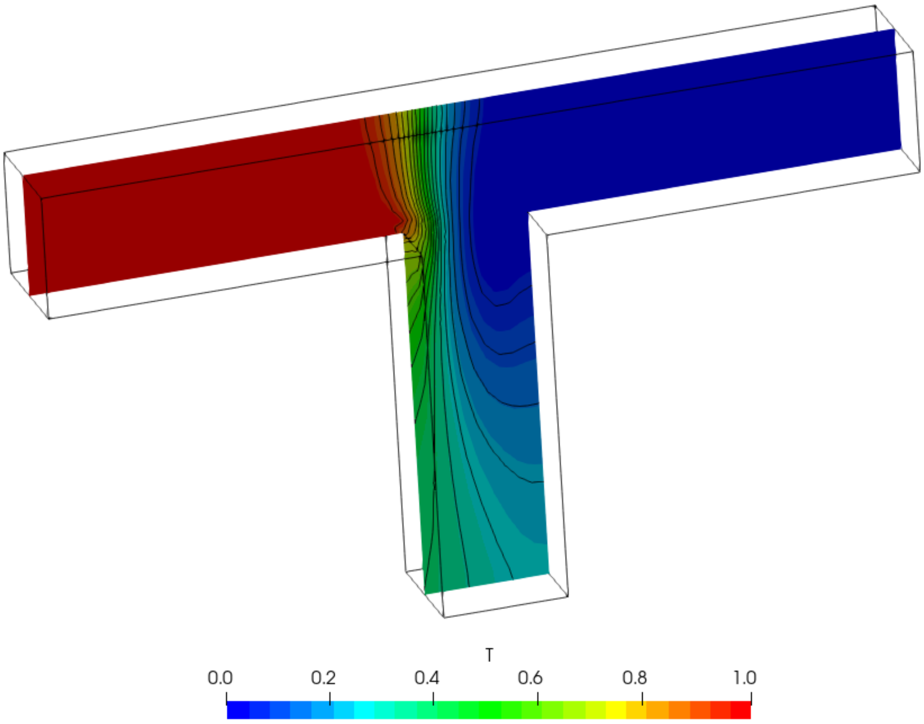
Tutorial Ten

Residence Time Distribution



Bahram Haddadi

ENGINEERING FLUID DYNAMICS

7th edition, March 2025

Contributors:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek
- Clemens Gößnitzer
- Sylvia Zibuschka
- Yitong Chen

Technische Universität Wien
Institute of Chemical, Environmental & Bioscience Engineering

Available from: www.fluiddynamics.at

Background

In this tutorial, we will carry out Residence Time Distribution (RTD) analysis of fluid flow through a T-junction pipe.

1. Residence Time Distribution (RTD)

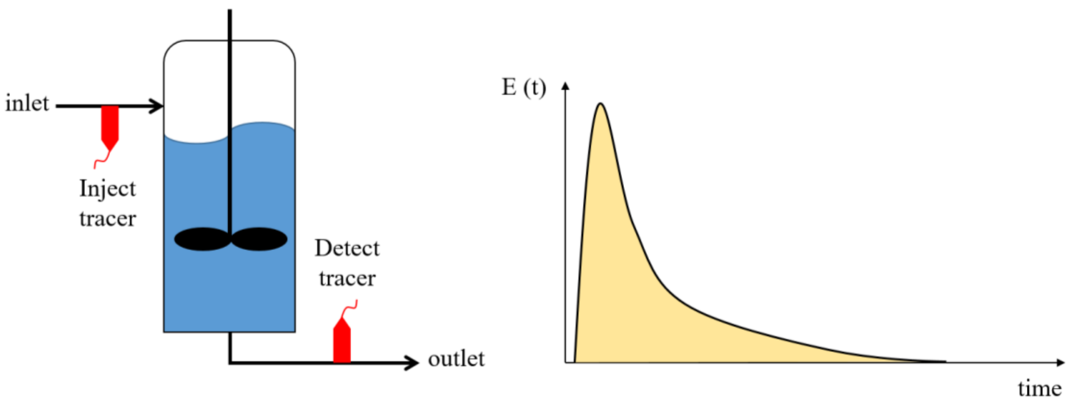
Residence Time Distribution (RTD) is a probability distribution function that describes the amount of time a fluid element spends in a process unit, such as a reactor, column, or pipe. Understanding RTD is crucial for analyzing the performance, efficiency, and mixing characteristics of industrial systems. Unlike ideal flow assumptions, most real-world fluid flows involve recirculation, bypassing, and dispersion, which RTD helps quantify. A few RTD applications:

- Optimizing reactor design by ensuring effective residence time for reactions.
- Identifying flow inefficiencies such as dead zones and short-circuiting.
- Enhancing mixing performance in industrial processes.
- Improving scale-up accuracy for chemical, pharmaceutical, and wastewater applications.

By understanding RTD, engineers can optimize designs, improve product yield, and enhance process reliability.

2. Tracer Analysis

Tracer analysis is a widely used technique for RTD measurement, where a tracer substance (such as dye, salt, or radioactive material) is injected into the system, and its concentration at the outlet is monitored over time. This provides insight into how fluid elements move through the process unit and allows engineers to quantify the RTD function.



Tracer analysis and RTD distribution of an ideal process

Based on the above diagram, first the tracer is injected into the inlet, and then the exit tracer concentration, $C(t)$, is measured at regular time intervals. This allows the exit age distribution, $E(t)$, to be calculated.

$$E(t) = \frac{C_T(t)}{\int_0^\infty C_T(t)\,dt} = \frac{Tracer\ concentration\ at\ time\ t}{Total\ tracer\ concentration}$$

It is clear from the above equation that the fraction of tracer molecules exiting the reactor that have spent a time between t and $t + dt$ in the process unit is $E(t)dt$. Since all tracer elements will leave the unit at some point, RTD satisfies the following relationship:

$$\int_0^\infty E(t)\,dt = 1$$

Types of Flow Patterns Identified by RTD

- Ideal Plug Flow: All fluid particles have the same residence time, resulting in a sharp, narrow RTD peak.
- Perfectly Mixed Flow (CSTR - Continuous Stirred Tank Reactor): Fluid particles experience a wide range of residence times, leading to a broad RTD distribution.
- Dead Zones and Recirculation: Cause multiple peaks in the RTD curve, indicating poor mixing and stagnation.
- Bypassing Flow: Results in a steep initial RTD rise, meaning some fluid exits much earlier than expected.

incompressibleFluid & functions – TJunction

Tutorial outline

Use the incompressibleFluid and functions to simulate the flow through a square cross section T pipe and calculate RTD (Residence Time Distribution) for both inlets using a step function injection:

- Inlet and outlet cross-sections: 1 × 1 m^2
- Gas in the system: air at ambient conditions
- Operating pressure: 10^5 Pa
- Inlet 1: 0.1 m/s
- Inlet 2: 0.2 m/s

Objectives

- Understanding RTD calculation using OpenFOAM®
- Using multiple solvers for a simulation

Data processing

Plot the step response function and the RTD curve.

1. Pre-processing

1.1. Copying tutorial

Copy the following tutorial to your working directory as a base case:

$FOAM_TUTORIALS/incompressibleFluid/pitzDaily

Replace the system directory with the system directory from the following tutorial:

$FOAM_TUTORIALS/incompressibleFluid/pitzDailySteadyExperimentalInlet

Copy the *pitzDaily* file for the pitzDaily geometry from following directory to your system directory:

$FOAM_TUTORIALS/resources/blockMesh

1.2. 0 directory

Update p, U, nut, nuTilda, k and epsilon files with the new boundary conditions (in this simulation the following boundaries should be set inlet_one, inlet_two, oulet and walls), e.g. for file *U*:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet_one
    {
        type            fixedValue;
        value           uniform (0.1 0 0)
    }
    inlet_two
    {
        type            fixedValue;
        value           uniform (-0.2 0 0)
    }
    outlet
    {
        type            zeroGradient;
    }
    walls
    {
        type            fixedValue;
        value           uniform (0 0 0)

    }
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
```

1.3. constant directory

Check momentumTransport file for the turbulence model (`kEpsilon`).

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
simulationType  RAS
RAS
{
    model        kEpsilon;

    turbulence      on;

    printCoeffs     on;

}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
```

1.4. system directory

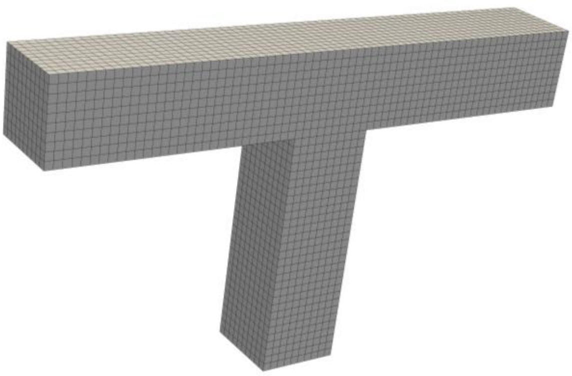
Rename the pitzDaily file to blockMeshDict and edit it to create the geometry.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
convertToMeters 1.0;

vertices
(
    (0 4 0) // 0
    (0 3 0) // 1
    (3 3 0) // 2
    (3 0 0) // 3
    (4 0 0) // 4
    (4 3 0) // 5
    (7 3 0) // 6
    (7 4 0) // 7
    (4 4 0) // 8
    (3 4 0) // 9
    (0 4 1) // 10
    (0 3 1) // 11
    (3 3 1) // 12
    (3 0 1) // 13
    (4 0 1) // 14
    (4 3 1) // 15
    (7 3 1) // 16
    (7 4 1) // 17
    (4 4 1) // 18
    (3 4 1) // 19

);
blocks
(
    hex (0 1 2 9 10 11 12 19) (10 30 10) simpleGrading (1 1 1)
    hex (9 2 5 8 19 12 15 18) (10 10 10) simpleGrading (1 1 1)
    hex (8 5 6 7 18 15 16 17) (10 30 10) simpleGrading (1 1 1)
    hex (2 3 4 5 12 13 14 15) (30 10 10) simpleGrading (1 1 1)
);
boundary
(
    inlet_one
    {
         type patch;
         faces
         (
            (0 10 11 1)
         );
    }
    inlet_two
    {
         type patch;
         faces
         (
            (7 6 16 17)
```

```
          );
     }
     outlet
     {
          type patch;
          faces
          (
             (4 3 13 14)
          );
     }
     walls
     {
          type wall;
          faces
          (
             (0 1 2 9)
             (2 5 8 9)
             (5 6 7 8)
             (2 3 4 5)
             (10 19 12 11)
             (19 18 15 12)
             (18 17 16 15)
             (15 14 13 12)
             (0 9 19 10)
             (9 8 18 19)
             (8 7 17 18)
             (2 1 11 12)
             (3 2 12 13)
             (5 4 14 15)
             (6 5 15 16)
          );
     }
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
```

2. Running simulation

```
>blockMesh
```



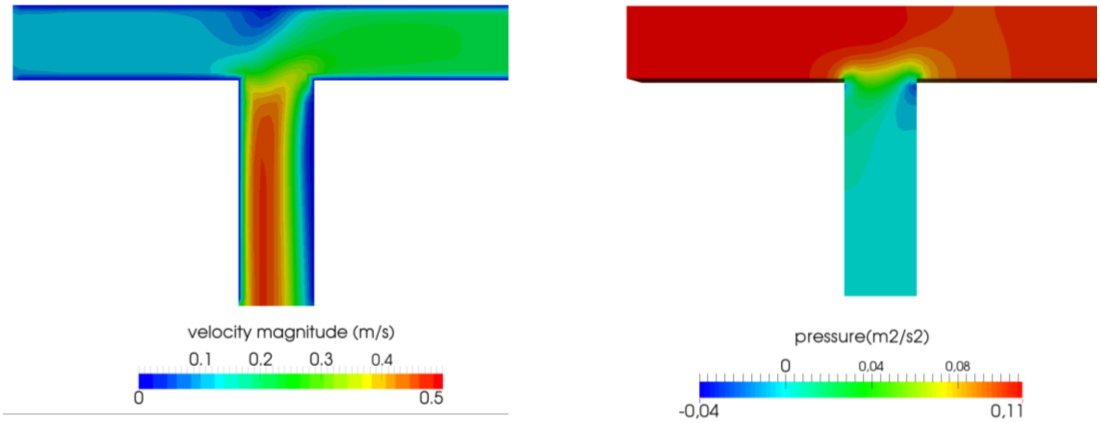
Mesh created using blockMesh

```
>foamRun -solver incompressibleFluid
```

Wait for simulation to converge. After convergence, check the results to make sure about physical convergence of the solution.

```
>foamToVTK
```

The simulation results are as follows (results are on the cut plane in the middle):



Simulation results after convergence (~65 iterations)

3. RTD calculation

3.1. Copy tutorial

Copy following tutorial to your working directory:

`$FOAM_TUTORIALS/incompressibleFluid/pitzDailyScalarTransport`

In the 0 directory, just keep the *T* file and delete all other files.

3.2. 0 directory

Copy and paste the *U* and *p* files from the latest time step of the simulation in the first part of the tutorial (use the latest time step velocity field from previous part of simulation to calculate RTD for this geometry). There is no need to modify or change it. The solver will use this field to calculate the scalar transportation.

Update *T* (T will be used as an inert scalar in this simulation) file boundary conditions to match new simulation boundaries, to calculate RTD of the `inlet_one` set the `internalField` value to `0`, T value for `inlet_one` to `1.0` and T value for `inlet_two` to `0`.

3.3. constant directory

In the *momentumTransport* file set the `simulationType` to `laminar`.

3.4. system directory

Copy the blockMeshDict file from the first part of tutorial.

In the controlDict file change the `endTime` from `0.2` to `120` (approximately two times ideal resistance time) and `deltaT` from `0.0001` to `0.1` (Courant number approximately 0.4).

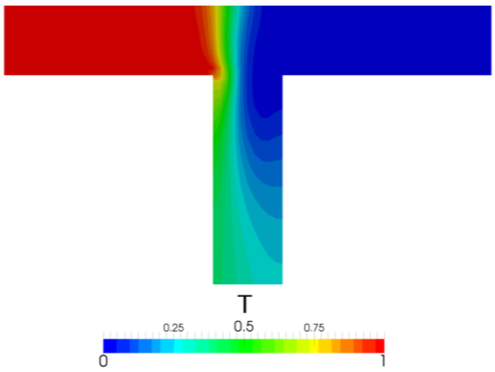
4. Running Simulation

```
>blockMesh
```

```
>foamRun -solver functions
```

```
>foamToVTK
```

5. Post-processing



Contour plots scalar T at 120 s for inlet 1

5.1. Calculating RTD

To calculate RTD the average T value at the outlets should be calculated first. The “integrate variables function” of ParaView can be used for this purpose.

```
>foamToVTK
```

Load the outlet VTK file into paraview using following path:

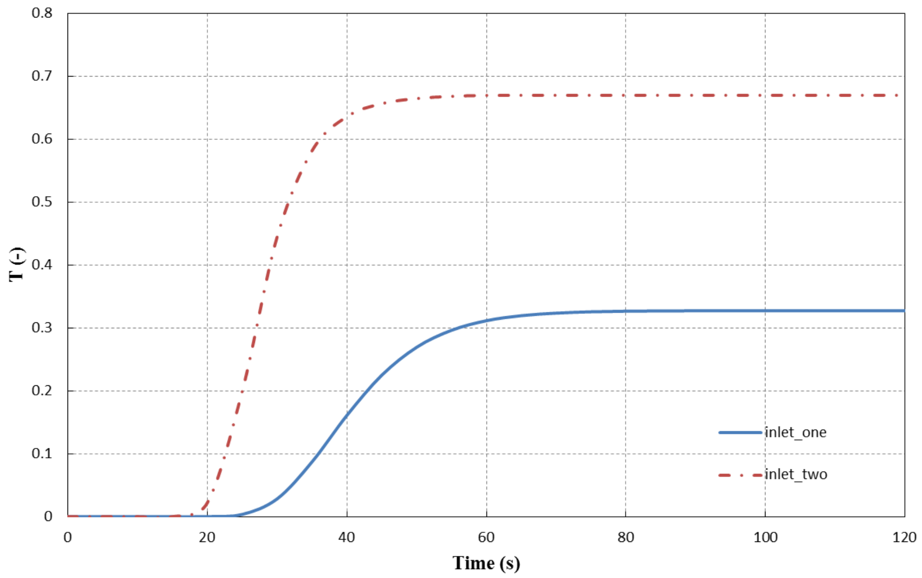
File > Open > VTK > outlet > outlet_..vtk > OK > Apply

Select T from variables menu, and then integrate the variables on the outlet:

Filters > Data Analysis > Integrate Variables > Apply

The values given in the opened window are integrated values in this specific time step. By changing the time step values for different time steps are displayed. As mentioned before, the average value of the property is needed. Therefore, these values should be divided by outlet area to get average values (1m × 1m).

After finishing the RTD calculations for `inlet_one`, the same procedure should be followed for calculating RTD of `inlet_two`, except T value for `inlet_one` should be `0` and for `inlet_two` it should be `1.0`.

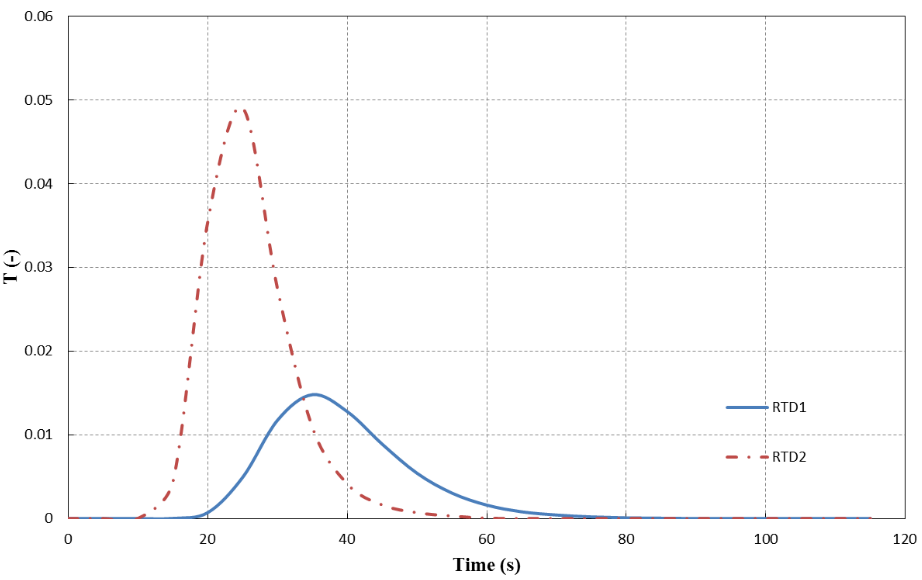


Average value of T on the outlet for two inlets versus time

The average value of T for each outlet approaches a certain constant value, which is the ratio of that scalar mass inlet to the whole mass inlet. For plotting data over time "Plot Selection Over Time" option in ParaView can be used, in the opened SpreadSheetView window (IntegrateVariables) select the set of data which you want to plot over time and then:

Filters > Data Analysis > Plot Selection Over Time > Apply

Next, to obtain the RTD plots, export the data to a spreadsheet program (e.g. Excel), calculate and plot the gradient of changes in average value of T on the outlet from time 0 to 120s for both inlets.



RTD of two inlets