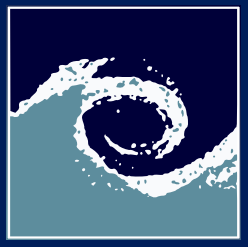


Advaced post-processing

Open-Source CFD Course 2021 – Lecture 5

Miklós BALOGH
and
Josh DAVIDSON

2021

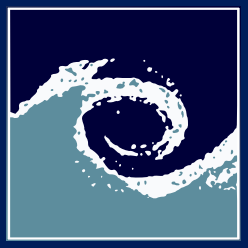
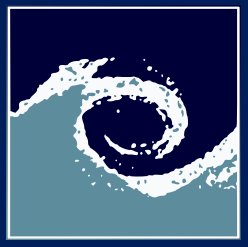
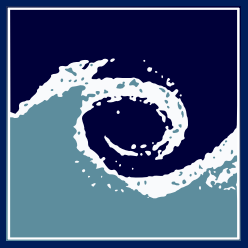


Table of Contents



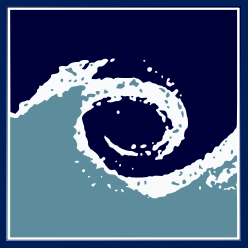
Why is it important?

- Post-processing CFD data is the key to
  - derive the right conclusions from the numerical output,
  - convert the output into interpretable form,
  - present the results to decision-makers.
- Interpretable form
  - graphs for quantitative analysis,
  - images and videos for qualitative analysis.
- Types
  - conventional post-processing occurs after a simulation,
  - run-time processing is performed during the simulation.



Tools

- External post-Processing utilities
  - gnuplot, octave and python for graphs,
  - paraView (paraFoam) for images,
  - paraView and ffmpeg for videos.
- Builtin functionality for derived fields and quantities
  - postProcess utility,
    - example 1: postProcess -func 'vorticity'
    - example 2: simpleFoam postProcess -func 'yPlus'
  - controlDict entries for run-time post-processing,
  - noise utility for aeroacoustics.



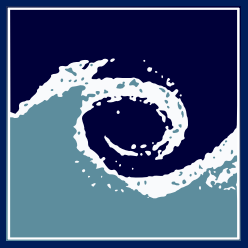
Run-Time Post-Processing

Template for controlDict entry

```
functions
{
    <user-defined name>
    {
        type        <object type>;
        libs        (<list of library names>);
        ...
    }
}
```

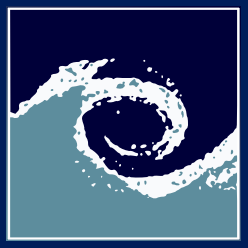
Example on Courant number

```
Co1
{
    type            CourantNo;
    libs            (fieldFunctionObjects);

    // Optional entries
    field           phi; // Flux for the calculation
    result          courantNumber; // Name of output
}
```



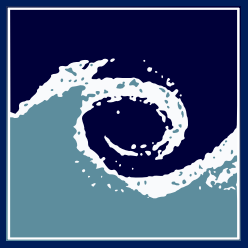
Post-Processing Utilities - Field calculation

- **CourantNo**: Calculates the Courant Number field from the flux field.
- **MachNo**: Calculates the Mach Number field from the velocity field.
- **Q**: Calculates the second invariant of the velocity gradient tensor.
- **turbulenceIntensity**: Calculates and writes the turbulence intensity field I.
- **vorticity**: Calculates the vorticity field, i.e. the curl of the velocity field.
- **yPlus**: Calculates the turbulence y^+, outputting the data as a yPlus field.
- Many more can be listed: postProcess -list



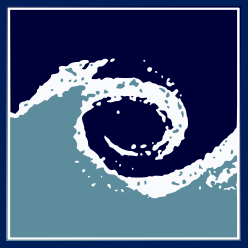
Post-Processing Utilities - Flow rate calculation

- **flowRateFaceZone**: Calculates the flow rate through a specified face zone by summing the flux on patch faces. For solvers where the flux is volumetric, the flow rate is volumetric; where flux is mass flux, the flow rate is mass flow rate.
- **flowRatePatch**: Calculates the flow rate through a specified patch by summing the flux on patch faces. For solvers where the flux is volumetric, the flow rate is volumetric; where flux is mass flux, the flow rate is mass flow rate.
- **volFlowRateSurface**: Calculates volumetric flow rate through a specified triangulated surface (e.g. STL file) by interpolating velocity onto the triangles and integrating over the surface area. Triangles need to be small for an accurate result.



Post-Processing Utilities - Forces and coefficients

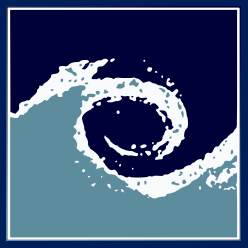
- **forceCoeffsCompressible**: Calculates lift, drag and moment coefficients by summing forces on specified patches for a case where the solver is compressible, the unit of pressure is Pa.
- **forceCoeffsIncompressible**: Similar to the previous, but for incomplessible solvers, the pressure is the kinematic pressure $[m^2/s^2]$.
- **forcesCompressible**: Calculates pressure and viscous forces over specified patches for a case where the solver is compressible, the unit of pressure is Pa.
- **forcesIncompressible**: Similar to the previous, but for incomplessible solvers, the pressure is the kinematic pressure $[m^2/s^2]$.



Post-Processing Utilities - Sampling for graph plotting

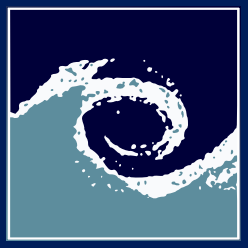
- **singleGraph**: Writes graph data for specified fields along a line, specified by start and end points.

```
functions
{
  type                sets;
  libs                (sampling);
  writeControl        writeTime;
  interpolationScheme cellPoint;
  setFormat           raw;

  setConfig
  {
    type    lineCell;
    axis    distance;  // x, y, z, xyz
  }

  sets
  (
    line
    {
        $setConfig;
        start (0 0 0);
        end   (0 1 0);
    }
  );
  fields  (U);
}
```



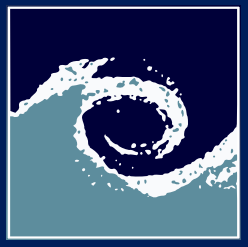
Post-Processing Utilities - Monitoring min/max

- **cellMax/cellMin**: Writes out the maximum/minimum cell value for one or more fields.
- **faceMax/faceMin**: Writes out the maximum/minimum face value for one or more fields.
- **minMaxComponents**: Writes out the minimum and maximum values, by component for non-scalar fields, and the locations where they occur.
- **minMaxMagnitude**: Writes out the minimum and maximum values, by magnitude for non-scalar fields, and the locations where they occur.



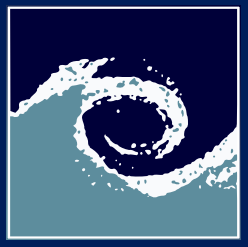
Post-Processing Utilities - Monitoring numerical data

- **residuals**: For specified fields, writes out the initial residuals for the first solution of each time step; for non-scalar fields (e.g. vectors), writes the largest of the residuals for each component.
- **time**: Writes run time, CPU time and clock time and optionally the CPU and clock times per time step.



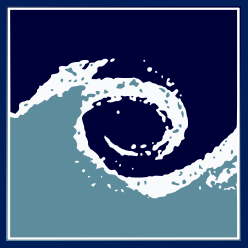
Post-Processing Utilities - Pressure tools

- **pressureDifferencePatch**: Calculates pressures onto 2 specified patch surfaces and calculates the difference between the average pressures.
- **pressureDifferenceSurface**: Interpolates pressures onto 2 specified triangulated surfaces (e.g. from stl files) and calculates the difference between the average pressures.
- **staticPressure**: Calculates the pressure field in Pa from kinematic pressure by scaling by a specified density.
- **totalPressureCompressible**: Calculates the total pressure field for a case where the solver is compressible (pressure is in Pa).
- **totalPressureIncompressible**: Calculates the total pressure field for a case where the solver is incompressible (pressure is kinematic).



Post-Processing Utilities - Probes

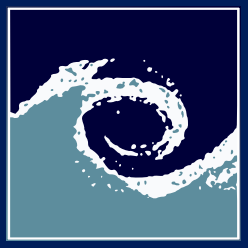
- **boundaryCloud**: Writes out values of fields at a cloud of points, interpolated to specified boundary patches.
- **interfaceHeight**: Reports the height of the interface above a set of locations. For each location, it writes the vertical distance of the interface above both the location and the lowest boundary. It also writes the point on the interface from which these heights are computed.
- **internalCloud**: Writes out values of fields interpolated to a specified cloud of points.
- **probes**: Writes out values of fields from cells nearest to specified locations.



Post-Processing Utilities - Pluggable solvers

- **icoUncoupledKinematicCloud**: Tracks a cloud of parcels driven by the flow of the continuous phase.
- **scalarTransport**: Solves a transport equation for a scalar field.

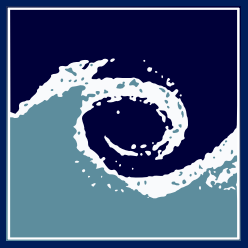
```
functions
{
  s1Transport
  {
    type            scalarTransport;
    libs            (solverFunctionObjects);
    resetOnStartUp  false;
    active          true;
    field           s1;

    // Name of field to use as diffusivity, default = 'none'
    nut             nut;
  }
}
```



Post-Processing Utilities - Visualisation tools

- **streamlines**: Writes out files of streamlines with interpolated field data in VTK format.
- **surfaces**: Writes out surface files with interpolated field data in VTK format, e.g. cutting planes, iso-surfaces and patch boundary surfaces.

```
functions
{
    isoSurfaceQ
    {
        type            surfaces;
        libs            (sampling);
        writeControl    writeTime;
        surfaceFormat   vtk;

        fields          ( U );

        interpolationScheme       cellPoint;
        surfaces
        (
          isoQ100
          {
              type                isoSurface;
              isoField            Q;
              isoValue            100.0;
              interpolate         true;
          }
        );
    }
}
```



Post-Processing Utilities - Video creation

- Save a sequence of field data in (e.g. in ensight format).
- Save a sequence of images from the ensight files with paraview (save animation).
- Create a video from the images in command line with ffmpeg:

```
# Options (not all of them are used in the final command):
# Set frame-rate: -framerate 24
# Set number of image from which the video will runs: -start_number 21
# Set the name of source images (e.g. for image.0001.png): -i image.%04d.png
# Set quality: -c:v libx264 -profile:v high -crf 20 -pix_fmt yuv420p
# Set output name and format: PIP_optimized_long.mp4
ffmpeg -framerate 24 -i <images_name>.%04d.png -c:v libx264 <video_name>.mp4
```