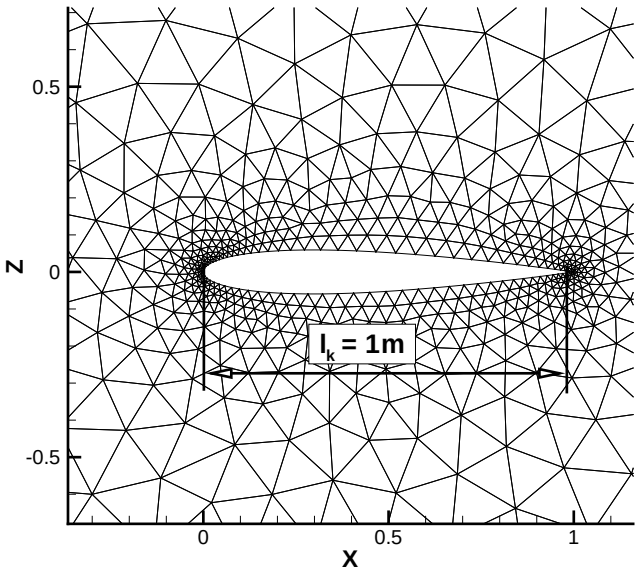

Figure 1: grid-1

For the calculation of the pitching oscillation it is useful to begin with a steady computation for the mean angle of attack. Here are some of the parameters of the file `para_steady` (grid-1 is used!):

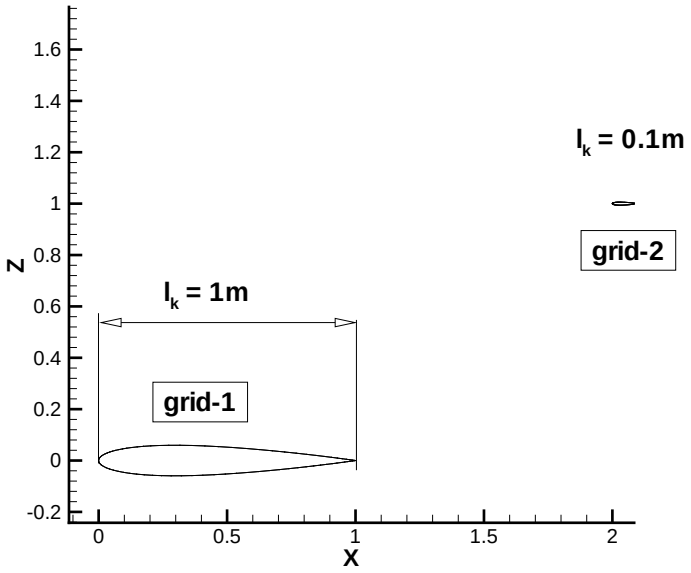

Figure 2: Profiles of grid-1 and grid-2

```
                                  Angle alpha (degree): 2.0
                                 Reference Mach number: 0.755
 . . .
                                      Minimum residual: 1e-5
#    the computation shall be stopped after a
#    convergence of 5 orders of magnitude of
#    the density residual
 . . .
                               Reference relation area: 0
#              area is calculated by the solver . . .
                 Reference length (pitching momentum): 1
#    the chord length is 1 for naca.grid (grid-1). the
#    chord length is usually taken as reference
#    length for the moment . . .
                                   Origin coordinate x: 0.25
                                   Origin coordinate y: 0
                                   Origin coordinate z: 0
#    the quarter point is the moment reference point
    Dual time -------------------------------------: -
                          Unsteady calculation (0/1/2): 0
#    0 means: perform a steady computation
```

```
 . . .
                                  Reference temperature: 278.75
# not really important for an euler computation . . .
```

If you were to use the small and translated grid-2 the following parameters would have to be changed, to get the same history with respect to the pitching moment:

```
                                      Reynolds length: 0.1
#    of course only for Navier-Stokes computations . . .
#    for the current data set there is no influence
                Reference length (pitching momentum): 0.1
                                   Origin coordinate x: 2.025
                                   Origin coordinate y: 0
                                   Origin coordinate z: 1
#    these are the coordinates of the quarter point of
#    the scaled and translated grid-2
```

To perform all necessary steps for the steady computation using grid-1, you can use the Makefile. Simply type

```
make steady
```

This automatically links the primary mesh and copies the parameter file `para_steady` to the working directory `NACA0012/Work`. Then the preprocessor is called to create the dual meshes followed by the run of the solver.

```
ptau3d.preprocessing para_steady
ptau3d.el para_steady output-steady
```

The standard output of the solver is redirected to the file `output-steady.solver.stdout` To visualize for example the symmetry plane of the converged solution in tecplot format, the postprocessing tool `tau2plt` is called:

```
tau2plt -x -o sym-steady.plt -m"1" naca.grid steady-solution.pval
```

The convergence history in tecplot format is extracted from `output-steady.solver.stdout` using the command

```
makeconv < output-steady.solver.stdout > conv-steady.plt
```

Figure 3 shows the density distribution on the symmetry plane and figure 4 the convergence history of the residual and the lift. For grid-2 you would get exactly the same convergence history. This you can verify by typing

```
make steady-small
```

In that case the parameter file `para_steady-small` is used. The code is executed in the directory `Work-small`.

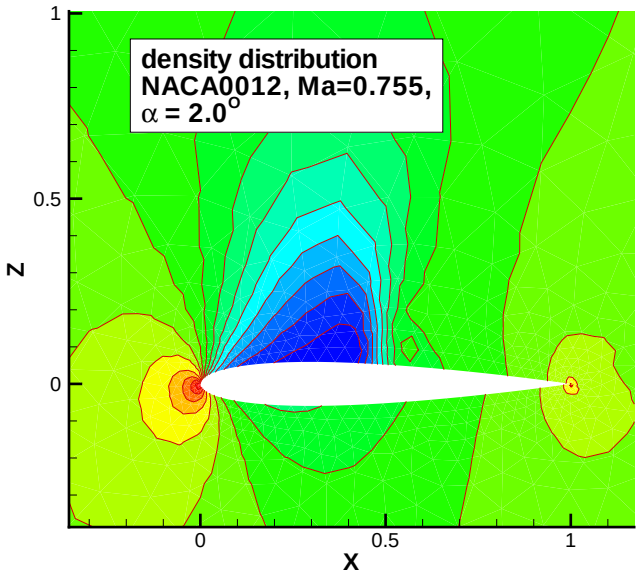

Figure 3: density distribution of steady computation using grid-1

Now we can start to perform the unsteady computation, using the steady solution for the mean angle of attack ($\alpha = 2.0^o$). The corresponding parameter file is `para_periodic-r`. At the end of the parameter file you will find the entry:

```
Restart-data prefix: steady-solution.pval
```

This ensures that the restart file `steady-solution.pval` is read in at the beginning of the computation. (If you have used the Makefile for creating the steady solution, the original file `solution.pval.84` is moved to the name `steady-solution.pval`. This helps you to mark the initial solution, which migth be used later on for other computations, for example with another reduced frequency. If you don't want to use the make file, you have to adapt the line to

```
Restart-data prefix: solution.pval.84
```

if the tolerance criterion is reached after 84 multigrid cycles.
Now there are several other parameters which have to be adapted or added to the parameter

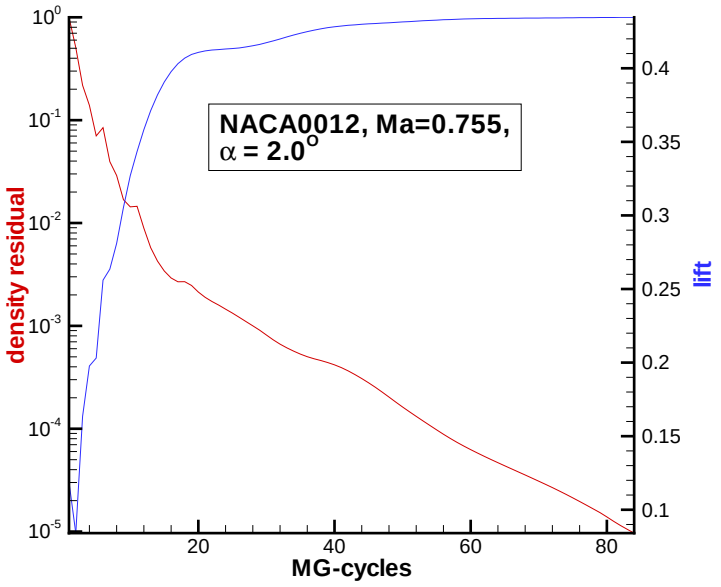

Figure 4: convergence history of steady computation using grid-1

file. For unsteady computations we use the dual time-stepping scheme of Jameson. In the following you'll find the corresponding parameters (taken from `para-periodic-r`)

```
Dual time ---------------------------------------: -
                    Unsteady calculation (0/1/2): 1
          Unsteady implicit scheme order (1/2/3): 2
           Unsteady show pseudo time steps (0/1): 1
Minimum number of inner iterations per time step: 5
         Unsteady inner iterations per time step: 200
                    Unsteady physical time steps: 150
                    Unsteady residual type (0/1): 0
```

To switch on the dual time-stepping, `Unsteady calculation (0/1/2):` is set to 1 (0 means steady computation, 2 means explicit integration). For unsteady computations currently only the dual time-stepping should be used!! The temporal order can be selected with the parameter `Unsteady implicit scheme order (1/2/3):`. The default is 2 and this is usually a good compromise of accuracy and memory needed (for third order an additional time level has to be kept in memory). To control the convergence of the inner iterations, the parameter `Unsteady show pseudo time steps (0/1):` should be set to 1. In that case the convergence information of the inner iterations is printed to stdout. A physical timestep is finished if the tolerance criterion is reached

```
                                    Minimum residual: 1e-5
```

or the maximal number of inner iterations allowed is reached, which is set with the parameter `Unsteady inner iterations per time step:`. To use the same normalization of the residual as for a steady computation, set the switch `Unsteady residual type (0/1):` to 0. The number of physical timesteps to be performed is set with `Unsteady physical time steps:`. Here we select 150. We want to resolve each oscillation period with 50 timesteps, so in total 3 periods will be computed. For a periodic motion the timestep size is not specified. It is computed automatically using the reduced frequency (we come to that later)
To describe the movement of the profile the `Grid movement` part has to be added to the parameter file:

```
Grid movement ----------------------------------: -
                            Type of grid movement: 2
                          Motion hierarchy filename: (thisfile)
                       Motion description filename: (thisfile)

                                         Node name: naca0012
                                Node reference frame: inertial
                             Node controls grid block: 1
                           Node motion description id: aircraft
                                          hdf end

                      Number of timesteps per period: 50

                                Motion description id: aircraft
                                     Type of movement: periodic
                  Origin of local coordinate system: 0.25  0.0 0.0
            Degree of Fourier series for rotation: 1
                   Reduced frequency for rotation: 0.0 0.1628  0.0
               Reduced frequency reference length: 1.0 1.0     1.0
    Fourier coefficients for rotation (cos) pitch: 0.0 0.0
    Fourier coefficients for rotation (sin) pitch: 0.0 2.51
```

If the parameter `Type of grid movement:` is set to 2, as in our example, the rigid-body motion capabilities of the code are enabled. The parameters beginning with the word *Node* define the hierarchy tree for the given simulation, and the parameter `Motion description id:` is the connection between hierarchy tree and motion description (see the Tau User-Guide for further details). If the parameter `Type of movement:` is set to *periodic*, a periodic simulation is performed (other grid movement types will be described later). In that case, the timestep size is calculated automatically using the parameter `Number of timesteps per`

`period:`. For a pitching oscillation, a reference point for the rotation of the grid has to be specified. Here the profile shall pitch around the moment reference point, which is equal to the quarter point. The reference point is set using the parameter `Origin of local coordinate system:`. The oscillation of the angle of attack of the profile

$$\alpha = 2.0^o + 2.51^o \sin(\omega * t)$$

is specified in the TAU code by Fourier-series. A Fourier serie for three different angles (*yaw, pitch, roll*) can be specified. For a pitching oscillation the rotation is parallel to the y-axis. The rotation is specified by the *pitch* angle (we come to the other angles later, see subsection 1.5). For each Fourier serie of the angles we can specify a reduced frequency `Reduced frequency for rotation:`. The second row is associated to the *pitch* angle. It is set to 0.1628, the value we know from the experiment. The others, which are not used, are set to 0.0. For a pitching oscillation a degree of 1 is sufficient for the Fourier serie.

$$\sum_{k=0}^{N_{FR}} (c_k \cos(k\omega t) + d_k \sin(k\omega t))$$

The coefficients c_k and d_k are associated with the parameters `Fourier coefficients for rotation (cos) pitch:` and `Fourier coefficients for rotation (sin) pitch:`. For each reduced frequency a corresponding reference length has to be set. In our case the chord length has to be used which is 1 (see figure 1). To compute the timestep size, the TAU code uses the definition of the reduced frequency (see above) and of the angular velocity $\omega = 2\pi/T_{period}$, so

$$\Delta t = \frac{T_{period}}{50} = \frac{2\pi}{50\omega} = \frac{2\pi}{50}\frac{l_k}{kq_\infty} = 0.00305489s$$

If you want to plot for your unsteady computation the time history of the lift and drag, or perhaps the lift over the angle of attack, then you should adapt the monitoring list:

```
Monitoring values: Residual_C-lift_C-drag_Angle-a_T/tperiod
```

The additional entry in the monitoring list (`T/tperiod`) is useful, if you want to plot the global forces over the time nondimensionalized with the period length.

Now we are ready to start the unsteady computation. If you want to use the Makefile, simply type

```
make periodic-r.
```

Then automatically the TAU code is started, performs three oscillation periods (150 timesteps). Afterwards tecplot files for the time and convergence history are created (`conv-periodic-r.plt` and `time-periodic-r.plt`).Before we look at the results, let's have a look to a part of the standard output (stored in the file `output-periodic-r-solver.stdout`, if the Makefile is used):

```
Initial residual was:  1.549e+01
Starting mainloop  on Level 1

-------------------------------------------------------------------------
 Lift/Drag Referenz Area: 0.996822 [grid units] 0.996822 [m^2]
-------------------------------------------------------------------------
 Time scheme: DUAL_TIME
-------------------------------------------------------------------------
          Time|   Residual|     C-lift|     C-drag|    Angle-a|  T/tperiod|
-------------------------------------------------------------------------
!          0|  7.937e-02|  4.411e-01|  1.802e-02|  2.315e+00|  2.000e-02|
!          1|  5.841e-02|  4.604e-01|  1.890e-02|  2.315e+00|  2.000e-02|
!          2|  3.351e-02|  4.672e-01|  1.910e-02|  2.315e+00|  2.000e-02|
!          3|  2.101e-02|  4.702e-01|  1.918e-02|  2.315e+00|  2.000e-02|
!          4|  2.605e-02|  4.717e-01|  1.930e-02|  2.315e+00|  2.000e-02|
!          5|  1.520e-02|  4.718e-01|  1.931e-02|  2.315e+00|  2.000e-02|
!          6|  8.227e-03|  4.717e-01|  1.926e-02|  2.315e+00|  2.000e-02|
!          7|  5.181e-03|  4.716e-01|  1.925e-02|  2.315e+00|  2.000e-02|
!          8|  2.973e-03|  4.716e-01|  1.925e-02|  2.315e+00|  2.000e-02|
!          9|  2.028e-03|  4.715e-01|  1.926e-02|  2.315e+00|  2.000e-02|
!         10|  2.062e-03|  4.714e-01|  1.925e-02|  2.315e+00|  2.000e-02|
!         11|  1.532e-03|  4.713e-01|  1.925e-02|  2.315e+00|  2.000e-02|
!         12|  1.100e-03|  4.713e-01|  1.924e-02|  2.315e+00|  2.000e-02|
!         13|  9.807e-04|  4.712e-01|  1.923e-02|  2.315e+00|  2.000e-02|
!         14|  8.993e-04|  4.712e-01|  1.923e-02|  2.315e+00|  2.000e-02|
!         15|  8.095e-04|  4.712e-01|  1.923e-02|  2.315e+00|  2.000e-02|
!         16|  7.127e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         17|  6.249e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         18|  5.333e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         19|  4.354e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         20|  3.521e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         21|  2.943e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         22|  2.532e-04|  4.711e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         23|  2.190e-04|  4.710e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         24|  1.883e-04|  4.710e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         25|  1.608e-04|  4.710e-01|  1.922e-02|  2.315e+00|  2.000e-02|
|    3.055e-03|  1.369e-04|  4.710e-01|  1.922e-02|  2.315e+00|  2.000e-02|
!         27|  8.136e-02|  5.013e-01|  2.314e-02|  2.624e+00|  4.000e-02|
!         28|  5.557e-02|  5.003e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         29|  2.412e-02|  5.001e-01|  2.311e-02|  2.624e+00|  4.000e-02|
```

```
!         30|  1.327e-02|  4.997e-01|  2.312e-02|  2.624e+00|  4.000e-02|
!         31|  1.596e-02|  4.997e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         32|  9.132e-03|  4.996e-01|  2.308e-02|  2.624e+00|  4.000e-02|
!         33|  4.373e-03|  4.997e-01|  2.309e-02|  2.624e+00|  4.000e-02|
!         34|  2.523e-03|  4.998e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         35|  1.594e-03|  4.999e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         36|  1.083e-03|  4.999e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         37|  9.527e-04|  5.000e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         38|  7.574e-04|  5.001e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         39|  5.438e-04|  5.001e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         40|  4.522e-04|  5.001e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         41|  4.046e-04|  5.002e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         42|  3.972e-04|  5.002e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         43|  3.902e-04|  5.002e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         44|  3.612e-04|  5.002e-01|  2.310e-02|  2.624e+00|  4.000e-02|
!         45|  3.110e-04|  5.002e-01|  2.311e-02|  2.624e+00|  4.000e-02|
!         46|  2.503e-04|  5.002e-01|  2.311e-02|  2.624e+00|  4.000e-02|
!         47|  1.938e-04|  5.002e-01|  2.311e-02|  2.624e+00|  4.000e-02|
|    6.110e-03|  1.512e-04|  5.002e-01|  2.311e-02|  2.624e+00|  4.000e-02|
!         49|  2.872e-03|  5.293e-01|  2.720e-02|  2.924e+00|  6.000e-02|
!         50|  8.040e-03|  5.309e-01|  2.735e-02|  2.924e+00|  6.000e-02|
. . .
```

The '!' in the first row marks that convergence output (inner iterations) are shown. In that case the first number is not the time, it is the current number of multigrid cycles. If a timestep is finished (after reaching the tolerance criterion or the maximal number of inner iterations) information for the time history is given. In that case the output line starts with '|'. The first number is the corresponding physical time in seconds.
Figure 5 shows the history of the lift with respect to the time nondimensionalized with the period length and the lift over the angle of attack for grid-1 and grid-2 (files `time-periodic-r.plt` and `time-periodic-r-small.plt`). It is observed that 3 periods seem to be sufficient to reach a periodic solution. Figure 6 (file `conv-periodic-r.plt`) shows the convergence history of the residual and the lift for the first 10 timesteps. It is observed that all physical timesteps are well converged. The convergence criterion of 5 orders of magnitude for the density residual is always reached and (what is even more important) the lift is stable during the last iterations of a physical timestep (horizontal tangent). A useful hint for a proper selection of the convergence criterion is the convergence of the steady computation (compare figure 4). It is observed that the lift is stable, if the density residual has reached 5 orders of magnitude. So we select the same value for the unsteady computation (using the same normalisation as done for steady computations by setting the parameter `Unsteady residual type (0/1):` to 0. Figure 7 shows what happens, if the maximal number of inner iterations

is too low (only set to 5). It is obvious that there is still a gradient in the global forces like the lift at the end of a physical timestep.

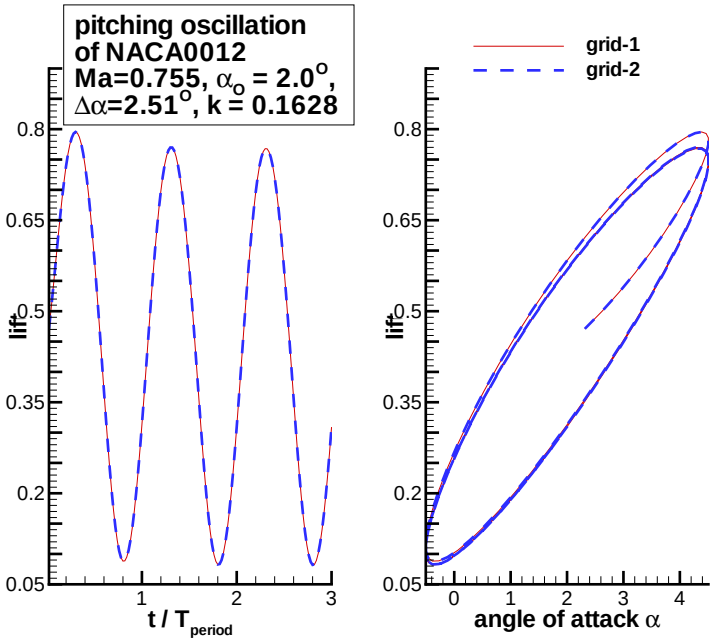

Figure 5: History of the lift (starting from steady solution)

What has to be changed, if instead of grid-1 the scaled grid-2 is used? Here are the parameters, which have to be adapted:

```
                    Origin of local coordinate system: 2.025  0.0 1.0
# this is the corresponding quarter point of grid-2
                  Reduced frequency reference length: 0.1    0.1 0.1
# . . . because the chord length of grid-2 is only 0.1,
# the similarity parameter 'reduced' frequency is for
# both computations the same
```

A corresponding parameter file is stored as `para_periodic-r-small`. To verify that the resulting solutions are really the same, simply use the Makefile again, and type:

```
make periodic-r-small
```

The resulting history of the lift over the time nondimensionalized with the period length and the angle of attack is plotted, together with the results for grid-1, in figure 5. It is shown that the results are the same, as they should be, if all similarity parameters are the same.
What has to be done, if you want to select the timestep size on your own? You only have

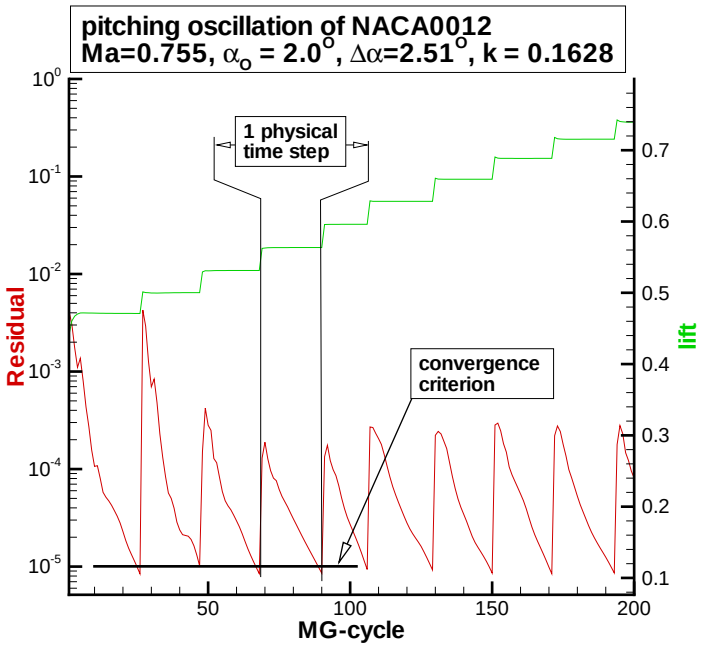

Figure 6: convergence history of the residual and the lift (1st 10 timesteps)

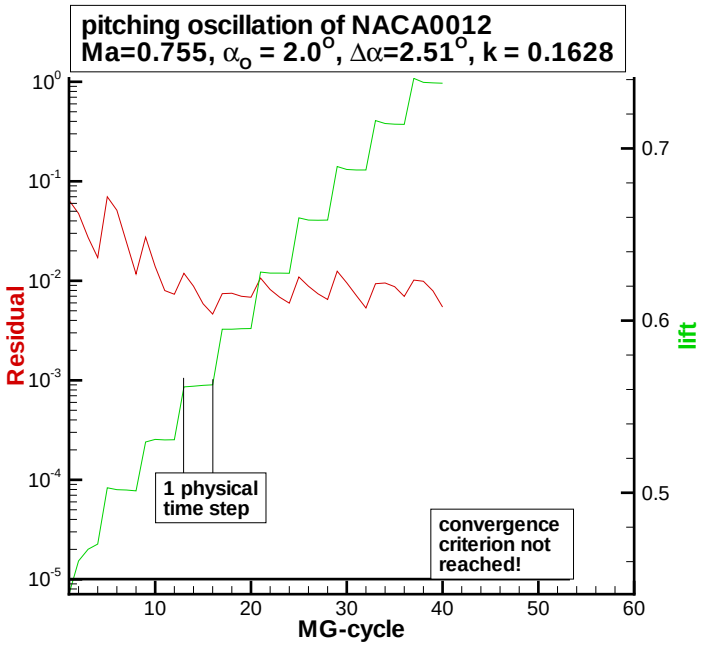

Figure 7: convergence history of the residual and the lift, if only 5 inner iterations are performed (1st 10 timesteps)

to switch the parameter `Type of movement:` from *periodic* to *rigid*. Then the parameter `Number of timesteps per period:` is not used. The following line has to be added in that case to the parameter file, if grid-1 is used:

```
Unsteady physical time step size:  0.00305489
```

The timestep size has to be specified in seconds (see parameterfile `para_periodic-r-nonperiodic`)! For grid-2 the physical timestep has to be reduced by a factor of 10 (for grid-2 the reduced frequency reference length l_k which is equal to the chord length is only 0.1), so you would have to set

```
Unsteady physical time step size:  0.000305489
```

Of course it is possible to start an unsteady computation without an initial steady computation. If you want to use the Makefile, simply type

```
make periodic-s
```

Then the TAU code is started using the parameter file `para_periodic-s` from scratch. The only difference to `para_periodic-r` is the missing last line for the `Restart-data prefix:`. Again three periods are calculated. Figure 8 shows the lift versus t/T_{period}. It becomes obvious, that the solution is not periodic after 150 timesteps.

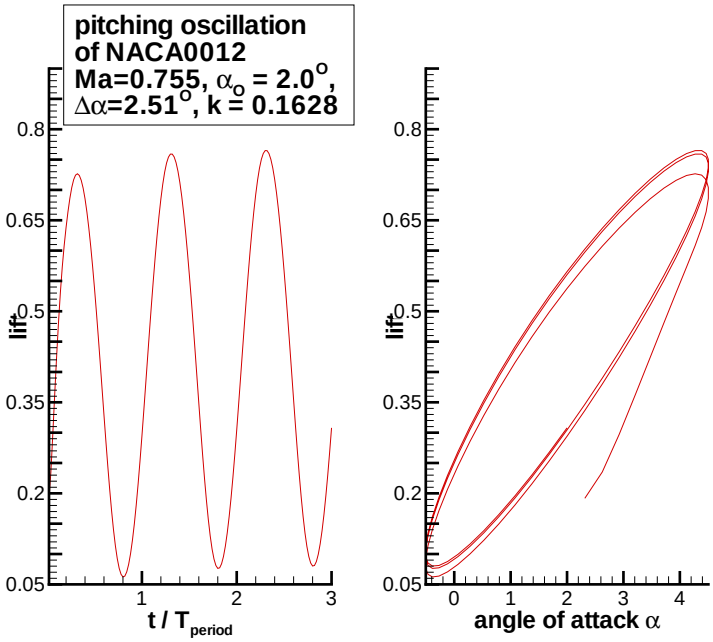

Figure 8: History of the lift (starting from scratch)

If you want to add two more periods, you can adapt the parameter file in the working directory. So change to `NACA0012/Work` and adapt the line

```
Unsteady physical time steps: 150
```

to

```
Unsteady physical time steps: 100
```

Then start the Tau code again

```
ptau3d.el para_periodic-s output-periodic-s-2ndrun
```

TAU reads in the solution file from the last run (the parameter file is automatically updated, because the parameter `Automatic parameter update (0/1):` is set to 1. So the line

```
Restart-data prefix: solution.pval.unsteady_i=150_t=4.5823e-01
```

is added to `para_periodic-s`. Figure 9 shows now the history of the lift of the 4th and 5th period. The closed ellipse of the lift hysteresis shows that the solution is now really periodic.

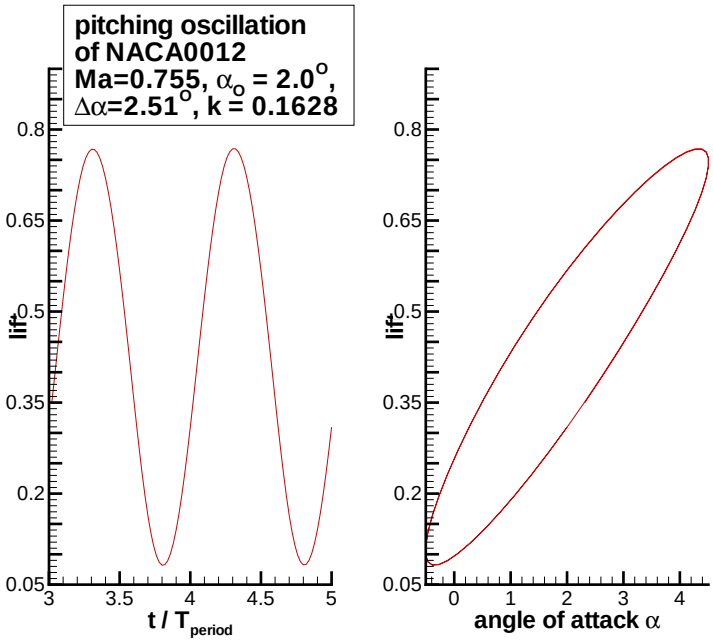

Figure 9: History of the lift for 4th and 5th period

If you want to create a movie or a Fourier analysis of for example the cp distribution on the profile, it is useful to save the solution after each timestep. To do this, change the parameters

```
Surface output period: 99999
        Output period: 99999
```

to

```
Surface output period: 1
        Output period: 1
```

Then after each timestep has finished, a restart file and a surface output file is written out, for example:

```
solution.pval.unsteady_i=162_t=4.9489e-01
solution.surface.pval.unsteady_i=162_t=4.9489e-01
```

The integer number in the file name is the number of the physical timestep, the float number is the corresponding physical time in seconds. If you want to see the rotation of the profile in the movie, then you should add `xyzgeod` to the parameter `Field output values:` (`xyzgeod` stands for the geodesic coordinates.) Figure 10 shows a solution after 162 timesteps, where the angle α is close to the maximum value of 4.51 degrees and the corresponding cp distribution on the profile.

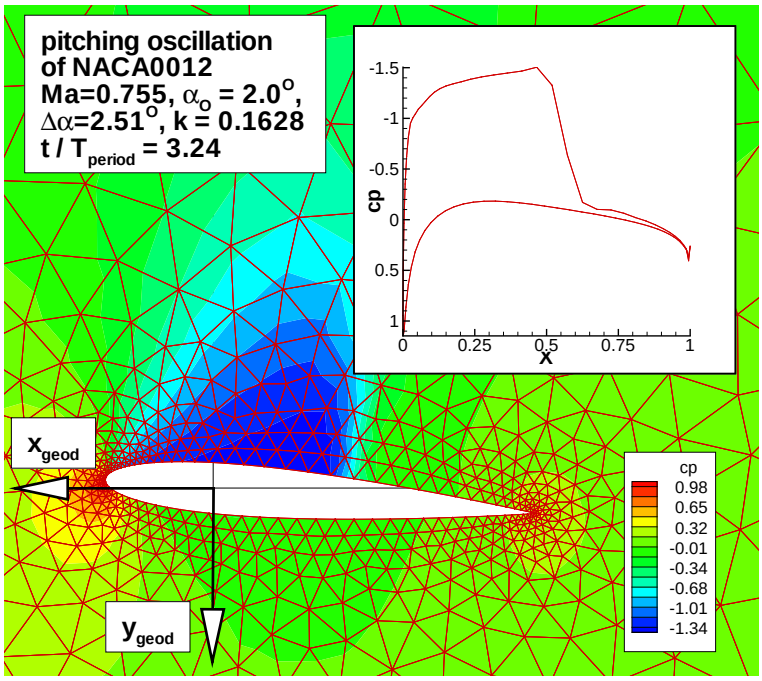

Figure 10: time slice for $t/T_{period=3.24}$

1.3 Flapping oscillation

To compute a flapping oscillation, only a few parameters have to be changed compared to the pitching oscillation. Here we have to specify a periodic translation of the z-coordinate instead of an angle. To perform three periods of a flapping oscillation with the same reduced frequency

and an amplitude of 0.05 chord length, you can use the parameter file `para_periodic-r-flap`. The main change is that the parameters for the Fourier-series for the prescription of the *yaw* angle are replaced by:

```
            Degree of Fourier series for translation: 1
                   Reduced frequency for translation: 0.0 0.0 0.1628
                  Reduced frequency reference length: 1.0 1.0 1.0
Fourier coefficients for translation (cos) z: 0
Fourier coefficients for translation (sin) z: 0.0 -0.05
```

Additionally the monitor list is adapted, because we are now more interested in the variation of the translation in z-direction.

```
Monitoring values: Residual_C-lift_C-drag_Trans-z_T/tperiod
```

To perform the computation, you can again use the Makefile. Simply type

```
make periodic-r-flap
```

Figure 12 shows the history of the lift over the time normalized with T_{period} and over the deflection Δz. Please note that the translation is prescribed in the body-fixed coordinate system of the reference frame (in this case the inertial frame), where x and z point in the opposite direction compared to the standard TAU definition (see figure 11)

1.4 'Rotating' airfoil

This is a so called 'equivalent' test case from our unsteady test matrix. 'Equivalent' in this context means: Creating the same solutions following different ways. For example we can compute the steady flow field around a NACA0012 for a Mach number of 0.755 and an angle of attack of 0.0^o (To get a reference solution, simply edit the file `para_steady` and set the angle of attack to 0.0^o). We should be able to obtain the same results by specifying a rotation around a reference point which is far away from the profile. The Mach number is set to 0.0. The translation or rotation of the airfoil has to be adapted in such a way, that the speed of the airfoil is equal to the onflow velocity prescribed by the Mach number of the reference test case. We will rotate the NACA0012 around a reference point located 1000 chord lengths (=1000m) below the quarter point. The angular velocity of the rotation is now set in such a way that the velocity $q_{rot} = \omega * R_0$ of the profile is equal to the velocity q_∞ of the reference solution or $Ma_\infty = Ma_{rot}$. R_0 is the distance from the rotation reference point (1000m). So we get for ω

$$\omega = \frac{q_{rot}}{R_0} = \frac{q_\infty}{R_0} = \frac{Ma_{rot} sqrt(\gamma R T_\infty)}{R_0} = 0.25267 rad/s = 14.477 degree/s$$

The rotation of the profile can be prescribed by a polynom of the form

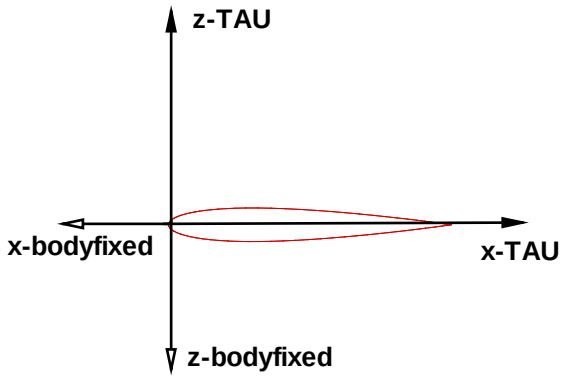

Figure 11: Standard TAU and body-fixed coordinate system

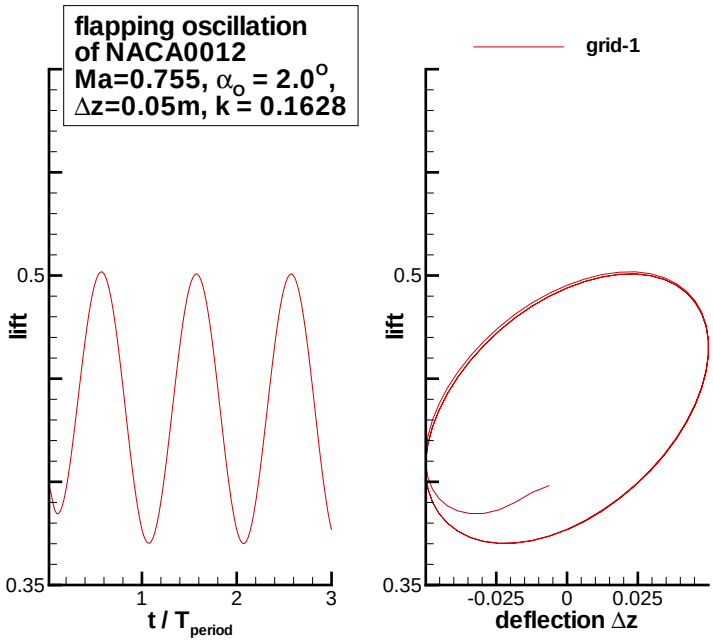

Figure 12: history of the lift for a flapping oscillation

$$\psi(t) = \sum_{k=0}^{N_{FR}} (\psi_k t^k)$$

For our simple example we only need a polynom of first order

$$\psi(t) = \psi_1 t$$

So the coefficient ψ_1 is equal to the absolute value of ω. We have to select a negative value in order to rotate the profile in the correct direction (compare Figure 13). For one turn around of the profile the time

$$T = \frac{2\pi}{\omega} = 24.8668s$$

is needed. To resolve one turn around with 100 timesteps, we select $\Delta t = 0.248668$ s. We will see that only 10 physical timesteps are needed to reach a steady solution (REMARK: the solution is steady in the rotating frame, it is unsteady with respect to the inertial coordinate system!! In one of the next releases it will be possible to perform a steady computation directly in a rotating frame, which will be important for helicopter or turbomachinery applications). In this test-case we also demonstrate the ability to build up a specific motion by combining multiple motion descriptions using the motion hierarchy. Here are some of the main parameters of the parameter file `para_rotate`.

```
    Grid movement ----------------------------------: -
                                Type of grid movement: 2
                             Motion hierarchy filename: (thisfile)
                           Motion description filename: (thisfile)
                          ----------------------------: -
                                           Node name: geodesic
                                 Node reference frame: inertial
                              Node controls grid block: -1
                             Node motion description id: virtual
                                              hdf end


# the 'geodesic' node is specified as a
# virtual node (not connected to a grid-block) by
# setting 'Node controls grid block:' to -1
```

```
                          ----------------------------: -
                                           Node name: aircraft
                                  Node reference frame: geodesic
                                Node controls grid block: 1
                               Node motion description id: aircraft
                                                 hdf end
# the 'aircraft' node uses the 'geodesic' node
# as its reference frame
                          ----------------------------: -
                        Number of timesteps per period: 50
                          ----------------------------: -
                                 Motion description id: aircraft
                                      Type of movement: rigid
                     Origin of local coordinate system: 0.25 0.0 0.0
                     Degree of polynomial for rotation: 1
              Polynomial coefficients for translation z: -1000.0
                                                 mdf end
# the origin of the airfoil is set at the 1/4-chord
# and the airfoil is set at a location 1000m above
# the origin
                          ----------------------------: -
                                 Motion description id: virtual
                                      Type of movement: rigid
                     Origin of local coordinate system: 0.25 0.0 0.0
                     Degree of polynomial for rotation: 1
            Polynomial coefficients for rotation pitch: 0.0 -14.4771138
                                                 mdf end
# the parent-frame of the airfoil is set at the
# same origin as the airfoil, and then a rotation
# is specified for the parent-frame, which will be
# inherited by the child-frame (aircraft).

# pitch        = pitch_0 + pitch_1 * t + pitch_2 * t^2 + . . .
#              =        -14.477 * t
# dpitch / dt  = pitch_1 + 2 * pitch_2 * t + . . .
# ==> abs(omega)  = pitch_2 = 14.477 degree / s ~ 0.25267 rad / s
# ==> q_rot    = abs(omega) * 1000 = 252.67 m/s
# ==> Ma_rot   = q_rot / a_inf = q_rot / sqrt(1.4 R * T_inf)
#              = 252.67 / sqrt(1.4 * 287 * 278.75) = 0.755
# ==> T_360_deg= 2 pi / abs(omega) = 24.8668 s
```

```
Dual time ---------------------------------------: -
                      Unsteady calculation (0/1/2): 1
             Unsteady show pseudo time steps (0/1): 1
                  Unsteady physical time step size: 0.2486683499
                      Unsteady physical time steps: 10
            Unsteady inner iterations per time step: 200
                      Unsteady residual type (0/1): 0
```

Again the Makefile can be used to perform this test case. Simply type

```
make rotate
```

Figure 13 shows the comparison of the density distribution for the reference case and for the rotating airfoil. As expected, the density contours are very close together.
REMARK 1: For the rotating airfoil the reference Mach number is 0. So it is not possible to plot cp, lift, drag and other values using the dynamic pressure for normalization!! Instead you can plot the static pressure and the forces and moments without normalization.
REMARK 2: This example can (currently) not be used for viscous computations, because q_∞ is used for the specification of the Reynolds number. To get correct results q_{rot} would have to be used. This will be changed in one of the next patches or releases.

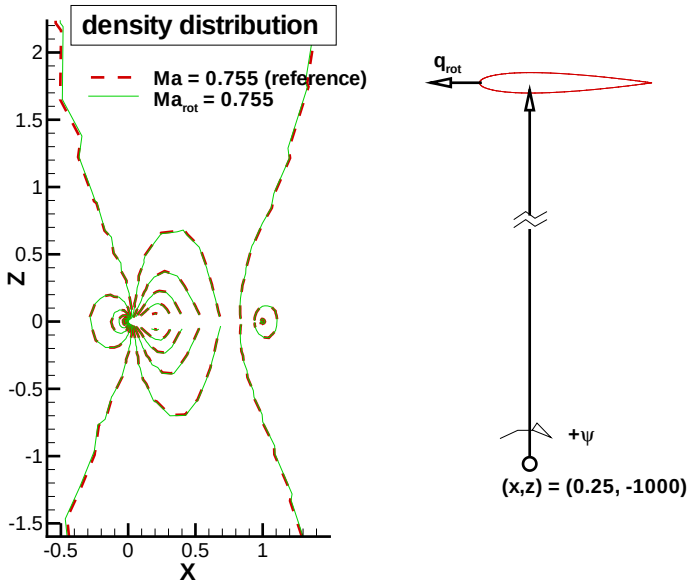

Figure 13: Comparison of steady references solution ($Ma_\infty = 0.755$) and unsteady equivalent solution for $Ma_{rot} = 0.755$

1.5 The rotation angles

In the following section the prescription of rotations within the TAU-code will be described in greater detail. As mentioned above, three different rotation angles, *yaw*, *pitch* and *roll*, can be specified. The sequence of the rotations is yaw - pitch - roll. The following figure illustrates the effects of applying the angles to a NACA0012 wing. The angles are associated with the body-fixed coordinate system. The *yaw* angle (ξ) is associated with the body-fixed z-axis, *pitch* (ψ) with the body-fixed y-axis, and *roll* (ϕ) with the body-fixed x-axis. Please note, that the body-fixed coordinate system is not equal to the standard TAU coordinate-system.

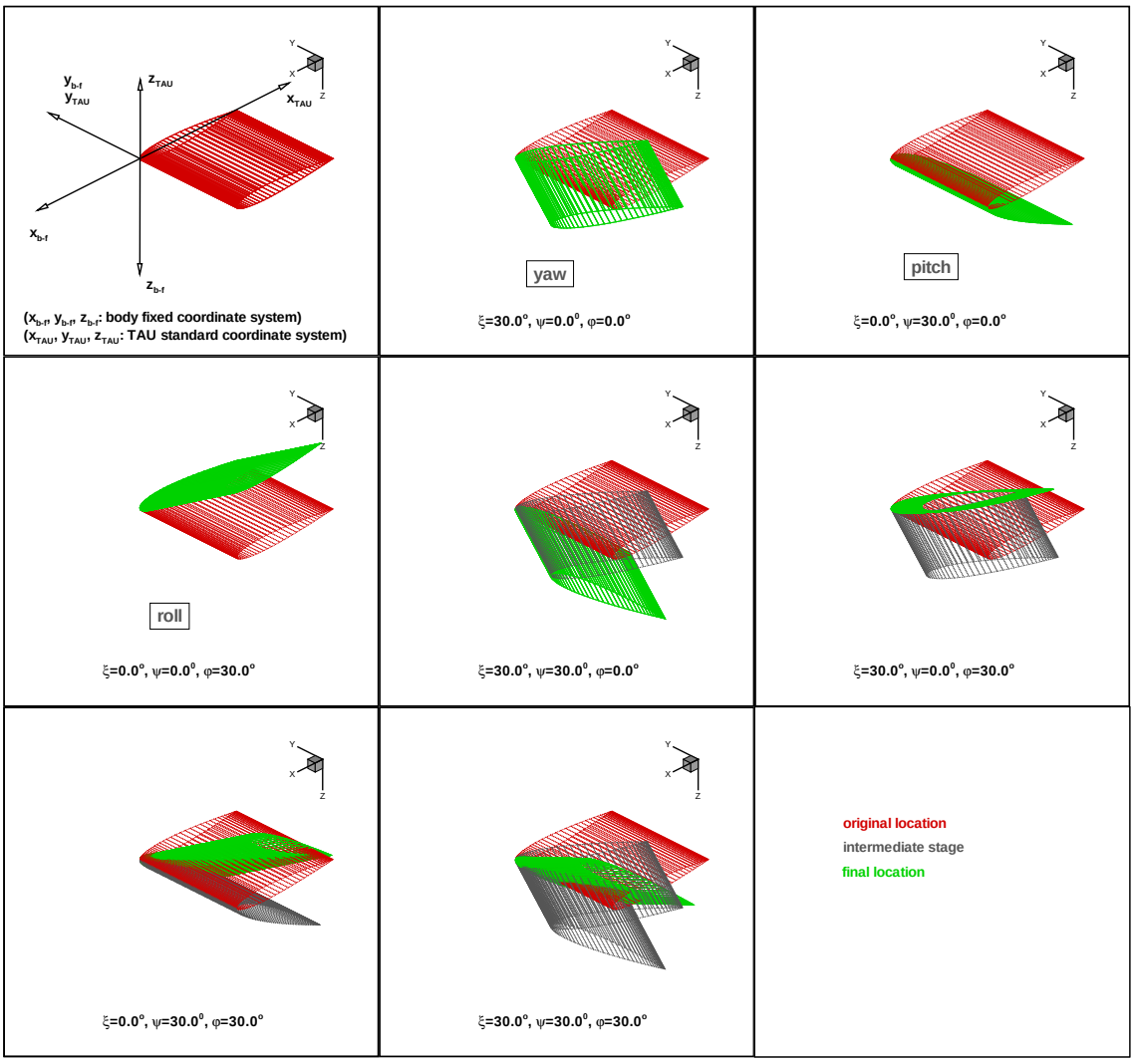

Figure 14: Effects of rotation angles

For each angle you can specify a polynom and a Fourier serie. The value of the angle is the sum of the value of the polynom and the Fourier serie. The names in the parameter file

associated to ξ, ψ and ϕ are `yaw`, `pitch` and `roll`. This is an example for the specification of the *yaw* angle (ξ):

```
          Origin of local coordinate system: 0.25 0.0 0.0
     Degree of Fourier series for rotation: 1
            Reduced frequency for rotation: 0.0   0.1628  0.0
        Reduced frequency reference length: 1.0   1.0     1.0
Fourier coefficients for rotation (cos) yaw: 0.0   0.00
Fourier coefficients for rotation (sin) yaw: 0.0   2.51
         Degree of polynomial for rotation: 1
  Polynomial coefficients for rotation yaw: 0.0  -1.0
```