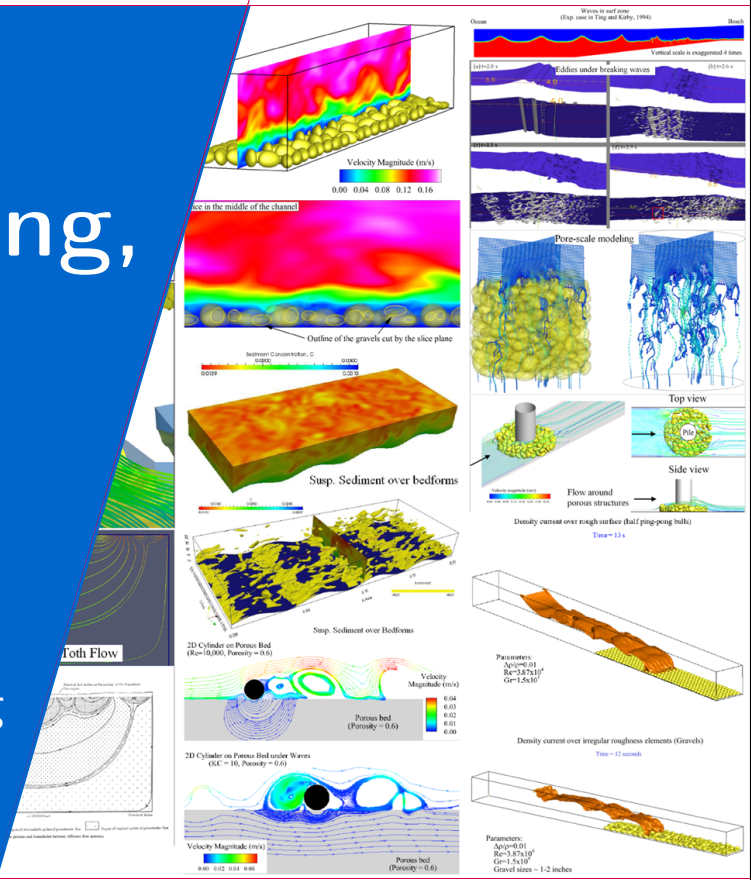
PENNSTATE

1855

CE597D Special Topics, Fall, 2014

Chapter 8: Postprocessing, Data Analysis, and Visualization

Xiaofeng Liu, Ph.D., P.E.
Assistant Professor
Department of Civil and Environmental Engineering
Pennsylvania State University
xliu@engr.psu.edu



What will be covered in this chapter?

- Postprocessing in OpenFOAM®
- Simple data processing and plotting
- Visualization using Paraview and other software

Outline

Postprocessing in OpenFOAM

Simple data processing and plotting

Visualization using Paraview and other software

Postprocessing

Postprocessing is the step during or after the simulation based on the simulation results to

- produce derivative data: e.g., calculate vorticity, Q, λ_2, enstrophy, turbulence quantities (Reynolds stress, fluxes, etc.), max, min, average, integrate on a surface or volume, y^+ on wall patches, wall shear stress
- get subset of data: sample, probe, etc.
- data conversion: to other formats
- ...

Postprocessing can be done:

- when the simulation is totally finished: limited by results only at output steps
- or during the simulation (on the fly): fine grain control on the timing of postprocssing operations (every time step, output time, etc.). In OpenFOAM® , it can be realized through `functionObject` specified in `controlDict` of a simulation case.

Postprocessing in OpenFOAM

We have used many postprocessing tools in OpenFOAM® . In this section, we will demonstrate more.

- probeLocations and sample
- Related to flow field: vorticity, Lambda2, Q

If the functionality you need is not available, you can easily program a new tool!

Postprocessing in OpenFOAM

`probeLocations`:

- find the value of specified solution variables at specified locations.
- typically result in file for each probed field in `postprocessing/probes`.
- the probed data can be plotted or compared against experimental data.
- Note: the values are at cell center. They are not interpolated.
- Two modes of use:
  - 1. run `probeLocations` on output time steps. Specification in `system/probesDict` file

```
// Fields to be probed.
fields
(
    p
    U
);

// Locations to be probed.
probeLocations
(
    (0.05 0.05 0.005)
    (0.025 0.025 0.005)
);
```

Postprocessing in OpenFOAM

Output in the `postProcessing/probes` directory:

pressure p field

```
#           x          0.05         0.025
#           y          0.05         0.025
#           z         0.005         0.005
#        Time
            0             0             0
         0.05    0.00495652   -0.00754243
          0.1  -0.000810915   -0.00842594
         0.15   -0.00128265   -0.00849437
          0.2   -0.00132184   -0.00849626
         0.25   -0.00132609   -0.00849464
          0.3   -0.00132823   -0.00849446
...
```

velocity U field

```
#           x          0.05         0.025
#           y          0.05         0.025
#           z         0.005         0.005
#        Time
            0       (0 0 0)       (0 0 0)
         0.05  (-0.188039 0.023509 0)    (-0.0741506 0.052183 0)
          0.1  (-0.195694 0.0261668 0)   (-0.0821798 0.0589721 0)
         0.15  (-0.196041 0.0263716 0)   (-0.0830591 0.0597943 0)
          0.2  (-0.196062 0.0263863 0)   (-0.0831344 0.0598661 0)
         0.25  (-0.196065 0.0263878 0)   (-0.0831453 0.0598736 0)
          0.3  (-0.196066 0.0263887 0)   (-0.0831509 0.0598784 0)
...
```

- The first column is the time: note it is the same time of the simulation result output.
- The next columns correspond to different probe locations.

Postprocessing in OpenFOAM

`probeLocations`:

- Two modes of use:
  - 2. run `probeLocations` (`probes` as its name to be more precise) using `functionObjects`. Specified in `controlDict`:

```
functions
{
    probes
    {
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );

        type            probes;

        // Name of the directory for probe data
        name            probes;

        // Write at same frequency as fields
        outputControl   outputTime;
        outputInterval  1;

        // Fields to be probed
        fields
        (
            p U
        );

        probeLocations
        (
            (0.05 0.05 0.005)
            (0.025 0.025 0.005)
        );
    }
}
```

- There is a keyword `outputControl` with the choice of `outputTime` or `timeStep`.
- You can also optionally specify `timeStart` and `timeEnd`.
- I found this mode does not report the probed value at start.

Postprocessing in OpenFOAM

`sample`:

- similar to `probeLocations`, but with more functionalities and controls:
  - sample on `set`: essentially probes
  - sample on `surface`
- specification in file `sampleDict`
- `interpolationScheme`: determines how to interpolate the field value to the sampling position
  - `cell`: use cell-centre value only; constant over cells (default)
  - `cellPoint`: use cell-centre and vertex values
  - `cellPointFace`: use cell-centre, vertex and face values.
  - `pointMVC`: use point values only (Mean Value Coordinates)
  - `cellPatchConstrained`: like `cell` but uses cell-centre except on boundary faces where it uses the boundary value.
- output format:
  - `setFormat` for `set`: `xmgr`, `jplot`, `gnuplot`, `raw`, `vtk`, `ensight`, and `csv`.
  - `surfaceFormat` for `surface`: `ensight`, `foamFile`, `dx`, `vtk`, and `raw`.

Postprocessing in OpenFOAM

`sample`:

- for `sets` (probing at points), we can define `uniform` (points on a line), `face` (one point per face), `cloud` (arbitrary points), `triSurfaceMeshPointSet` (points of the surface mesh), etc.
- for `surfaces`, we can define `plane`, `patch`, `patchInternalField`, `isoSurface`, `isoSurfaceCell`, `cuttingPlane`, `distanceSurface`, `sampledTriSurfaceMesh`
- To define the fields to be sampled:

```
fields
(
    p
    U
);
```

- the results will be in `postProcessing/sets`. Each time step will result in an individual subdirectory.

Postprocessing in OpenFOAM

`sample`:

- run the tool on the output results is relatively simple.

Postprocessing in OpenFOAM

Other tools:

- mapField
- foamToVTK
- foamLog
- foamCalc

Postprocessing in OpenFOAM

How to visualize flow field? Active research field.

- Velocity vector
- Streamlines (instantaneous tangent lines of **u**), streaklines (dye test), pathlines (particle trajectory)
- Vorticity, λ_2, Q:
  - Vorticity: $\omega = \nabla \times \mathbf{u}$
  - λ_2: the second largest eigenvalue of the sum of the square of the symmetrical and anti-symmetrical parts of the velocity gradient tensor

$$S_{ik}S_{kj} + \Omega_{ik}\Omega_{kj} \tag{1}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2}$$

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \tag{3}$$

$$\nabla \mathbf{u} = \frac{\partial u_i}{\partial x_j} = (S_{ij} + \Omega_{ij}) \tag{4}$$

Reference: J. Jeong and F. Hussain (1995). On the identification of a vortex. *Journal of Fluid Mechanics*, 285, pp 69-94

Postprocessing in OpenFOAM

How to visualize flow field? Active research field.

- Q: connected region where $Q > 0$ and the pressure is lower than the ambient identifies a vortex core.

$$Q = \frac{1}{2}\left(\Omega_{ij}\Omega_{ij} - S_{ij}S_{ij}\right) \tag{5}$$

  Reference: Hunt,J.C.R., Wray, A.A. and Moin ,P. (1988), Report CTR-S88

- All the above have been implemented as utility tools in OpenFOAM® in `applications/utilities/postProcessing/velocityField`
- Further readings on this topic:
  Y. Dubief and F. Delcayre (2000). On coherent-vortex identification in turbulence. Journal of Turbulence, 1, N11.

Postprocessing in OpenFOAM

An example figure in Dubief and F. Delcayre (2000)

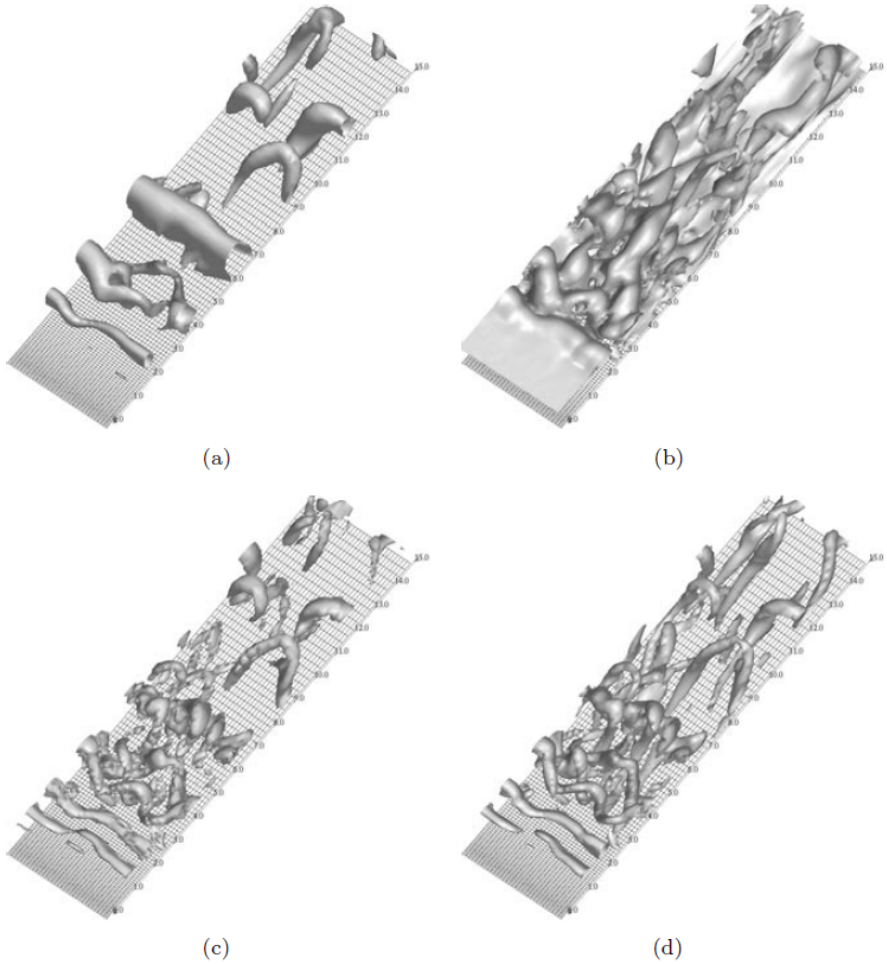


Figure 4. Vortical activity in a backward-facing step flow educed by various criteria. (a) Pressure isosurfaces, $P = -0.003\rho_0 U_0^2$. (b)Vorticity isosurfaces, $\omega = 1.5U_0/h$. (c) λ_2 isosurfaces, $\lambda 2 = -0.075$. (d) Q isosurfaces, $Q = 0.25$.

Another example figure in Dubief and F. Delcayre (2000)

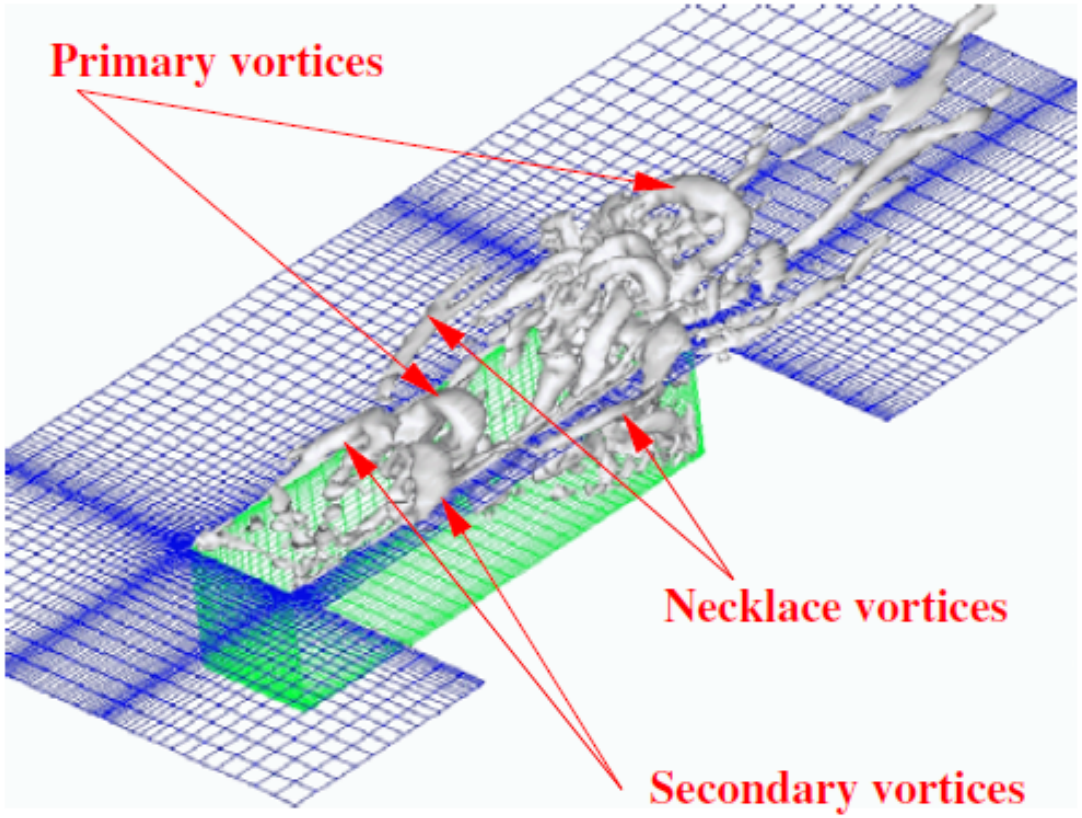


Figure: Coherent vortical structure identification using Q criterion

More on functionObjects

- `functionObjects` is a mechanism to attach a function to a solver at run-time.
- The solver does not need to know the existence the `functionObjects`. As such, you don't need to recompile the solver code to have additional functionalities.
- The coordination of the calling to the `functionObjects` is through the class `Time` (`src/OpenFOAM/db/Time`)
  - `Time` class keeps a list of all function objects

    ```
    mutable functionObjectList functionObjects_;
    ```
  - During the time loop, each of the the function object in the list is executed based on their specific timing (`outputTime` or `timeStep`)
- To add function objects to the simulation, a `functions` entry needs to be added to `system/controlDict`
- Most of the function objects are implemented in

  `src/postProcessing/functionObjects`
- Most of them run in parallel too!

More on functionObjects

Available `functionObjects`:

- http://www.openfoam.org/features/runtime-postprocessing.php
- `fieldAverage`, `fieldMinMax`, `forces`, `forceCoeffs`, `sampledSet`, `probes`, `isoSurface`, `cuttingPlane`, `streamline`, etc.

Didn't find what you need? Write one by yourself.

- Find the closest one and make a copy into your own directory
- Modify the code
- Modify `Make/files` and `Make/options`, and re-compile

More on functionObjects

`fieldAverage`:

- It calculates the time average and variance of specified fields.
- The definitions are very simple:

$$\text{mean: } \overline{\phi_N} = \frac{1}{N-1}\sum_{i=1}^{N}\phi_i, \tag{6}$$

$$\text{variance: } \sigma_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(\phi_i - \overline{\phi_N}\right)^2. \tag{7}$$

- However, there is one problem!
- In order to calculate these statistical quantities, theoretically we need to store the field values at all N time steps.
- This is not possible most of time.
- Usually we use other equivalent algorithms, also named `online` algorithms

More on functionObjects

`fieldAverage`:

- OpenFOAM® implements the following:

$$\overline{\phi}_N = \frac{Dt - dt}{Dt}\overline{\phi}_{N-1} + \frac{dt}{Dt}\phi_N, \tag{8}$$

  where dt is the current time step size, Dt is the total time since the starting of mean calculation.

- The new mean is improved by the weighted-average of the old mean and the new solution.
- If the time step is constant, then Equation 8 can be changed to

$$\overline{\phi}_N = \frac{N-1}{N}\overline{\phi}_{N-1} + \frac{1}{N}\phi_N, \tag{9}$$

  which is more frequently used in statistics.

More on functionObjects

`fieldAverage`:

- the variance is calculated in OpenFOAM® using:

$$
\begin{aligned}
\sigma_N^2 &= \frac{1}{N-1}\sum_{i=1}^{N}\left(\phi_i - \overline{\phi_N}\right)^2 \\
&= \frac{1}{N-1}\left[\sum_{i=1}^{N}\phi_i^2 - N\left(\overline{\phi_N}\right)^2\right] \\
&= \frac{1}{N-1}\left[\sum_{i=1}^{N-1}\phi_i^2 - (N-1)\left(\overline{\phi_N}\right)^2 + \phi_N^2 - \overline{\phi_N}^2\right] \\
&= \frac{1}{N-1}\left[\sigma_{N-1}^2(N-2) + \phi_N^2 - \overline{\phi_N}^2\right] \\
&= \frac{N-2}{N-1}\sigma_{N-1}^2 + \frac{1}{N-1}\left[\phi_N^2 - \overline{\phi_N}^2\right] \\
&= \frac{N-2}{N-1}\left(\sigma_{N-1}^2 + \overline{\phi_N}^2\right) + \frac{1}{N-1}\left[\phi_N^2 - \overline{\phi_N}^2\right] - \frac{N-2}{N-1}\overline{\phi_N}^2 \\
&= \frac{N-2}{N-1}\left(\sigma_{N-1}^2 + \overline{\phi_N}^2\right) + \frac{1}{N-1}\phi_N^2 - \overline{\phi_N}^2
\end{aligned}
$$

PENNSTATE

More on functionObjects

`fieldAverage`:

- this formula for variance calculation in OpenFOAM® uses some well known formulas in statistics.
- Also some special treatment has been done to avoid the loss of accuracy during the calculation.
- Otherwise the loss of accuracy might cause sever problem during the numerical computation, especially when the values of ϕ_i is large and the difference between them is small.

More on functionObjects

An example of using `fieldAverage` in the `tutorials/incompressible/pimpleFoam/channel395`. It is specified in the `system/controlDict` file:

```
functions
{
    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ( "libfieldFunctionObjects.so" );
        enabled         true;
        outputControl   outputTime;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }
        );
    }
}
```

More on functionObjects

`fieldAverage`:

- There are some other setups of interest:
  - the resumption of averaging upon restart and output through switches `resetOnRestart` and `resetOnOutput`
  - The default value for both is `false`.
  - The resumption of averaging on restart is through the `fieldAveragingProperties` dictionary file in the `uniform` directory of output time.
  - In this dictionary file, the total time and total iterations for each averaging field are recorded which can be used to resume the averaging.

Simple data processing and plotting

This section will demonstrate some tools to do data analysis and plotting

- Gnuplot
- Python and Matplotlib
- Octave (or Matlab)

Gnuplot

- Gnuplot is an open source, cross-platform plotting tool.
- http://www.gnuplot.info/
- It can generate 1D, 2D, and 3D plots.
- Support both interactive and scripting
- Support different output formats: pdf, png, jpeg, Latex, svg, etc.
- To use Latex for annotation is possible, but needs some extra steps.
- Scripting is good to keep all the generated figures have consistent style (font, size, color, etc.)

Demo of Gnuplot plot on sampled velocity result.

Python and Matplotlib

- Python and Matplotlib
- https://www.python.org/
- http://matplotlib.org/
- Python is more powerful than just plotting.

Demo of Python and Matplotlib.

A short tutorial of Paraview

- We will use Paraview v4.2: www.paraview.org
- Paraview is an open-source, multi-platform parallel data analysis and visualization software
- Built up the VTK (Visualization Toolkit) library
- It supports a wide variety of data formats
  - Structure grid
  - Unstructured grid
  - Images
  - Multi-block
  - AMR
  - Time series are automatically supported

A short tutorial of Paraview

- A rich selection of visualization tools
  - Contour
  - Isosurface
  - Cutting plane
  - Vectors (Glyphs)
  - Streamlines
  - Volume rendering
  - Clipping
  - ...
- Derived variables: Calculator and other filters
- Support of Python for scripting
- Support of export to different formats
- Support for parallel run

A short tutorial of Paraview

- VTK format (http://www.vtk.org/VTK/img/file-formats.pdf)
  - structured points
  - structured grid
  - rectlinear grid
  - polygonal data
  - unstructured grid
  - ...
- VTK can store scalar, vector, tensor, etc.
- VTK has several formats: Legacy, and more recently XML
- Python has good support for VTK data format. A good selection of libraries.
- Paraview can be easily extended: for example, read OpenFOAM® results

A short tutorial of Paraview

The concept of pipeline:

- All processing operations on data set will produce new data set
- The new data set can be further processed (pipeline)
- Example:
  - Operation 1: Extract a free surface (contour of $\alpha = 0.5$ from `interFoam` simulation)
  - Operation 2: Plot velocity vector on the free surface (glyphs)
  - ...
- With the combination of different operations, the result can be very complicated.
- For a list of all the operations (filters) available, have a look at the content inside menu `filters`

A short tutorial of Paraview

Scripting using Python:

- The purpose: automate the process
- Demostration on cluster

A short tutorial of Paraview

More tutorials can be found at:

- http://www.paraview.org/Wiki/The_ParaView_Tutorial

Questions?