Final Project

Group #4

Banik, Sagnik

Kasabe Bhupendra,Shreyas

1. Case description / Motivation

Generally, the transition from a laminar to a turbulent flow originates from instabilities in the laminar base flow. In particular, shear flows, i.e., flows with a non-vanishing velocity gradient, are prone to transition mechanisms and can be unstable. Likewise, this behaviour can be seen in unstable layers of resting fluid, e.g., a liquid with higher density resting above a liquid with smaller density under the effect of gravity.

The Rayleigh-Taylor instability is an instability of the interface separating two immiscible fluids: the densest of which rests on the lightest. Following a disturbance of the interface, the dense fluid enters the light fluid. The interface growths with a characteristic shape depending upon density ratio. In this report we did an extensive study on Rayleigh Taylor instability, it evolution characteristic with different density ratio along with stability analysis.

Rayleigh-Taylor Instability: -

We examine the stability of a horizontal interface between two fluids (at rest) with different density $\rho_2 \neq \rho_1$, as schematically shown in Figure 1.It is a 2D, incompressible, inviscid flow.

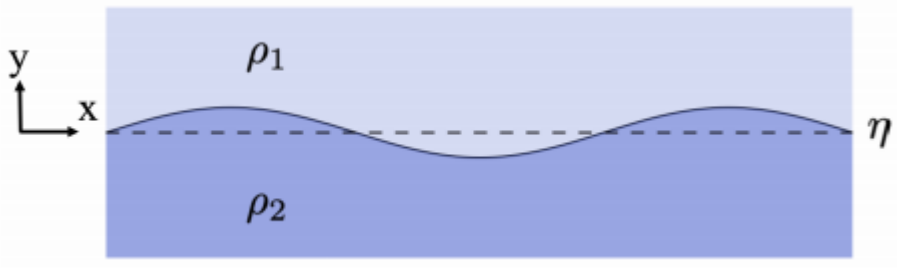


Fig.1 Rayleigh-Taylor instability: initial interface (dashed-line)

RT instability is a dynamic process whereby the two fluids tend to reduce their combined potential energy. The growth rate of the initial perturbation introduced in the interface η can be assessed from the dispersion relation given by:

$$\omega^2 = g\left(\frac{\rho 2 - \rho 1}{\rho 2 + \rho 1}\right)\kappa$$

Where κ is the wave number and ω is the frequency of the perturbation. Clearly for complex frequency mode $\omega = \pm i\sigma$, it would result in instability growth. We denote the Atwood number as

$$At = \frac{\rho_2 - \rho_1}{\rho_2 + \rho_1}$$

As it can be clearly deduced that for $\rho_2 < \rho_1$ will result in a unstable system and for $\rho_2 > \rho 1$ will result in a stable system.

Phenomenology:

Perturbation of the initial interface between two fluid layers of ρ2 < ρ1 (dashed line) leads to plume formation. Perturbation wave number (κ = 2π/λ) , wavelength of the instability (λ).

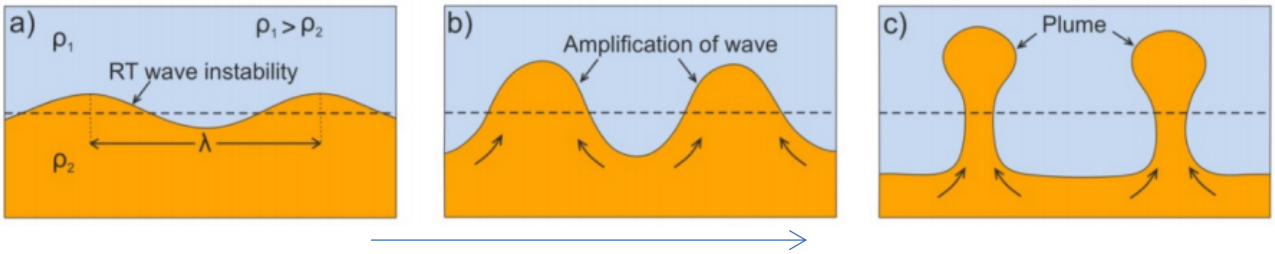


Fig.2 Evolution of RT instability

- Stage 1: linear stability analysis: η grow exponentially
- Stage 2: 3-dimensionality & nonlinearity
- Stage 3: development of mature plumes due to further nonlinear interaction of the unstable modes with other nonlinear modes.
  At=0: RT instability takes the forms of symmetric finger
  At=1: RT instability takes the forms of larger bubble-like plumes

For our case we have introduced a cosine wave perturbation at the interface given by

$$\eta = 0.1cos\left(\frac{2\pi x}{w}x\right)$$

of amplitude 0.1 and was the width of the channel. We have followed the study of Lee et al[2], which deduced that for this perturbation the concentration at the interface can be given as :

$$\alpha = tanh\left(\frac{y-\eta-0.1cos\left(\frac{2\pi x}{w}\right)}{\sqrt{2}\epsilon}\right)$$

We have used this results in our simulation with modification according to our defined domain. The parameter ϵ is taken as a measure of interface width.

2. Solution approach

a) Code

We have restricted our study within the domain of width 1 unit and height of 4 unit. Accordingly, the perturbation can be stated as:

$$\eta = 0.1cos(2\,\pi\,x)$$

$$\alpha = tanh\left(\frac{y - 2 - 0.1cos(2\,\pi\,x)}{\sqrt{2}\epsilon}\right)$$

where the interface is taken in the middle and $\epsilon = 0.01$.

We have worked out five cases in following manner: -

- **Case I-** Two liquids: Density ratio of 3:1, At = 0.5.
- **Case II-** Water and oil: Density ratio of 1.05:1, At= 0.025.
- **Case III-** Two fluids with lighter fluid on top: Density ratio of 1:3.
- **Case IV-** 3D RT instability with two fluids: Density ratio of 3:1, At=0.5.
- **Case V:** Evolution of RT for very long time to show how two liquids exchange their places

All the cases are performed under laminar case.The initial perturbation in each of the above mentioned cases can be initialized using following methods.

b) Interface initialization

Setting up the case requires the initialization of the interface between the heavy fluid and the light fluid with a cosine-like shape. We have set the case the using following three methods

- Use **funkySetFields**: This utility sets the value of a scalar or a vector field depending on an expression that can be entered via the command line or a dictionary. It can also be used to set the value of fields on selected patches.

- Use **setFields**: **setFields** is an OpenFOAM® utility for initializing a given field, including the possibility to use a stl format surface.
  .
- Use **codeStream**: **codeStream** offers the possibility of coding directly in the dictionaries of boundary conditions its instructions.

Through funkySetFields and setFields we have intialised through the interface cosine perturbation as a interface displacement. Through codeStream, a direct initialisation is provided for the α (interface concentration) field through the mentioned equation.

The code files of the initialization methods are as follows:-

1. **funkySetFieldsDict**:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      funkySetFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
expressions
(
        setfluid
        {
                field     alpha;
                condition "pos().y >  -0.1*cos(2*pi*pos().x)";
// Intial perturbation given as   cosine wave
                expression "1";
// Setting alpha =1 for all region above the intial perturbed curve
                keepPatches true;
        }
);
```

2. **setFieldsDict:**

```
defaultFieldValues
(
        volScalarFieldValue alpha 0
);

regions
(


    surfaceToCell
    {
        file                                                    "initialsurface.stl";
\\ the initialsurface.stl file is used as a boundary to introduce the cosine perturbation

        useSurfaceOrientation                   false;
        outsidePoints                           ((0 -0.1 0));
        includeCut                              false;
        includeInside                           true;
        includeOutside                          false;
        nearDistance                            -1;
        curvature                               0.9;
        fieldValues
        (
                volScalarFieldValue alpha 1
\\ setting alpha =1  for all region inside the .stl file
        );
    }
);
```

3. codeStream

```
internalField   #codeStream
{
    codeInclude
    #{
        #include "fvCFD.H"
    #};

    codeOptions
    #{
        -I$(LIB_SRC)/finiteVolume/lnInclude \
        -I$(LIB_SRC)/meshTools/lnInclude
    #};

    codeLibs
    #{
        -lmeshTools \
        -lfiniteVolume
    #};

    code
    #{
        const IOdictionary& d = static_cast<const        IOdictionary&>(dict);
        const fvMesh& mesh = refCast<const fvMesh>(d.db());

        scalarField alpha(mesh.nCells(), 0.);

        forAll(alpha, i)
        {
        const scalar x = mesh.C()[i][0];
        const scalar y = mesh.C()[i][1];
\\ defining the positions of x and y

        alpha[i] = 0.5 + 0.5*tanh((y -2 -0.1*cos(2*constant::mathematical::pi*x))/(0.01*1.414));
\\ giving the initial perturbation at the interface in terms of alpha.The equation is modified
accordingly to the domain and keeping alpha in [0,1]

        }

        alpha.writeEntry("", os);

    #};
};
```

c) Case setup

Meshing

The geometric model and its mesh were created through the *blockMesh* dictionary as shown below.

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       dictionary;
   object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
   (0 0 -0.1)
   (1 0 -0.1)
   (1 4 -0.1)
   (0 4 -0.1)

   (0 0 0.1)
   (1 0 0.1)
   (1 4 0.1)
   (0 4 0.1)
);

blocks
(
   hex (0 1 2 3 4 5 6 7) (200 500 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
   topwall
   {
      type wall;
      faces
      (
         (3 7 6 2)
      );
   }
```

```
lowerWall
    {
        type wall;
        faces
        (
            (1 5 4 0)
        );
    }
    leftWall
    {
        type symmetryPlane;
        faces
        (
            (0 4 7 3)
        );
    }
    rightWall
    {
        type symmetryPlane;
        faces
        (
            (2 6 5 1)
        );
    }
    frontAndBack
    {
        type empty;      // we have considered a 2D simulation so the z direction is not taken
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);

mergePatchPairs
(
);
```

The Images of the Geometry generated with blockMesh 2D cases, 3D case and .stl file used for setFieldsDict are shown below:

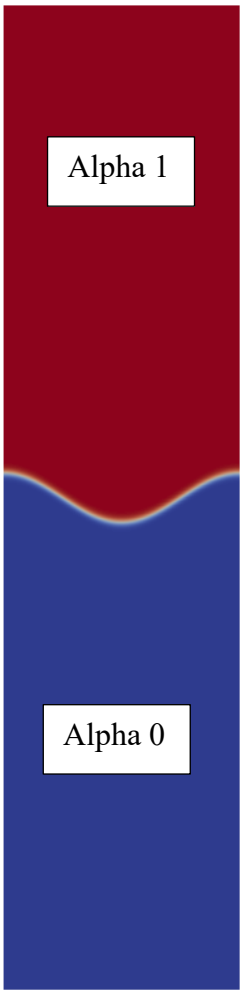


Fig.3 Geometry created with blockMesh (2D)

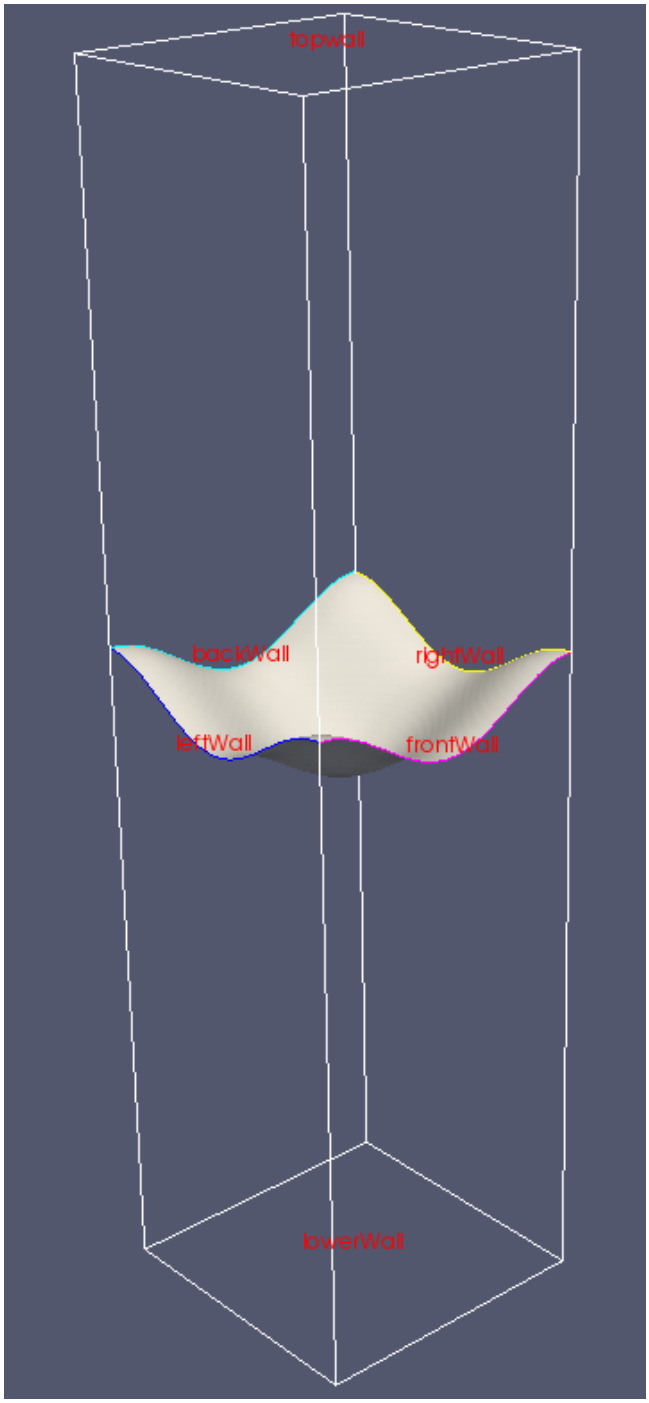


Fig.4 Geometry created with blockMesh (3D)

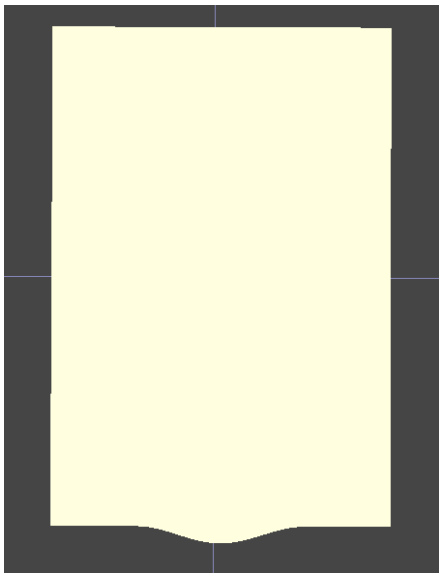


Fig.5 .stl file used for setFieldsDict

p_rgh:

```
dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    leftWall
    {
        type            symmetryPlane;
    }

    rightWall
    {
        type            symmetryPlane;
    }

    lowerWall
    {
        type            fixedFluxPressure;
        value           uniform 0;
    }

    topwall
    {
        type            fixedFluxPressure;
        value           uniform 0;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

The RT- Instability is gravity driven, hence p_rgh is initialised to be zero.

fvSchemes:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)  Gauss linear;
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         uncorrected;
}
```

transportProperties:-

```
phases (fluid 1 fluid2 );

fluid 1
{
   transportModel  Newtonian;

   nu              1e-03; // 1e-06-  for case 2
   rho             3;      //1000- for case 2
}

fluid 2
{
   transportModel  Newtonian;

   nu              3e-03; // 6e-05- for case 2
   rho             1;      // 950- for case 2
}


sigma                           0.000333; //0.037- for case 2
```

For **case III**, the density ratio is inversed to be 1:3, that is lighter fluid on top of denser fluid. For **cases IV and V**, transport properties of case I are used.

Boundary Conditions:

The boundary conditions for alpha , density and velocity fields files are created in 0 folder. They are presented below:

Fields	alpha.water	Density, p_rgh	Velocity, U
Internal field	**As per case	uniform 0;	uniform (0 0 0);
leftWall	type symmetryPlane;	type symmetryPlane;	type symmetryPlane;
rightWall	type symmetryPlane;	type symmetryPlane;	type symmetryPlane;
lowerWall	type zeroGradient;	type fixedFluxPressure; value uniform 0;	type noSlip;
topwall	type zeroGradient;	type fixedFluxPressure; value uniform 0;	type noSlip;
frontAndBack	type empty;	type empty;	type empty;

-For 3D case we have used symmtryPlane as Boundary condition for front and back walls.

symmetryPlane: to represent symmetry condition.

noSlip: to represent no slip condition at boundary.

fixedFluxPressure: zero gradient in pressure in case there is body forces like gravity and surface tension in this case.

zeroGradient: to represent no normal gradient at wall

decomposeParDict: We split the whole domain using decomposeParDict so that each part can be solved by individual processor for the interest of time.

```
numberOfSubdomains 4;
// no domains we divide the whole domain which will be solved by each processor individually

method          simple;

simpleCoeffs
{
    n             (1 4 1);
    delta         0.001;
}

hierarchicalCoeffs
{
    n             (1 1 1);
    delta         0.001;
    order         xyz;
}

manualCoeffs
{
    dataFile      "";
}

distributed     no;

roots           ( );
```

After the end of simulation, the results will in a number of processor files that are created during **decomposePar**.Now we have to use the command **reconstructPar** to collect the results all together for a single domain.

jobScript: It is basically used to simulation in a cluster under a specific set of sequence.

```
#!/bin/sh
#
#This is an example script to execute a parallel job with OpenFOAM
# ! Be sure to have decompose using decomposePar before submitting this script
#
#These commands set up the Grid Environment for your job:
# Output folder for the journal file
#PBS -o /nfs/home/ofoam4/OpenFOAM/ofoam-6/project/c1/RT1
//path of the case where it is saved in the cluster
#PBS -j oe
# Name of the job
#PBS -N experiment3procs
#PBS -q batch
#PBS -l nodes=1:ppn=4:buddies
# E-mail address at which notification will be sent
#PBS -M ge65rot@mytum.de
#PBS -m abe

# Load OpenFOAM-6
source /nfs/etc/bashrc
OF60
# Change the current directory to the case directory
cd /nfs/home/ofoam4/OpenFOAM/ofoam-6/project/c1/RT1
# Execute interFoam on the current case directory
mpiexec interFoam -parallel > log.interFoam
// the sequence in which the operation will take place
```

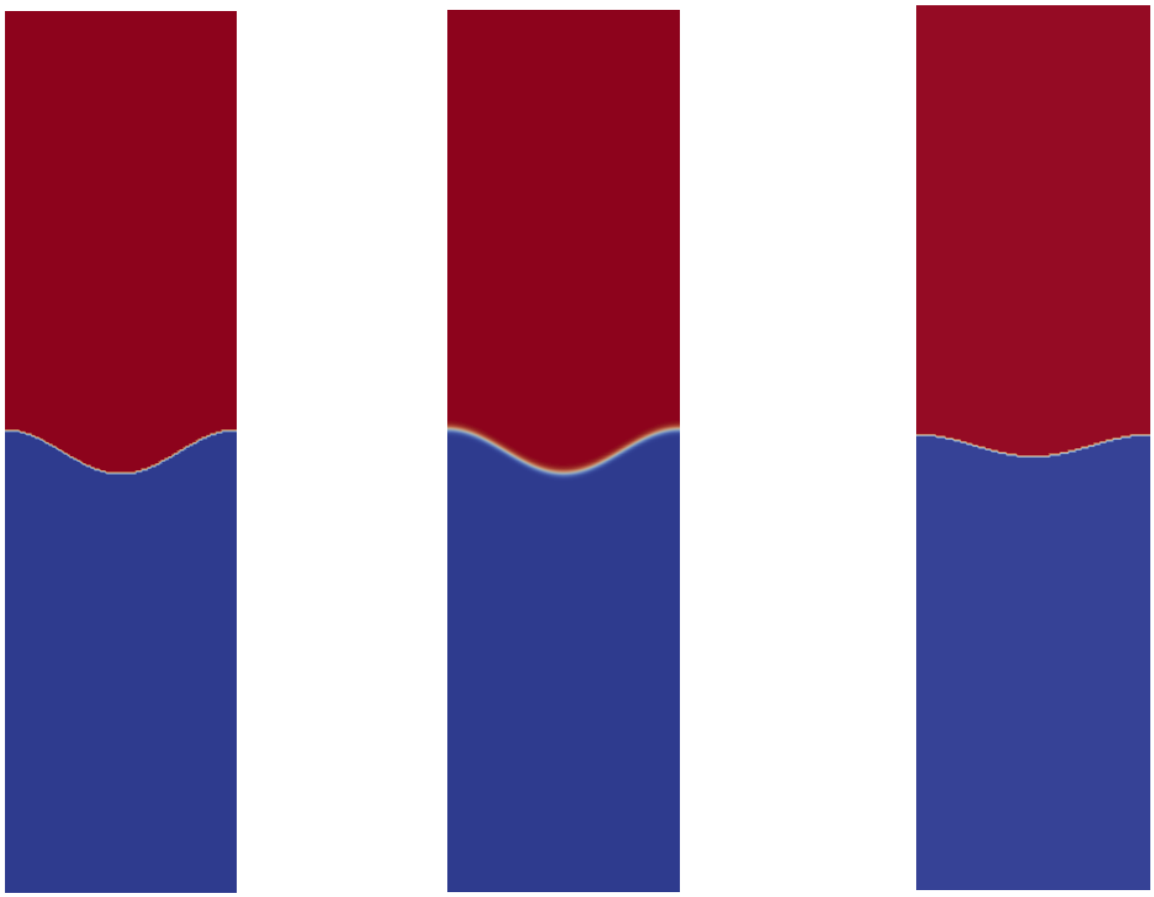
Solver interFoam:

The solver solves the Navier Stokes equations for two incompressible, isothermal immiscible fluids. That means that the material properties are constant in the region filled by one of the two fluids except at the interphase. It uses PIMPLE algorithm to solve the equations and p: pressure [Pa], p_rgh: pressure - hydrostatic contribution [Pa] and U: velocity [m/s] are mandatory fields to me given input.

3. Results

A. Case 1:

In this case the density ratio of denser fluid to lighter fluid is 3:1 and interface initialization is done by using funkySetFields, codeStream, setFieldsDict. The contour progression of the all three initialization cases at certain same time is shown below:

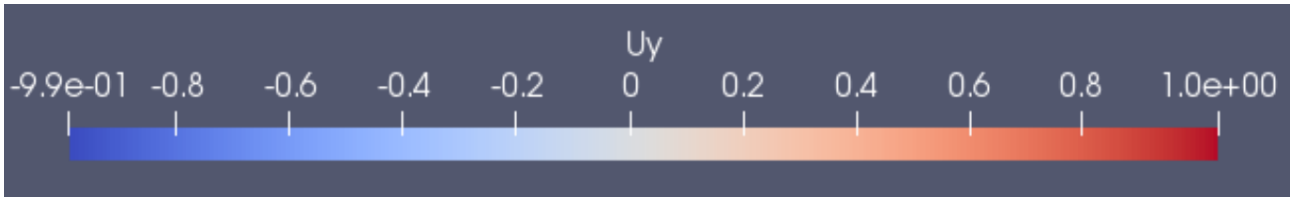


funkySetFields (0.1) codeStream(0.1) setFieldsDict(0.05)

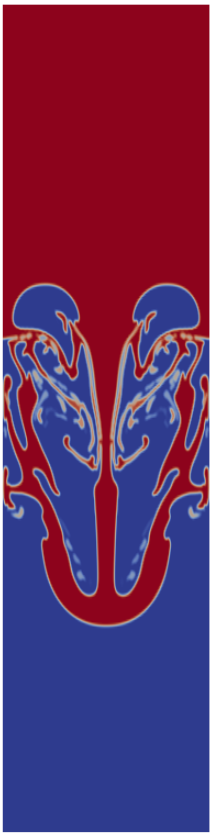
Fig.6 Contour at t=0 seconds

Legends:-

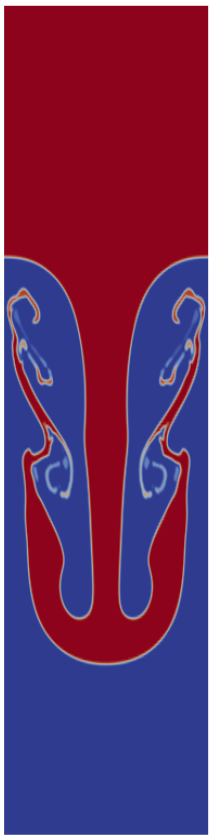
-for all alpha contours



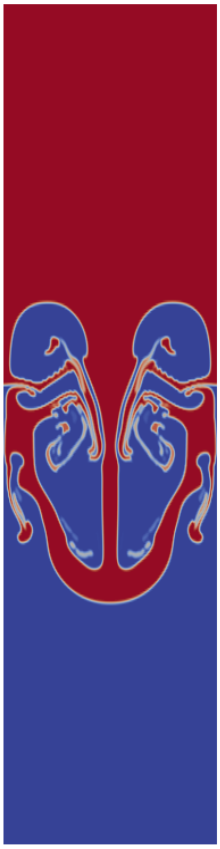
-for all velocity profiless



funkySetFields

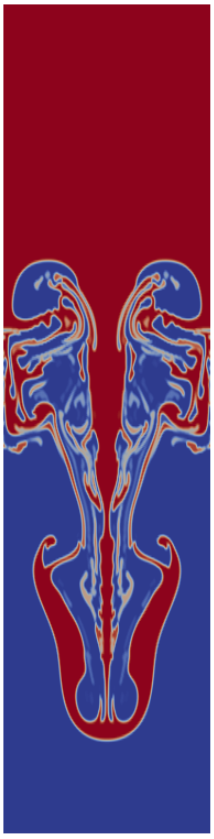


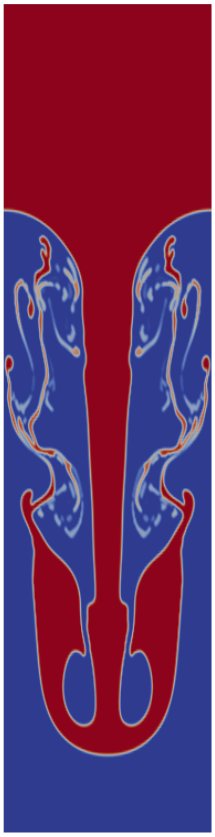
codeStream



setFieldsDict

Fig.7 Contour at t=1.3 seconds





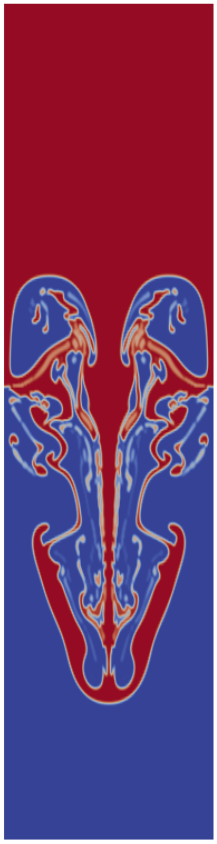


Fig.8 Contour at t=1.6 seconds

B. Case I and II:

In this case I (density ratio 3:1, At =0.5) and case II (density ratio 1.05:1, At-0.025) profile contours at same time are shown below :

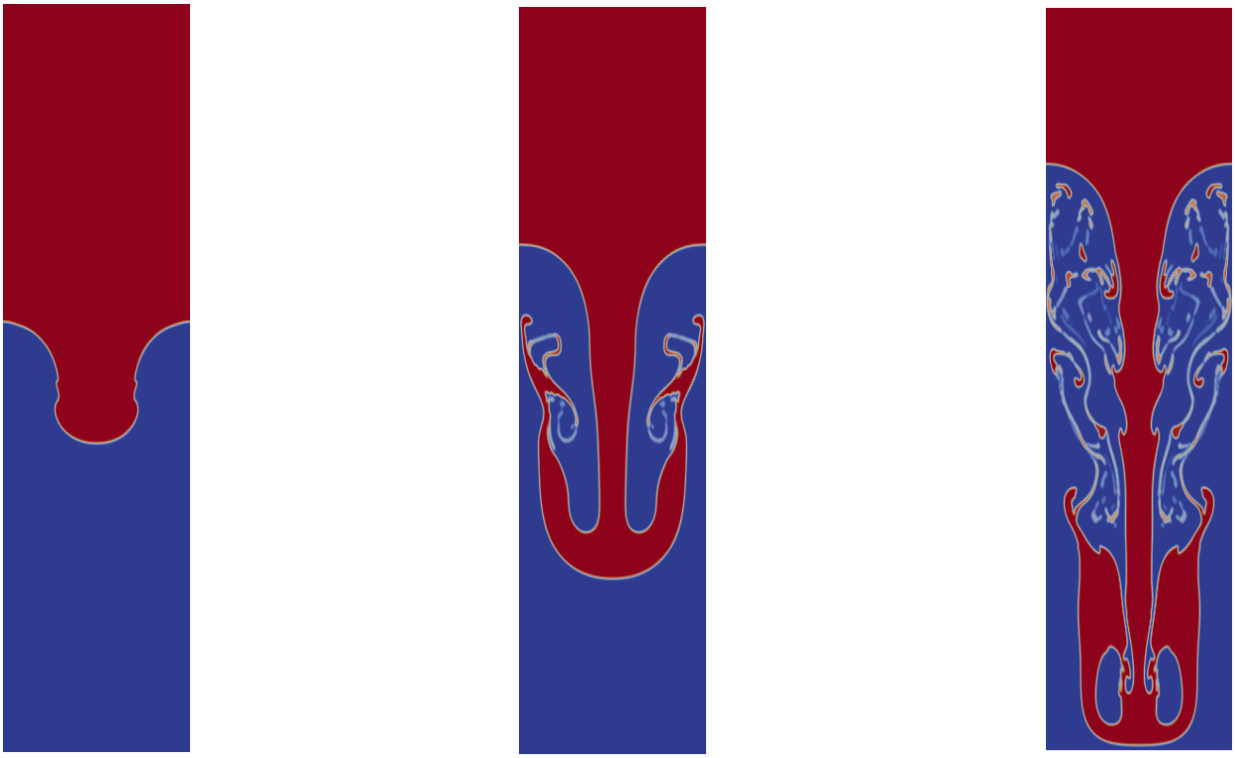


Fig.9 Case I contours at t = 0.5s, 1.2s, 1.8s

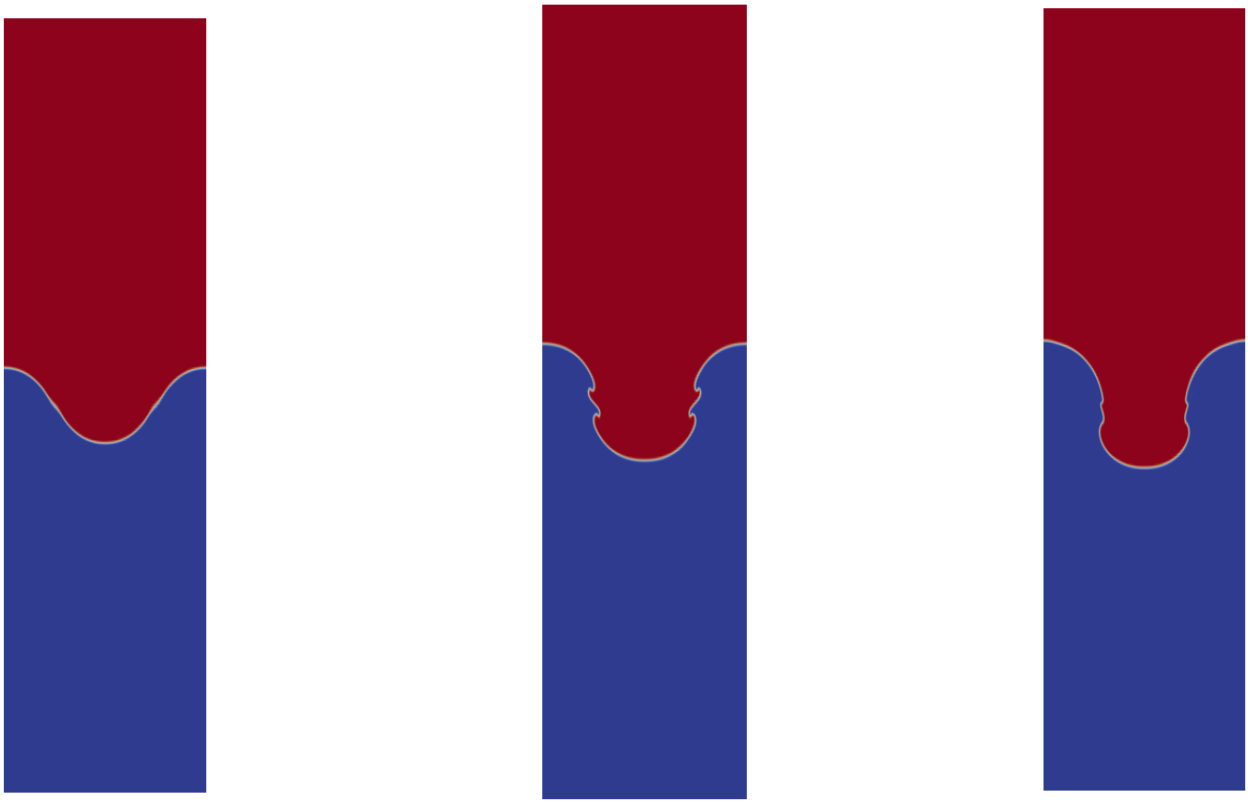


Fig10. Case II contours at t = 0.5s, 1.2s, 1.8s

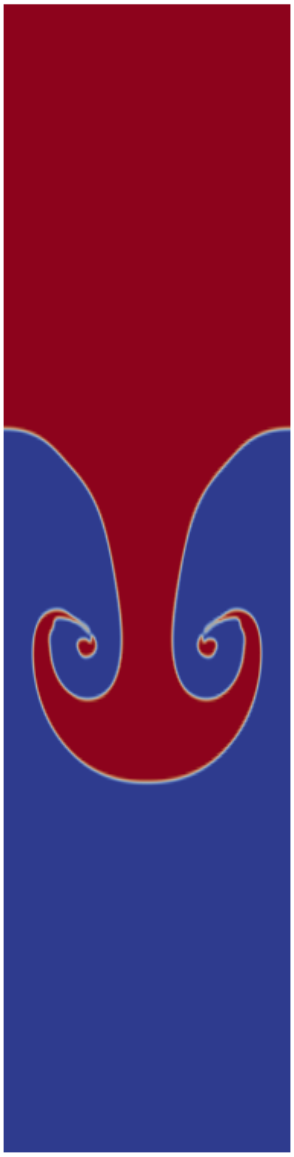


Fig11. Case I Bubble-type Plume (At = 0.5)

Fig.12 Case II Finger-type Plume (At=0.025)

C. Case I and III:

In this case I (density ratio 3:1) and case III (density ratio 1:3) profile contours at same time are shown below :

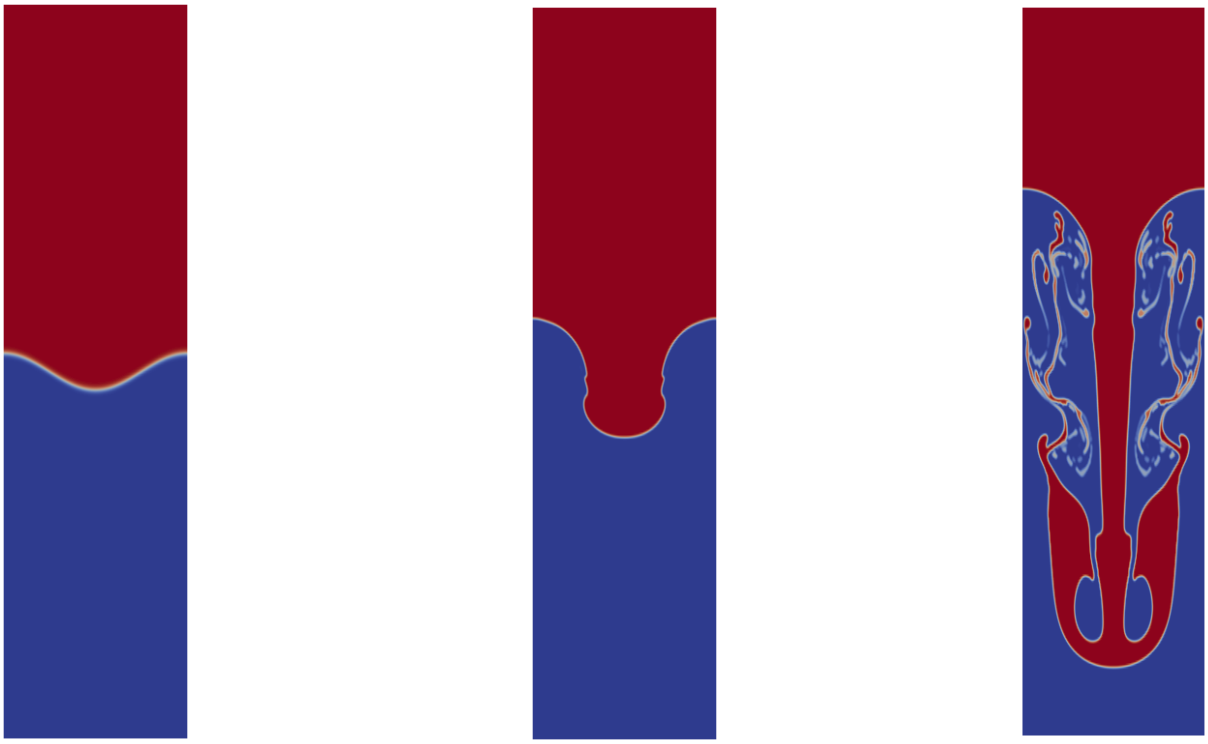


Fig.13 Case I contours at t = 0s,0.5s, 1.6s

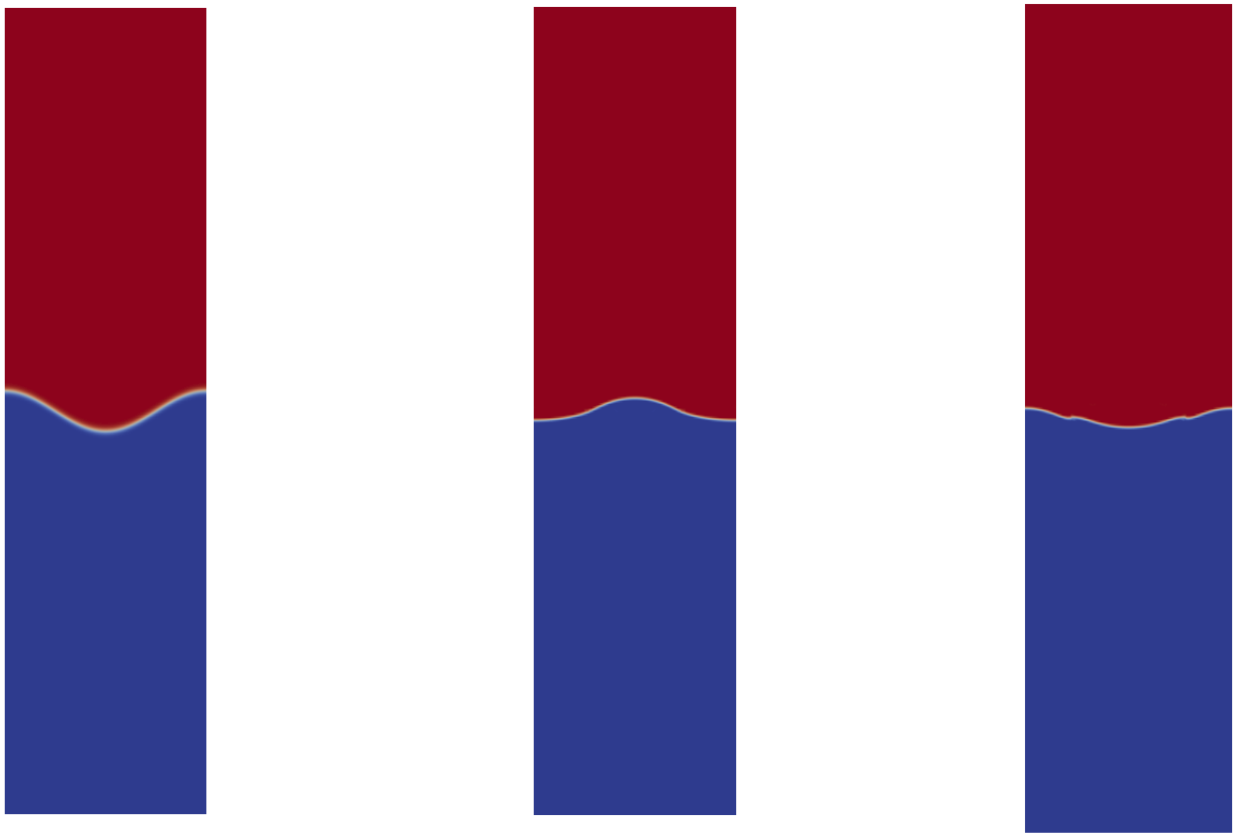


Fig.14 Case III contours at t = 0s,0.5s, 1.6s

D. Case IV:

The 3D RT instability plume contours are shown below:

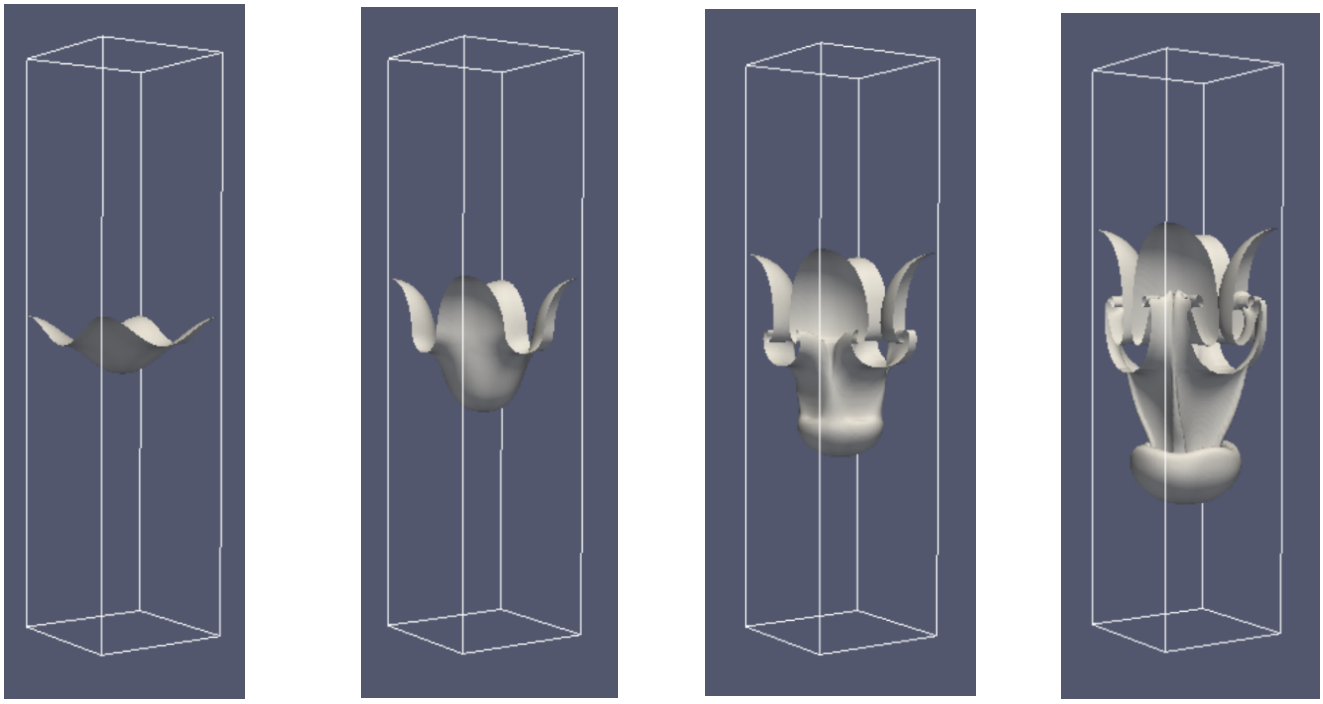


Fig.15 Case IV contours at t = 0s, 0.4 s, 0.6s, 0.8s

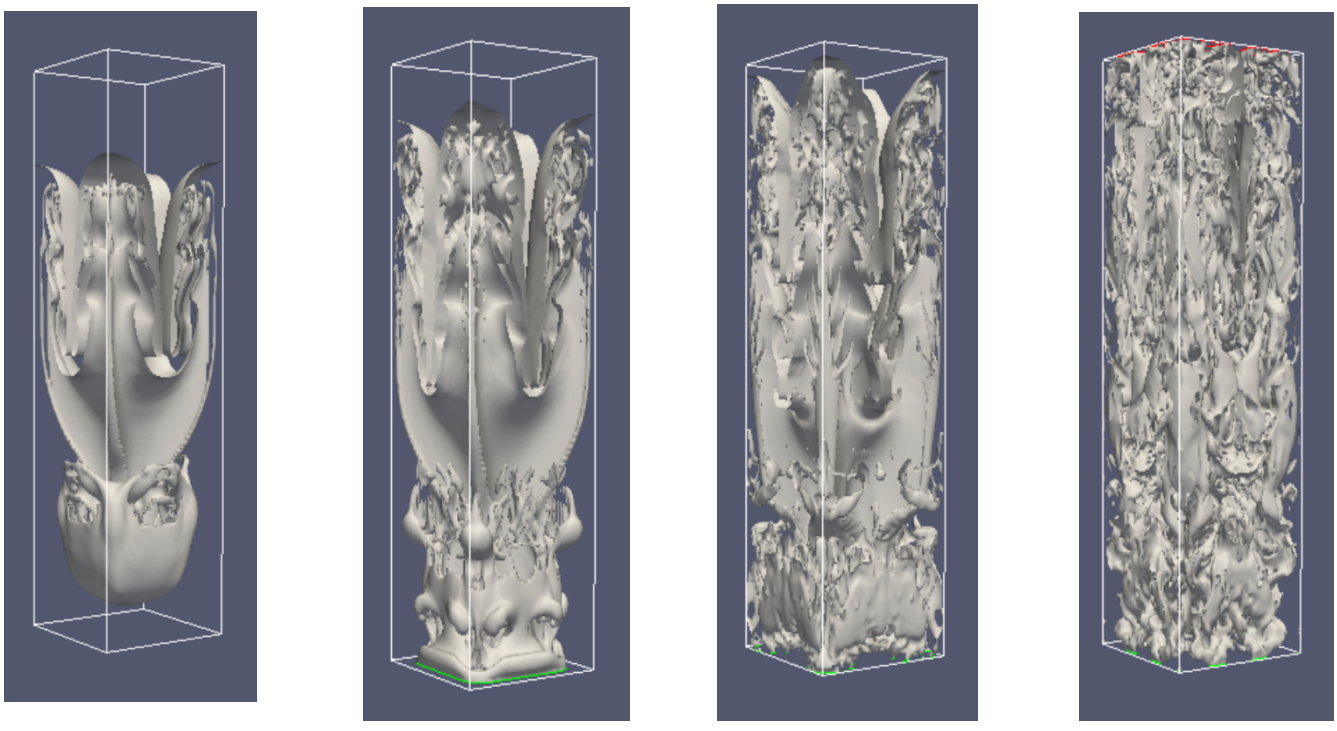


Fig.16 Case IV contours at t = 1.2s, 1.4s, 1.6s, 2s

E. Case V:

Evolution of RT for very long time.

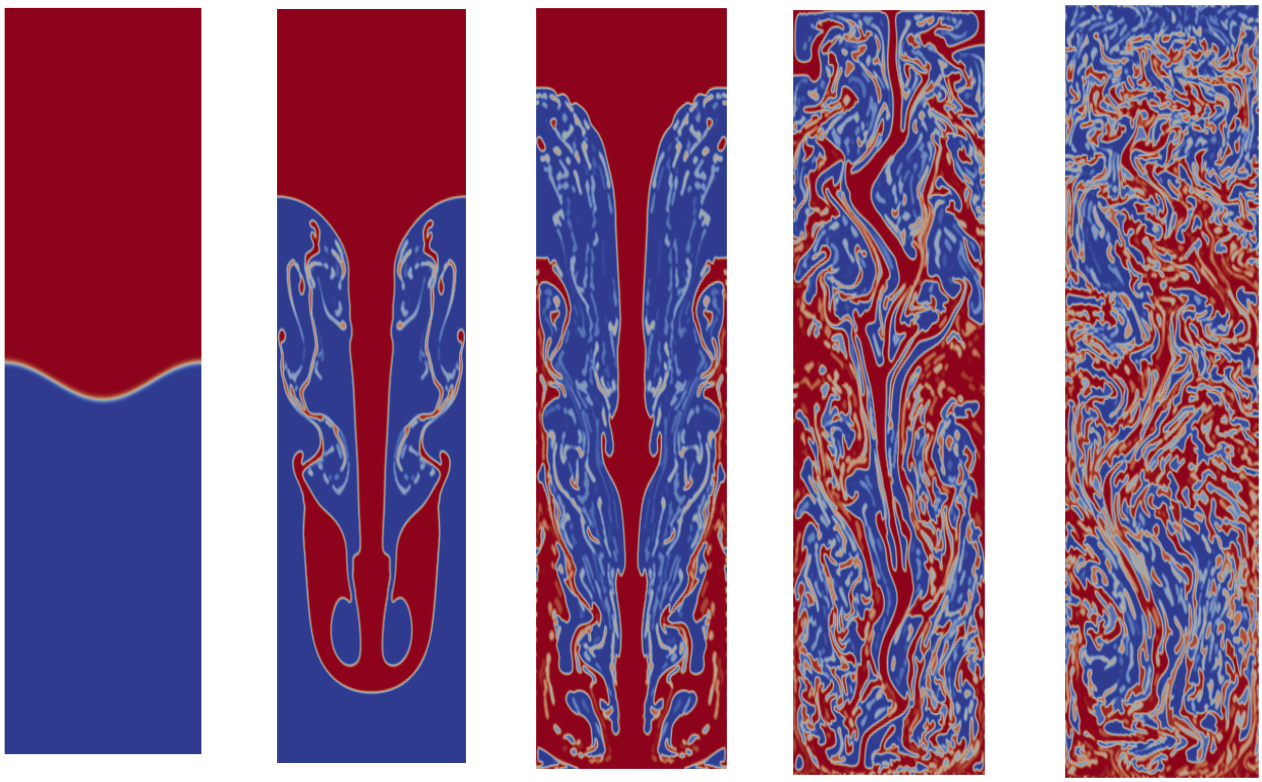


Fig.17 Case V contours at t = 0s, 1.6s, 2.4s, 3.6s, 5s

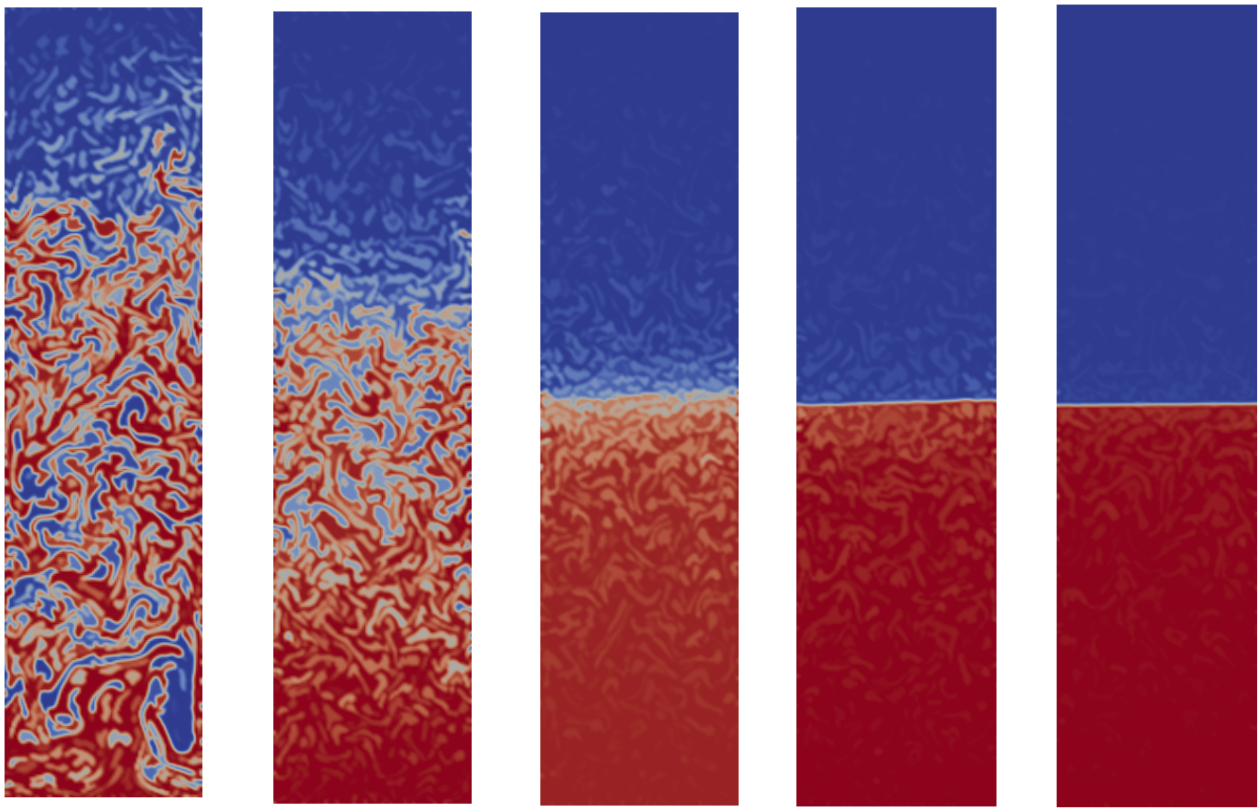


Fig.18 Case V contours at t = 10s, 20s, 40s, 60s, 100s

F. Velocity profiles of Case I & II- Uy

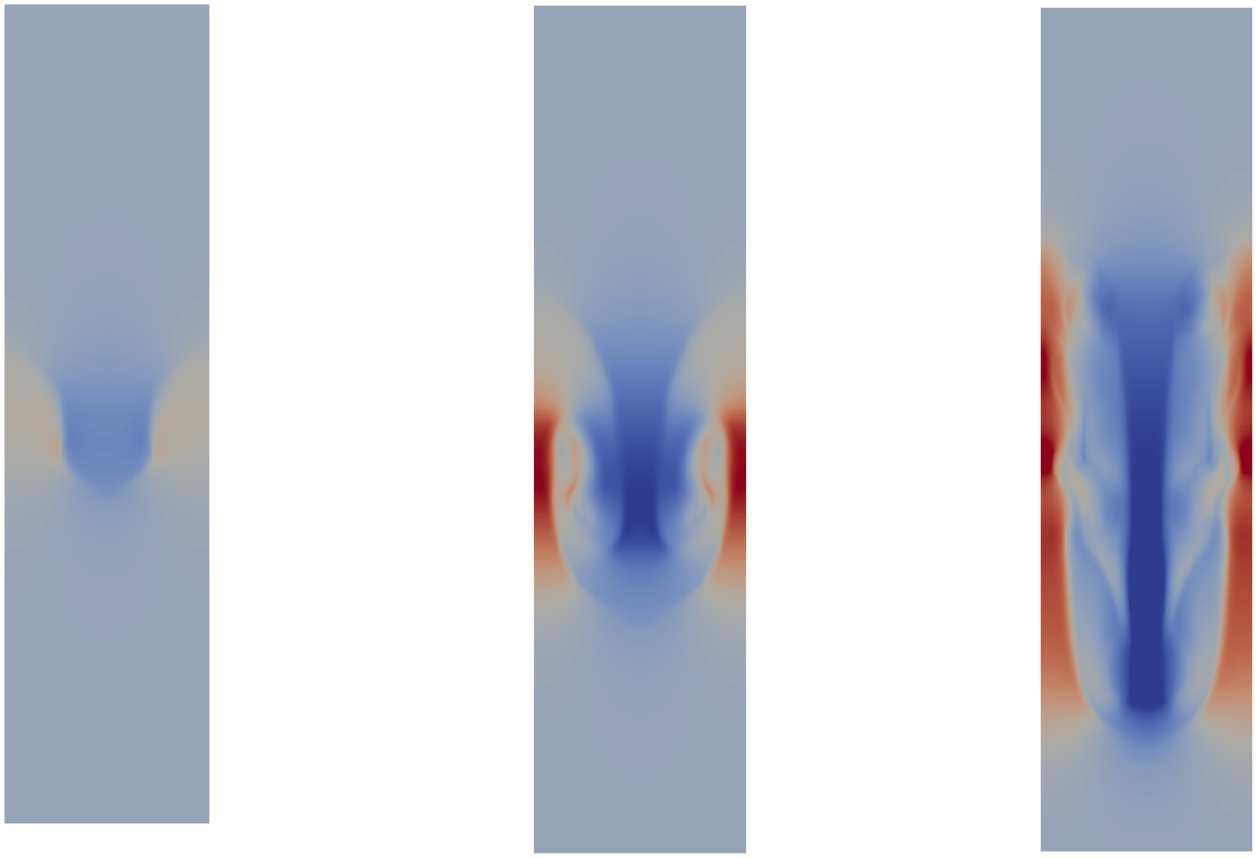


Fig.19 Case I velocity profiles at t =0s, 1s, 1.5s

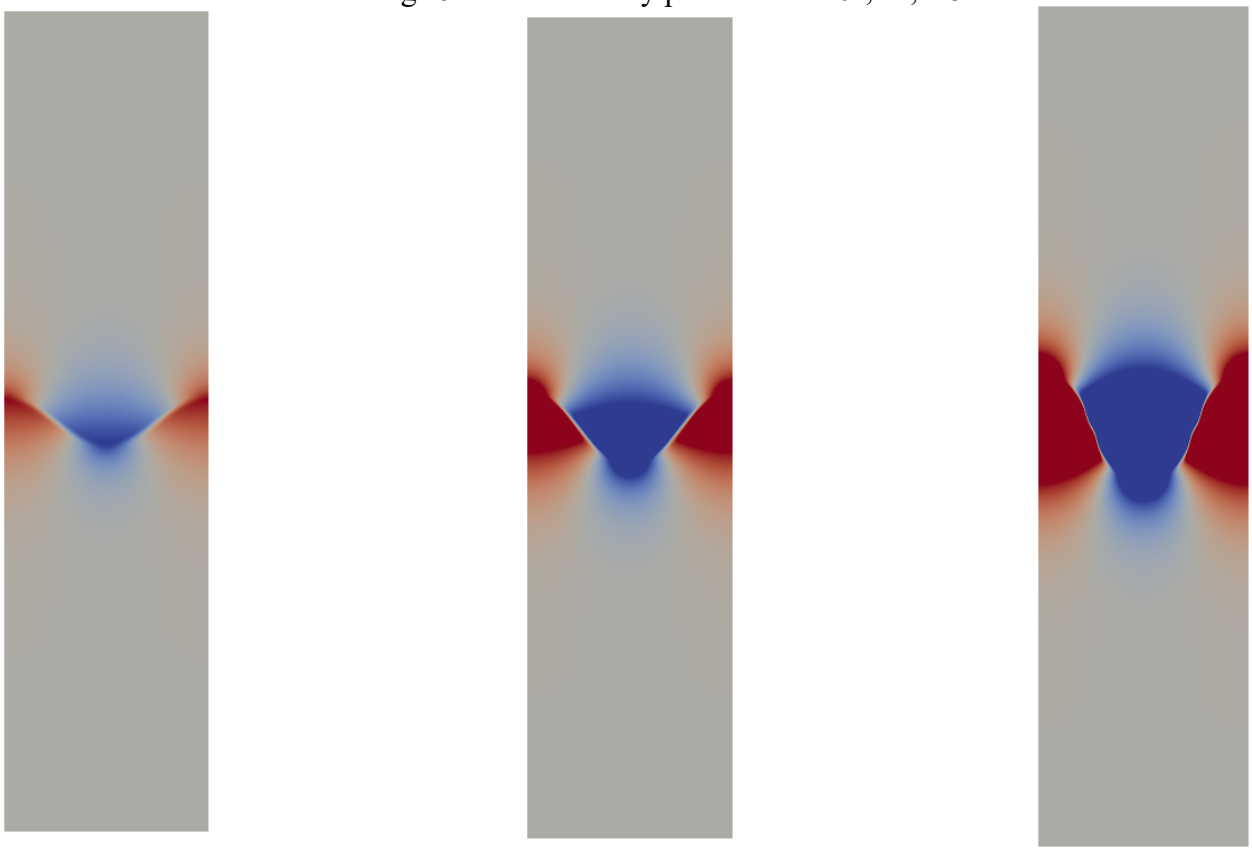


Fig.20 Case II velocity profiles at t =0s, 1s, 1.5s

G. Graphs:

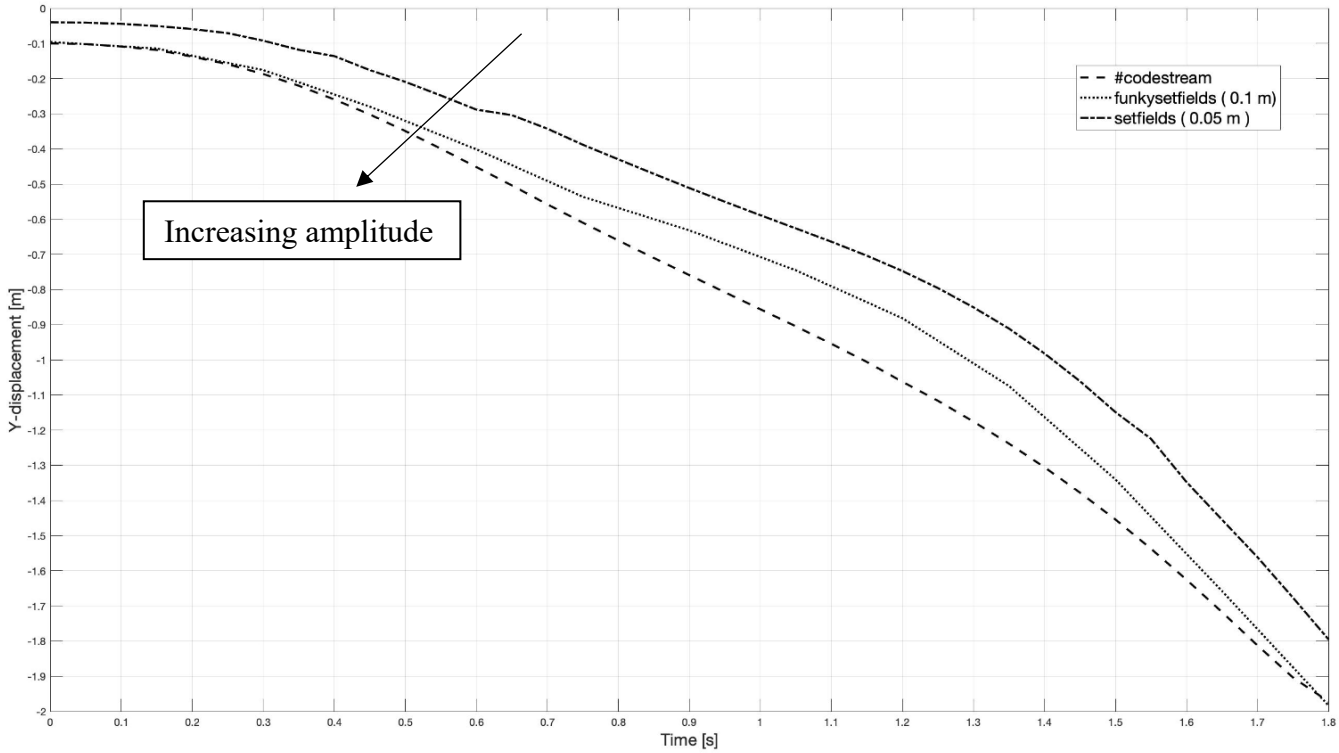


Fig.21 Case I: Max y-displacement vs time plot for different interface initialization

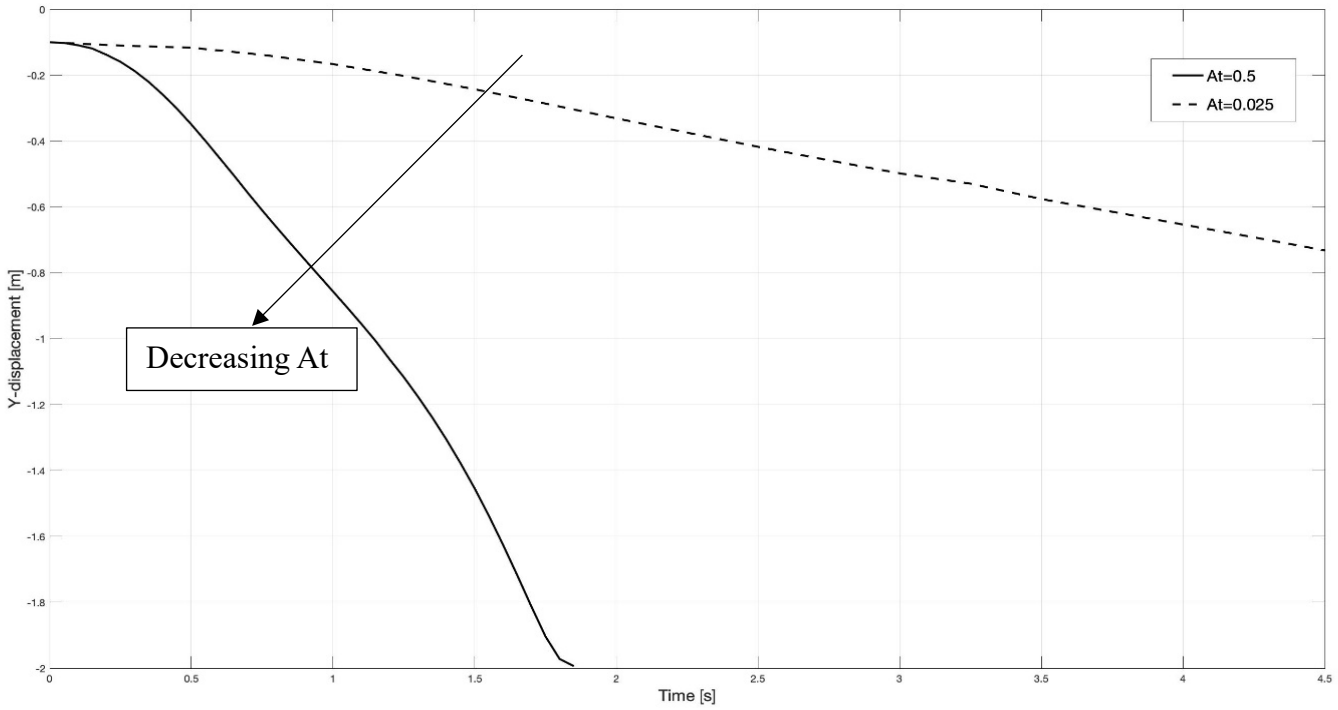


Fig.22 Max y-displacement plot vs time plot for case I & II (i.e. Atwood number 0.5 & 0.025 respectively)

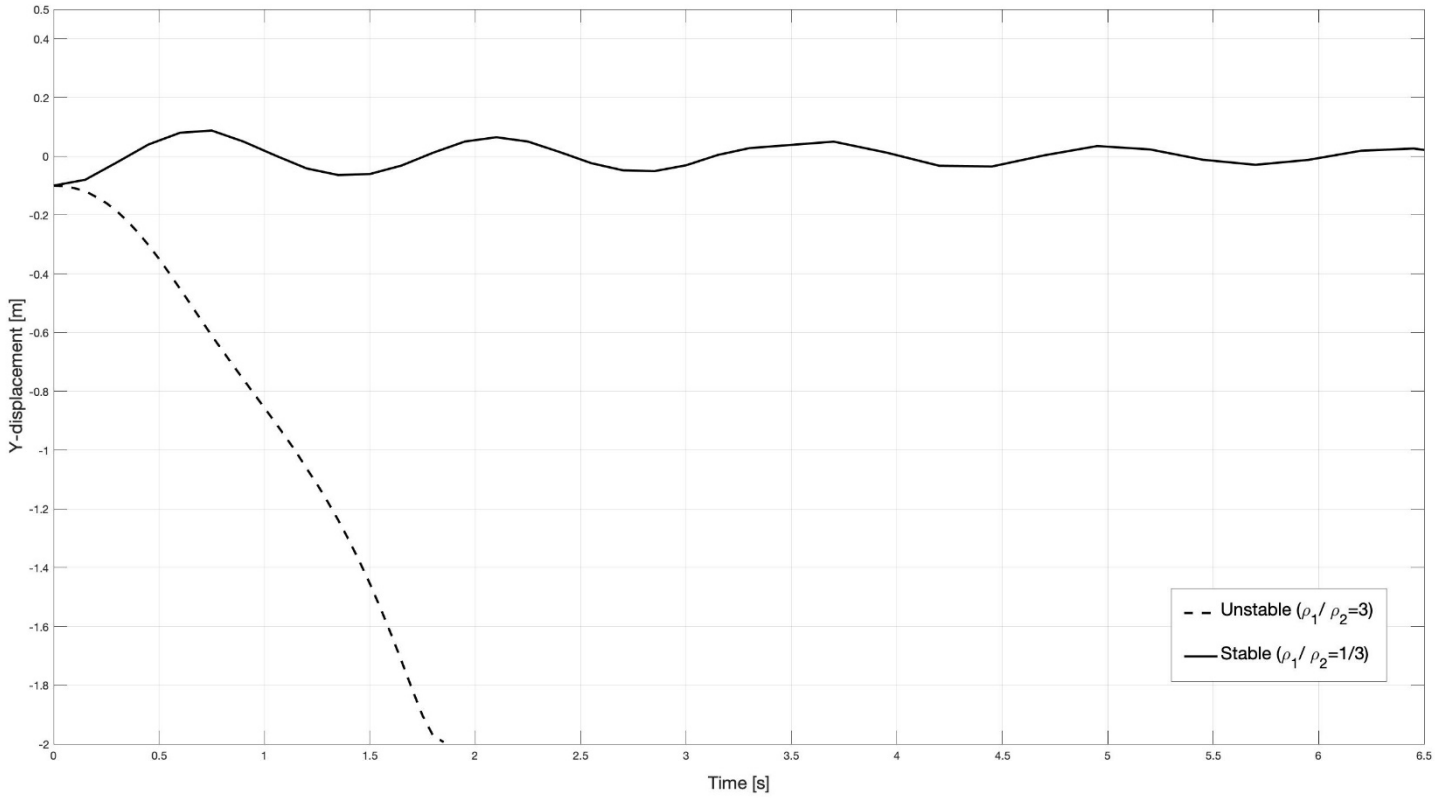


Fig.23 Max y-displacement vs time plot for case I and III

4. Conclusion

The simulations are performed for incompressible immiscible fluids under the influence of gravitation acceleration directed in negative y direction. As RTI is a transition phenomenon we have restricted to laminar case.

A. From **Result A**, as seen from graph Fig. 21 and contours Fig.6,7&8; the y-displacement progression follow same pattern for funkySetFields and setFieldDict interface initialization. The progression is slower for setFieldDict as its amplitude is smaller than of funkySetFields.The diffusion of interface is much more finer for codestream case , and much more chaotic for the other two.But as seen from the Fig 21, the y displacement follows the same nature for all the three cases.

B. From **Result B**, as seen in the contours Fig 9&10 and graph Fig. 22 that the larger is the difference between density of the two fluids, faster is the evolution. The surface tension also do play an important role as higher its value more slower will be the evolution. Also, as seen in Fig. 11&12 in terms of Atwood number, closer it is to '0' it will have finger-type evolution and closer it is to '1', it will have bubble-type evolution of plume. The RTI shows strong dependence on Atwood number and reducing it would delay its evolution.

C. From **Result C,** as seen in Fig 13 & 14 and graph (Fig.23) it has been observed that the stability condition (i.e. damping of perturbation) occurs when higher density fluid is below lower density fluid and it is unstable when higher density fluid is above the lower density fluid.

D. In **Result D**, as seen in the Fig.15&16; the evolution of 3D RT instability is shown. The initial perturbation leads to plume formation and then to smaller vortices from bigger one, and eventually turning into fully turbulent case.

E. In **Result E**, as seen in the Fig.17&18; 2D RT instability case has been simulated over a very long period of time. After very long period, the denser fluid comes down and settles below the lighter fluid. The observation is obvious as the RT stability is driven by gravity.

F. In **Result F**, velocity profiles of Case I & II are shown in Fig. 19 & 20. As seen from the figures, analogous to 'y-displacement' contours, velocity profile evolution for Case I is faster compared to Case II.

5. Validation

We have performed our simulation after studying different research papers and tried to bring close resemblance with them. Though analytical solution are specific cases, we tried to match our results. The analytical solution approach developed through linear theory is stated below:

$$\omega^2 = g\left(\frac{\rho_2 - \rho_1}{\rho_2 + \rho_1}\right)k - \left(\frac{\sigma}{\rho_2 + \rho_1}\right)k^3$$

The evolution of the displacement is given by: $\eta(t) = \eta(0)cosh(\omega t)$.
We are stressing on the fact that this solution is based on linear approach, so exact resemblance with the simulation result will be difficult.

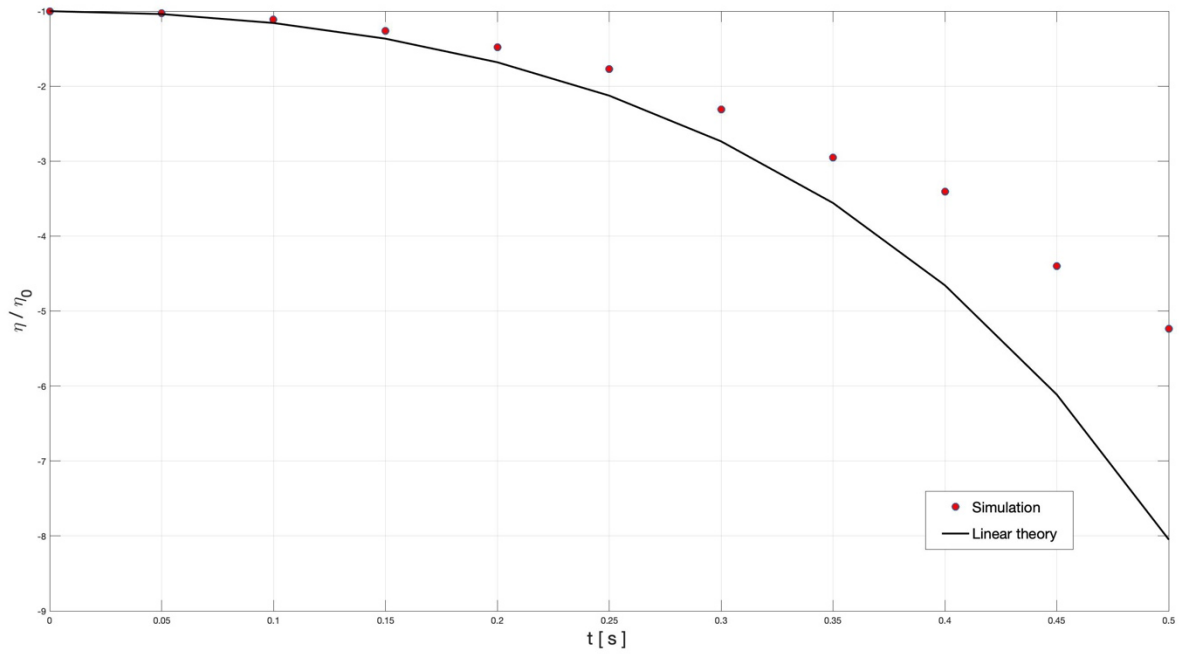


Fig 24. Comparison of the simulation data with linear theory analytical solution

As it can be inferred from the plot is that, with the passage of time, the simulation data shows increasing deviation from the linear analytical solution.
The resemblance with the simulation data is highly dependent on the initial amplitude of perturbation. The lower it is, the more closely it will behave with the linear theory.The contours that we have shown shows close resemblance with the RT research papers that we can taken as reference.

6. Bibliography

- An overview of Rayleigh Taylor Instability , D.H Sharp
- Numerical simulation of the three-dimensionl Rayleigh–Taylor instability, Kim, Lee
- Simulation of the two-dimensional Rayleigh-Taylor instability problem by using diffuse-interface model, Khan, Shah

Note: The case files setup are stored in the cluster in the path: /nfs/home/ofoam4/OpenFOAM/ofoam-6/FinalProject/finalproject

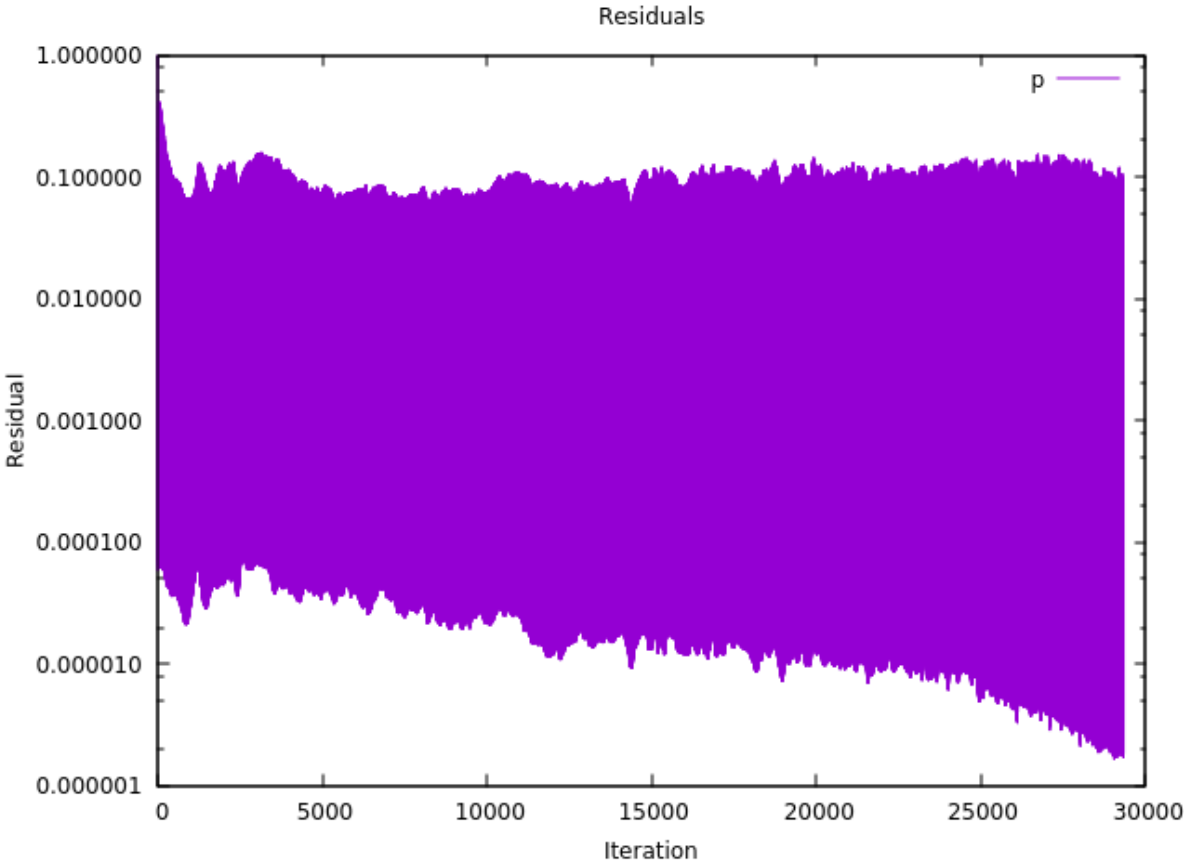


Fig 25. Residual plot for p_rgh

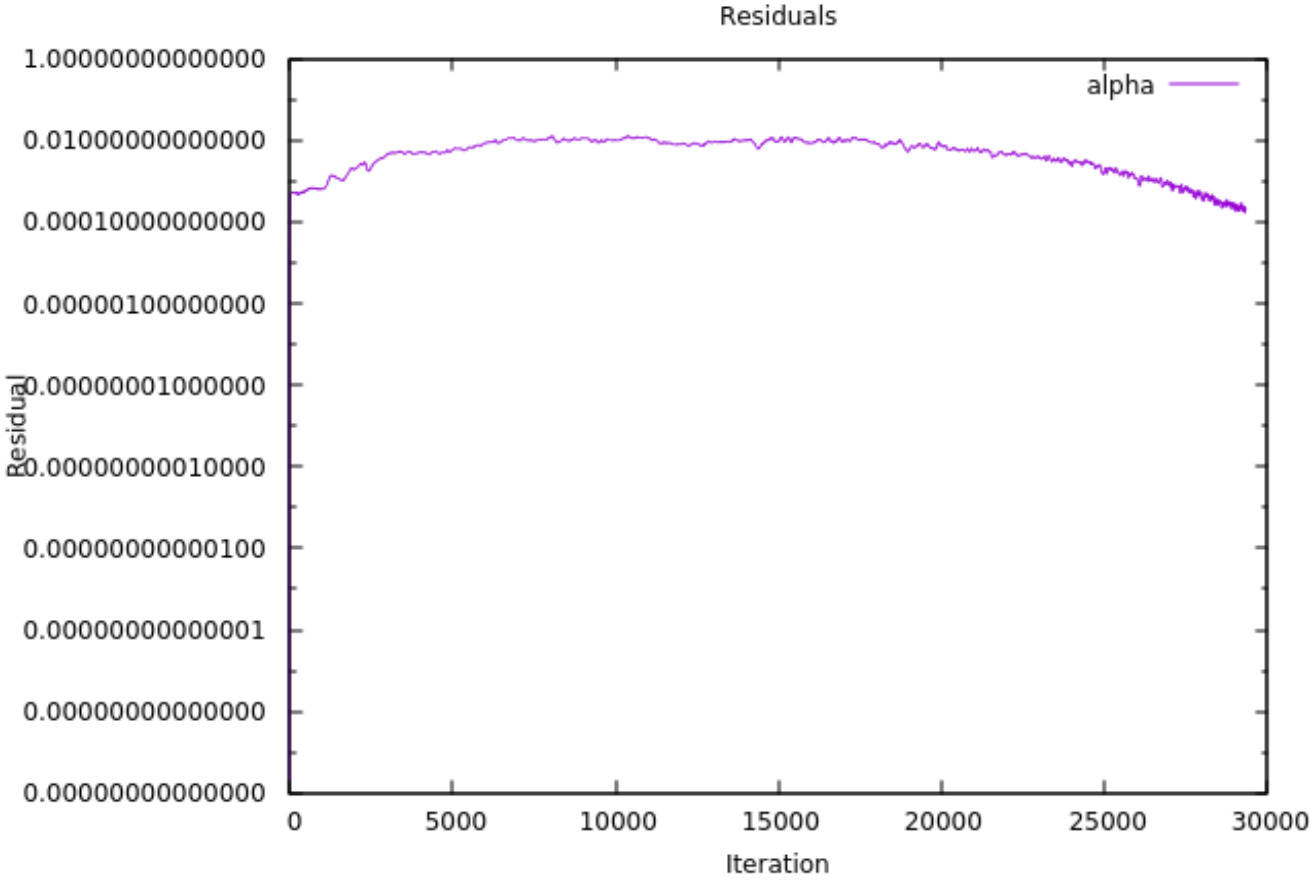


Fig 26. Residual plot of alpha