

Solution initialization using **codeStream**

- When it comes to initial conditions, you can use the utility `setFields`.
- This utility is very flexible, you can even read STL files and use them to initialize your fields.
- But in case that you can not get the desired results using `setFields`, you can implement your own initial conditions using **codeStream**.
- To implement initial conditions using **codeStream**, we proceed in a similar way as for boundary conditions.
- The source code and binaries are automatically generated and copied in the directory **dynamicCode** of the current case.
- The source code is compiled automatically at run-time.
- The use of **codeStream** is a very good alternative to avoid high level programming of initial conditions or the use of external libraries.
- Hereafter we will use **codeStream** to implement new initial conditions.

Solution initialization using codeStream

Body of the **codeStream** directive for initial conditions

```
internalField  #codeStream
```

```
{  
  {  
    codeInclude  
    #{  
      #include "fvCFD.H"  
    };  
  
    codeOptions  
    #{  
      -I$(LIB_SRC)/finiteVolume/lnInclude \  
      -I$(LIB_SRC)/meshTools/lnInclude  
    };  
  
    codeLibs  
    #{  
      -lmeshTools \  
      -lfiniteVolume  
    };  
  
    code  
    #{  
      };  
    };  
  }  
}
```

Initial conditions

Use codeStream to set the value of the initial conditions

Files needed for compilation

Compilation options

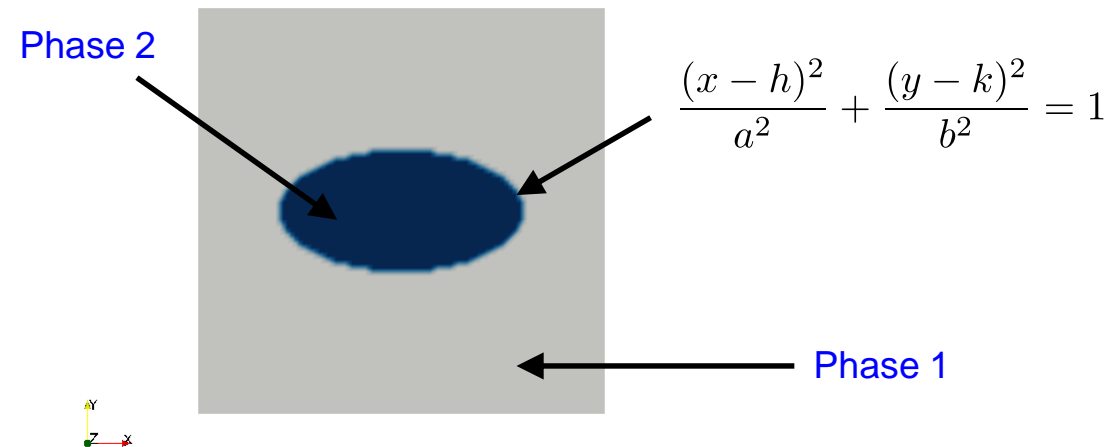
Libraries needed for compilation. Needed if you want to visualize the output of the initial conditions at time zero

Insert your code here.
At this point, you need to know how to access internal mesh information

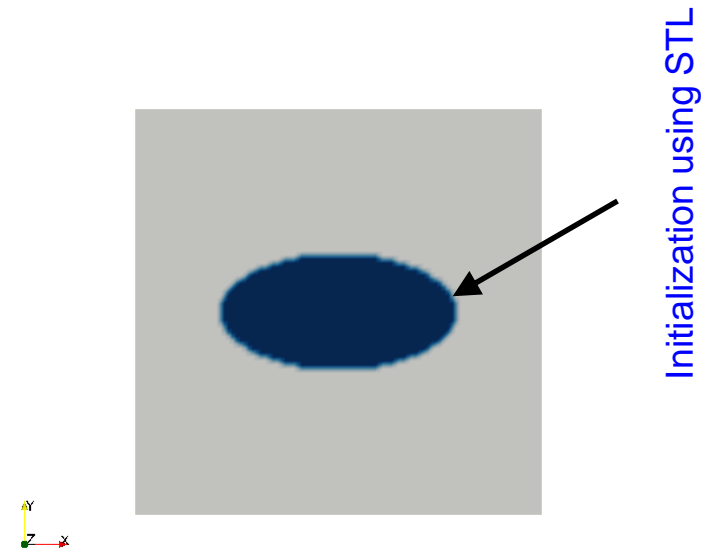
Solution initialization using codeStream

Implementation of an elliptic initialization using **codeStream**

- Let us implement an elliptic initialization using **codeStream**.
- The first step is to know your domain and identify the region that you want to initialize.
- Then you will need to do a little bit of math to get the expression for the initialization.
- In this example, we are also going to show you how to do the same initialization by reading a STL file with the utility `setFields`.



Initialization using **codeStream**



Initialization using a STL with `setFields`

Solution initialization using codeStream

- The **codeStream** IC in the body of the file *alpha.phase1* is as follows,

```
internalField  #codeStream
```

Use codeStream to set the value of the initial conditions

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

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

    codeLibs
    #{
      -lmeshTools \
      -lfiniteVolume
    };

    code
    #{
    };
  };
}
```

Depending of what are you trying to do, you will need to add new files, options and libraries.

For most of the cases, this part is always the same.

Insert your code here.
At this point, you need to know how to access internal mesh information

Solution initialization using codeStream

- The **code** section of the **codeStream** IC in the body of the file *alpha.phase1* is as follows,

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

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

    scalar he = 0.5;
    scalar ke = 0.5;
    scalar ae = 0.3;
    scalar be = 0.15;

    forAll(alpha, i)
    {
        const scalar x = mesh.C()[i][0];
        const scalar y = mesh.C()[i][1];
        const scalar z = mesh.C()[i][2];

        if ( pow(y-ke,2) <= ((1 - pow(x-he,2)/pow(ae,2) ) * pow(be,2)) )
        {
            alpha[i] = 1.;
        }
        alpha.writeEntry("", os);
    }
};
```

Access internal mesh information

Initialize scalar field to zero

Initialize variables

forAll loop to access cell centers and to assign alpha values. Notice the alpha was previously initialized. The size of the loop is defined by alpha and the iterator is i.

Access cell centers coordinates

Assign value to alpha

Write output to input dictionary

If this condition is true, do the following statement

$$(y - k)^2 \leq \left(1 - \frac{(x - h)^2}{a^2}\right) \times b^2$$

Solution initialization using codeStream

Implementation of an elliptic initialization using **codeStream**

- This case is ready to run, the input files are located in the directory `$PTOFC/101programming/codeStream_INIT/elliptical_IC`
- To run the case, type in the terminal,
 1. `$> cd $PTOFC/101programming/codeStream_INIT/elliptical_IC`
 2. `$> foamCleanTutorials`
 3. `$> blockMesh`
 4. `$> rm -rf 0`
 5. `$> cp -r 0_org 0`
 6. `$> paraFoam`
 7. `$> interFoam | tee log`
 8. `$> paraFoam`
- In step 6, we launch `paraFoam` to visualize the initialization.
- FYI, you can run in parallel with no problem.

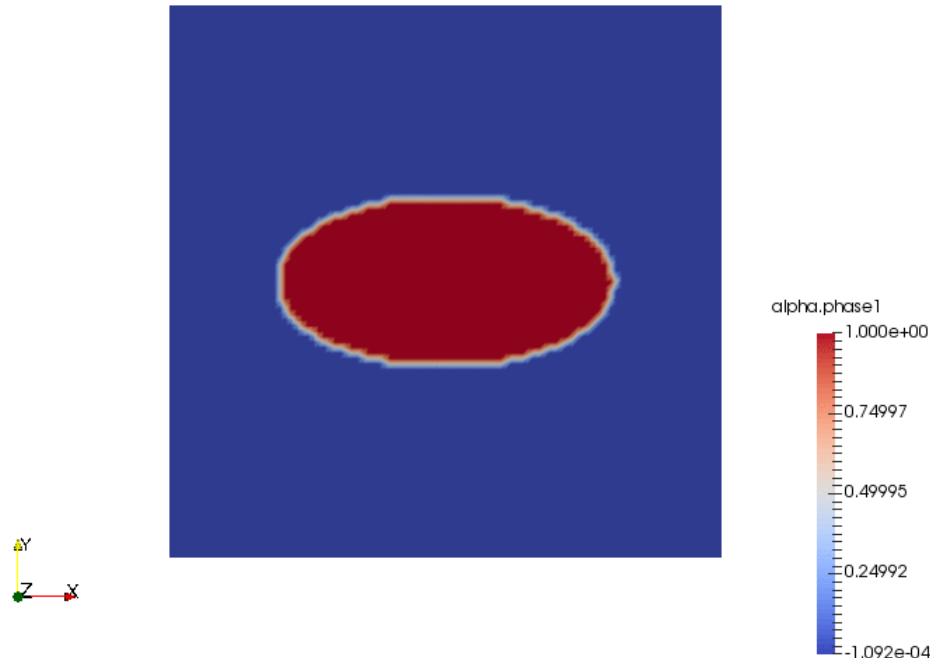
Solution initialization using codeStream

Implementation of an elliptic initialization using **codeStream**

- If everything went fine, you should get something like this



Time: 0.000000



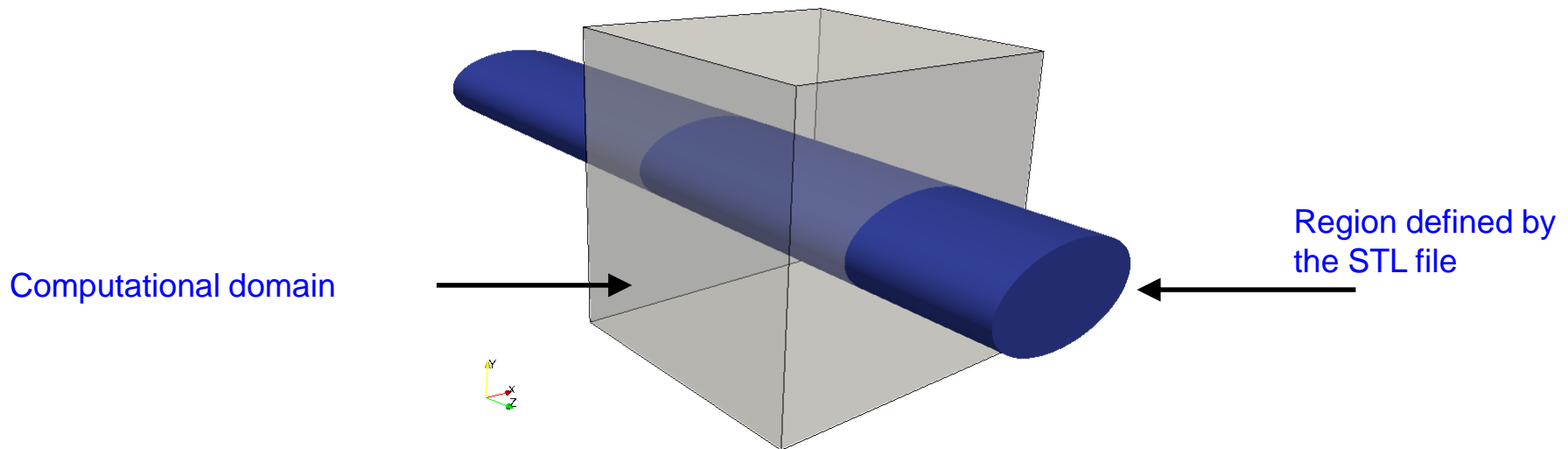
Visualization of volume fraction (alpha.phase1)
Surface tension driven flow or bubble in a zero gravity flow using interFoam

www.wolfdynamics.com/wiki/BCIC/bubble_zeroG.gif

Solution initialization using `codeStream`

Elliptic initialization using **setFields**

- Let us do the same initialization using a STL file with `setFields`.
- First you will need to create the solid model that encloses the region you want to initialize. For this you can use your favorite CAD/solid modeling software. Remember to save the geometry is STL format.
- Then you will need to read in the STL file using `setFields`. You will need to modify the `setFieldsDict` dictionary.



Solution initialization using codeStream

The **setFieldsDict** dictionary

```
defaultFieldValues  
(  
    volScalarFieldValue alpha.phase1 0  
);
```

}

Initialize the whole domain to zero

```
regions  
(
```

setFields method to read STL files.
If you want to know all the options
available use a word that does not exist
in the enumerator list (e.g. banana)

```
    surfaceToCell  
    {
```

```
        file "../geo/ellipse.stl";
```

Location of the STL file to read

```
        outsidePoints ((0.5 0.85 0));
```

A point located outside the STL

```
        includeInside true;
```

Use what is inside the STL

```
        includeOutside false;
```

Use what is outside the STL

```
        includeCut false;
```

Include cells cut by the STL

```
        fieldValues  
        (  
            volScalarFieldValue alpha.phase1 1  
        );
```

Initialize this value.
In this case the initialization will be inside
the STL

```
    }  
);
```

Solution initialization using codeStream

Elliptic initialization using **setFields**

- This case is ready to run, the input files are located in the directory `$PTOFC/101programming/codeStream_INIT/elliptical_IC`
- To run the case, type in the terminal,

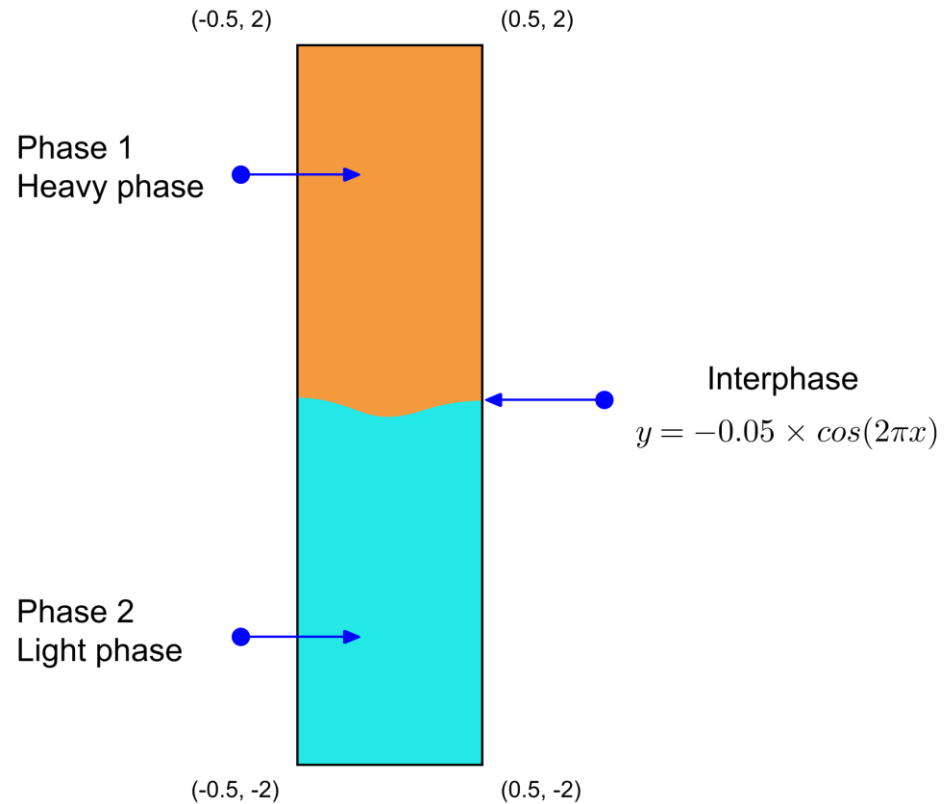
```
1. $> cd $PTOFC/101programming/codeStream_INIT/elliptical_IC
2. $> foamCleanTutorials
3. $> blockMesh
4. $> rm -rf 0
5. $> cp -r 0_org 0
6. $> setFields
7. $> paraFoam
```

- At this point, compare this initialization with the previous one.
- Also, feel free to launch the simulation using `interFoam`.

Solution initialization using codeStream

Rayleigh-Taylor instability initialization

- Let us study the Rayleigh-Taylor instability.
- In this case, we have two phases with different physical properties (one phase is heavier).
- To onset this instability, we need to perturbate somehow the interface between the two phases.
- We will use **codeStream** to initialize the two phases.
- For simplicity, we will only show the **code** section of the input files.
- The entries **codeInclude**, **codeOptions**, and **codeLibs**, are the same most of the times.



Solution initialization using codeStream

- The **code** section of the **codeStream** IC in the body of the file *alpha.phase1* is as follows,

```
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];

        if (y >= -0.05*cos(2*constant::mathematical::pi*x))
        {
            alpha[i] = 1.;
        }
    }

    alpha.writeEntry("", os);
#};
```

Access internal mesh information


Initialize scalar field to zero

Access cell centers coordinates

Assign value to alpha

Write output to input dictionary

$y = -0.05 \times \cos(2\pi x)$



- For simplicity, we only show the **code** section.
- The rest of the body of the **codeStream** IC is a template.

Solution initialization using codeStream

Rayleigh-Taylor instability initialization

- This case is ready to run, the input files are located in the directory `$PTOFC/101programming/codeStream_INIT/rayleigh_taylor`
- To run the case, type in the terminal,
 1. `$> cd $PTOFC/101programming/codeStream_INIT/rayleigh_taylor`
 2. `$> foamCleanTutorials`
 3. `$> blockMesh`
 4. `$> interFoam | tee log`
 5. `$> paraFoam`
- FYI, you can run in parallel with no problem.

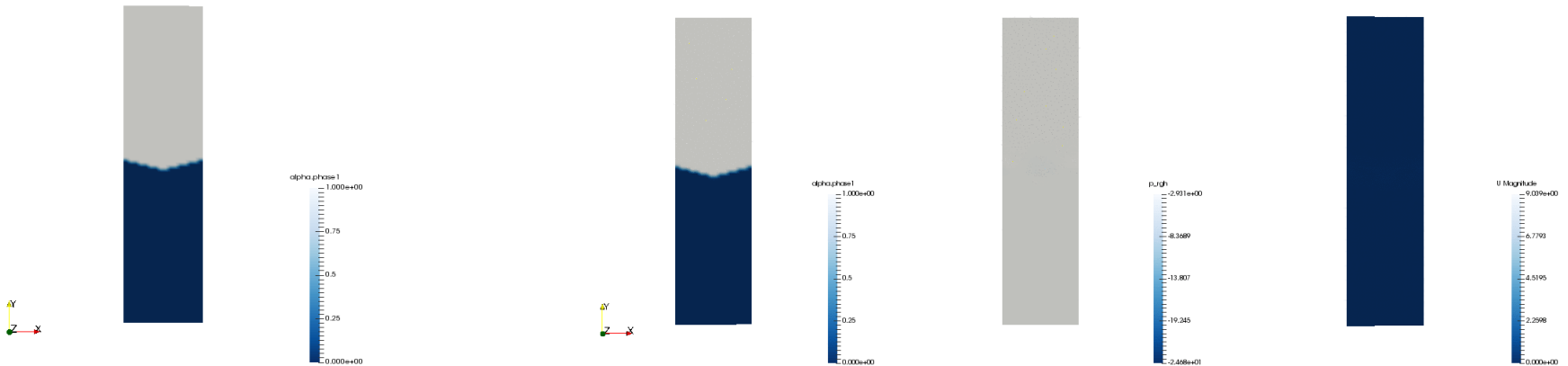
Solution initialization using codeStream

Rayleigh-Taylor instability initialization

- If everything went fine, you should get something like this



Time: 0.050000



Initial conditions

Visualization of volume fraction, static pressure and velocity magnitude

www.wolfdynamics.com/wiki/BCIC/rayleigh_taylor_ins1.gif