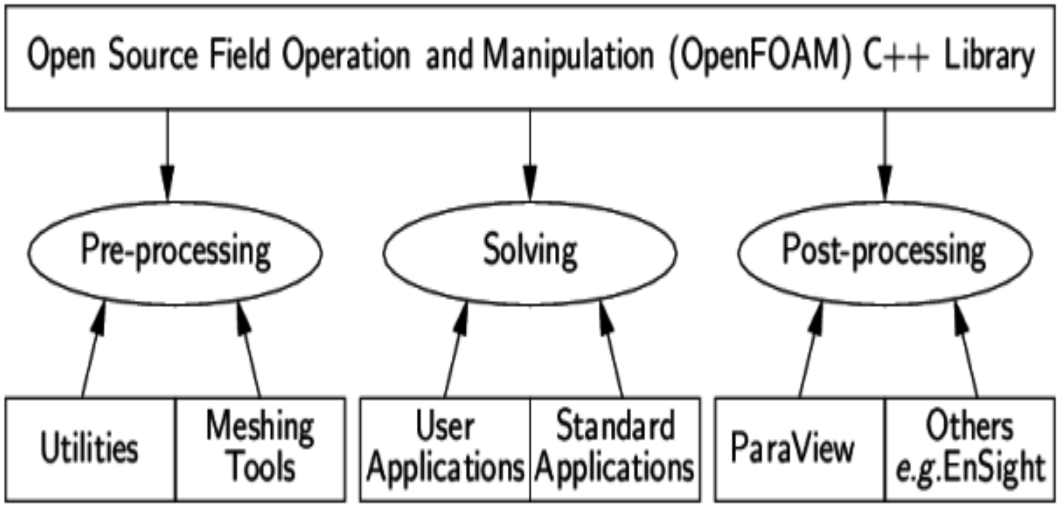
OpenFOAM

- Open source CFD toolbox, which supplies preconfigured solvers, utilities and libraries.

- Flexible set of efficient C++ modules---object-oriented.

- Use Finite-Volume Method (FVM) to solve systems of PDEs ascribed on any 3D unstructured mesh of polyhedral cells.

- Good parallelization.



Overview of OpenFOAM structures

- Resourceful community (CFD forum, *http://www.cfd-online.com/Forums/openfoam/*) contribution (user-defined libraries).

- interFoam, which is a solver for 2 incompressible fluids with interface tracking, is used in the present study.
  Tutorials: http://cfd.direct/openfoam/user-guide/dambreak/

0

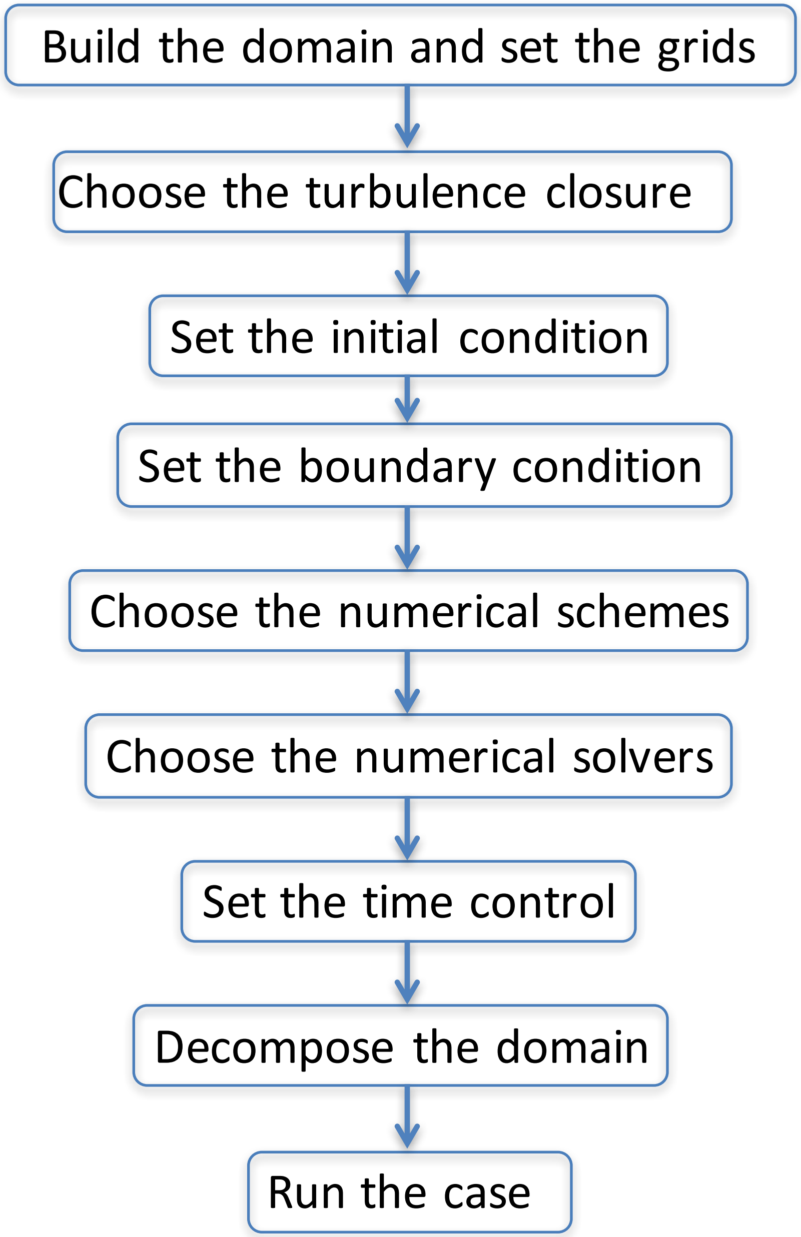
where boundary conditions are defined for *alpha.water, B, k, nuSgs, p_rgh, U*

constant

- Mesh building -- *blockMeshDict*
- Turbulence closure – *turbulenceProperties, LESProperties*

system

- Initial condition -- *setFieldslDict*
- Computational time control -- *controlDict*
- Numerical schemes -- *fvSchemes*
- Numerical solvers -- *fvSolution*
- Domain decomposition -- *decomposeParDict*



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

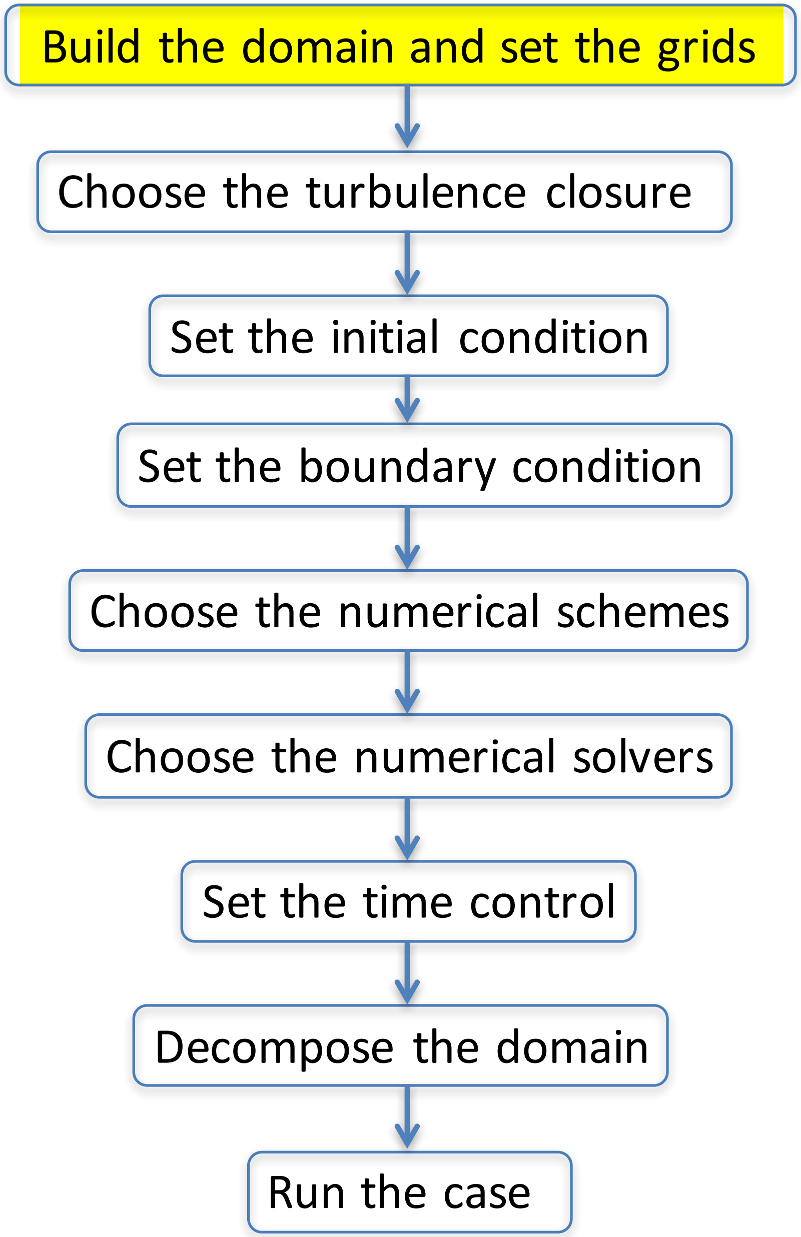
Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

Run the case



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

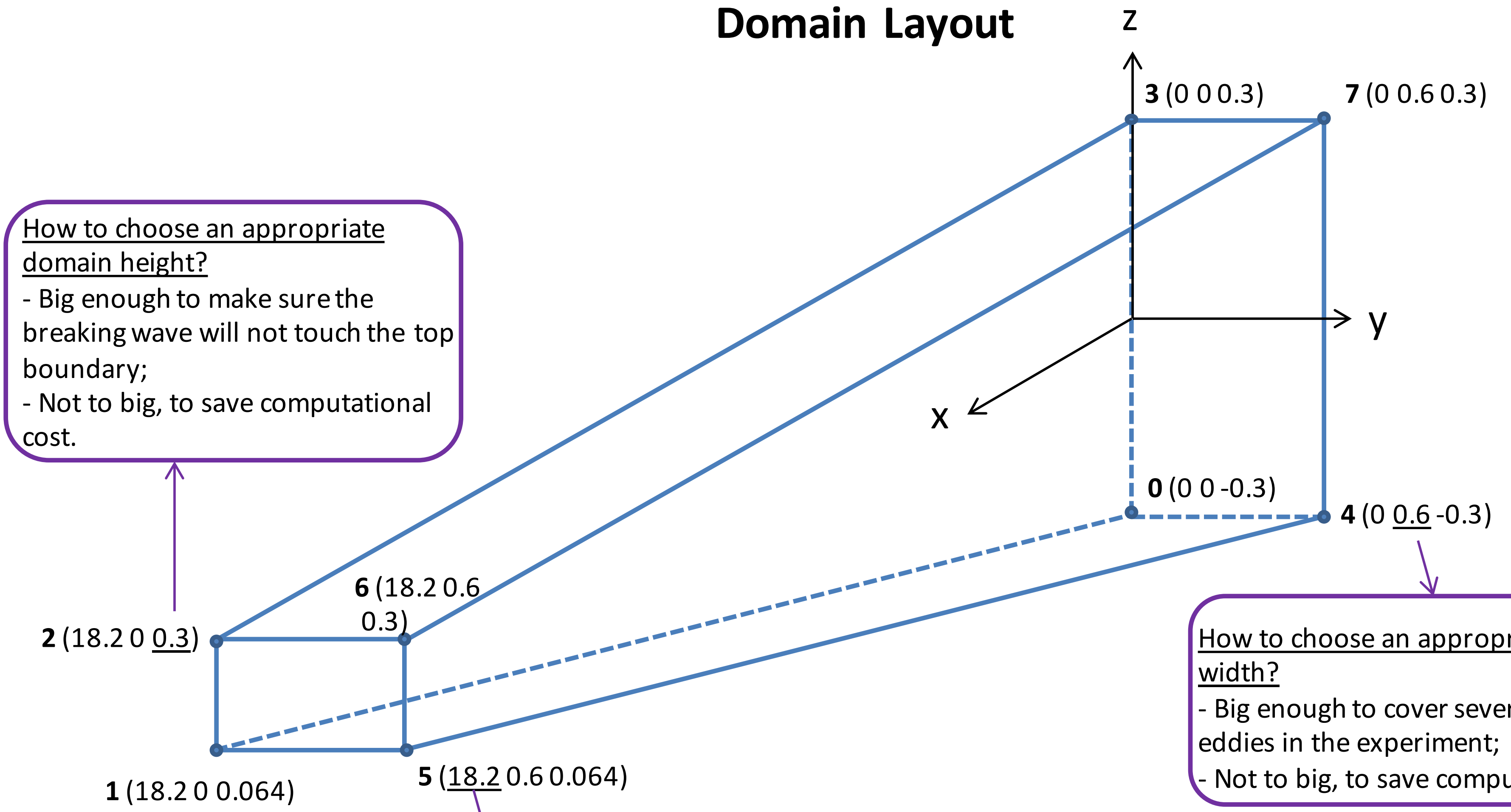
Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

Domain Layout



How to choose an appropriate domain height?

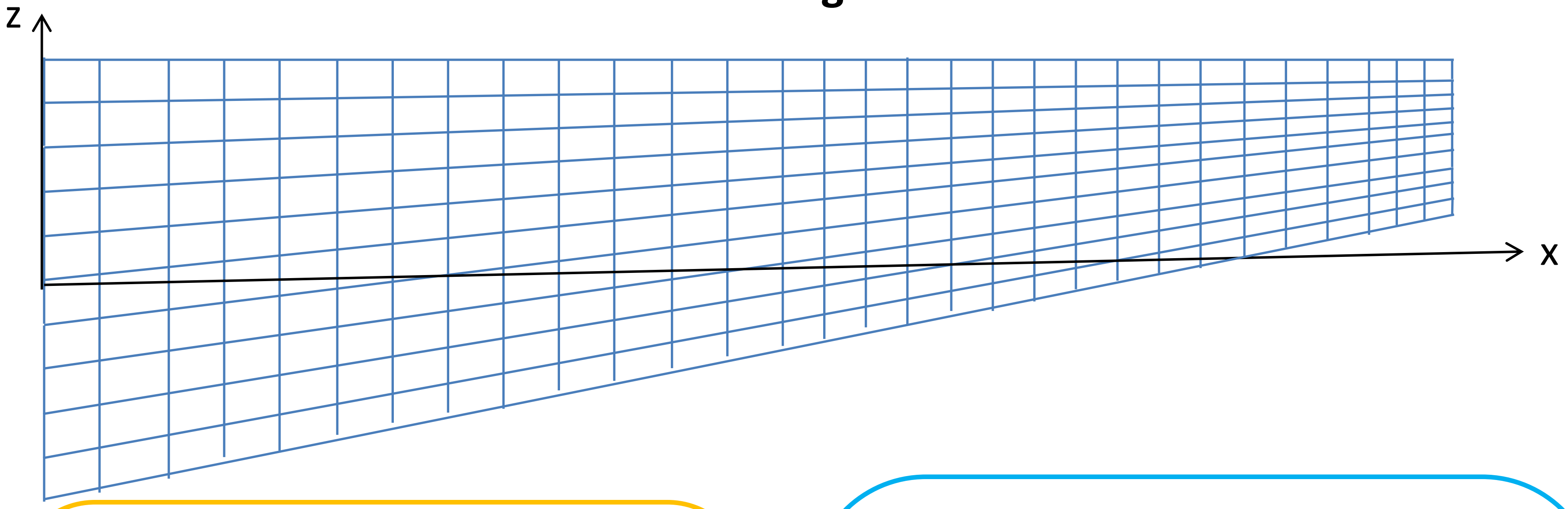
- Big enough to make sure the breaking wave will not touch the top boundary;
- Not to big, to save computational cost.

How to choose an appropriate domain width?

- Big enough to cover several largest eddies in the experiment;
- Not to big, to save computational cost.

In this case, the domain length is set to be smaller than that in the experiment [Ting 2006, 2008] but long enough to cover the initial shoreline and swash zone

Build the grids



Nz = 80.

How to determine the number of grids in the vertical direction?

-- Determined by the number of grids needed to solve the wave.

E.g., in this case, H_0=0.22 m. Using 30 grids to solve the wave ➡

Δz = 7.5 mm at the left boundary ➡

Nz = Domain height / Δz = 80

Ny = 80.

How to determine the number of grids in the spanwise direction?

Make sure the ratios of $\Delta y/\Delta z$ and $\Delta x/\Delta y$ are not too big. E.g., in this case, Δy = 7.5 mm

Nx = 2427.

How to determine the number of grids in the streamwise direction?

-- Make sure the ratio of $\Delta x/\Delta z$ is not too big (<5).

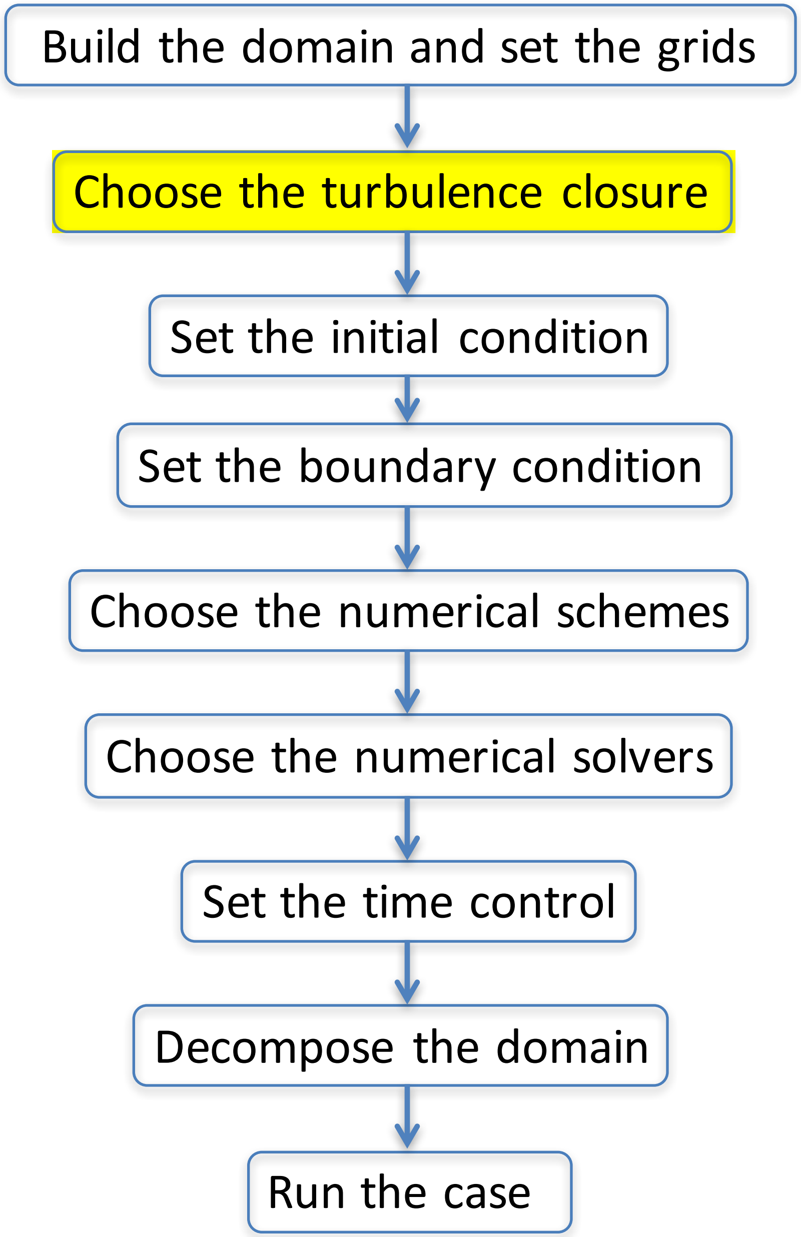
E.g., in this case, $\Delta z_{\max}$=7.5 mm at the left boundary; $\Delta z_{\min}$=3 mm at the right boundary.

To make $\Delta x/\Delta z$ < 5, choose $\Delta x_{\max}$=11.5 mm ($\Delta x/\Delta z$ = 3.8) at the left boundary

and

$\Delta x_{\min}$=4.6 mm ($\Delta x/\Delta z$ = 1.5) at the right boundary.

Δx is shrinking from the left to the right. (So is Δz)



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

Choose the turbulence closure

Large-eddy simulation in this simulation

Filter is defined as $\Delta = \sqrt[3]{V_{cell}}$

Dynamic Smagorinsky closure is used, and an improved version of dynamic Smagorinsky closure developed by Alberto Passalacqua is adopted.

How to install the improved dynamic Smagorinsky closure in OpenFOAM?

1. Download the source code using git:

```
git clone git://github.com/AlbertoPa/dynamicSmagorinsky.git
```

2. Enter the directory where the source code has been extracted, and compile it by typing:

```
wmake libso
```

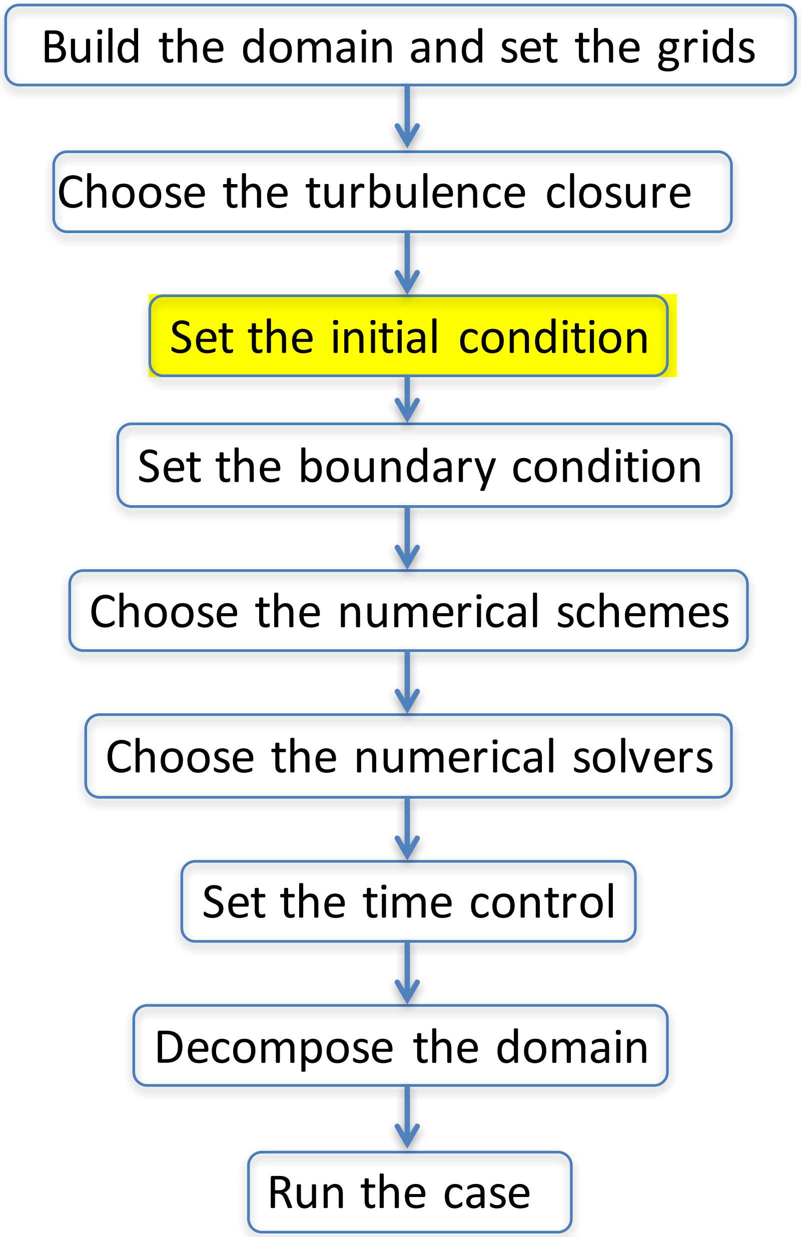
3. Add the following line to the controlDict of your case: libs ("libOpenFOAM.so" "libdynamicSmagorinskyModel.so") ;

4. Specify LESModel dynamicSmagorinsky; delta cubeRootVol; in LESModel. 5. Add the subdictionary

```
dynamicSmagorinskyCoeffs
{ filter simple;
  ce 1.048;
}
```

to LESModels.

https://github.com/AlbertoPa/dynamicSmagorinsky



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

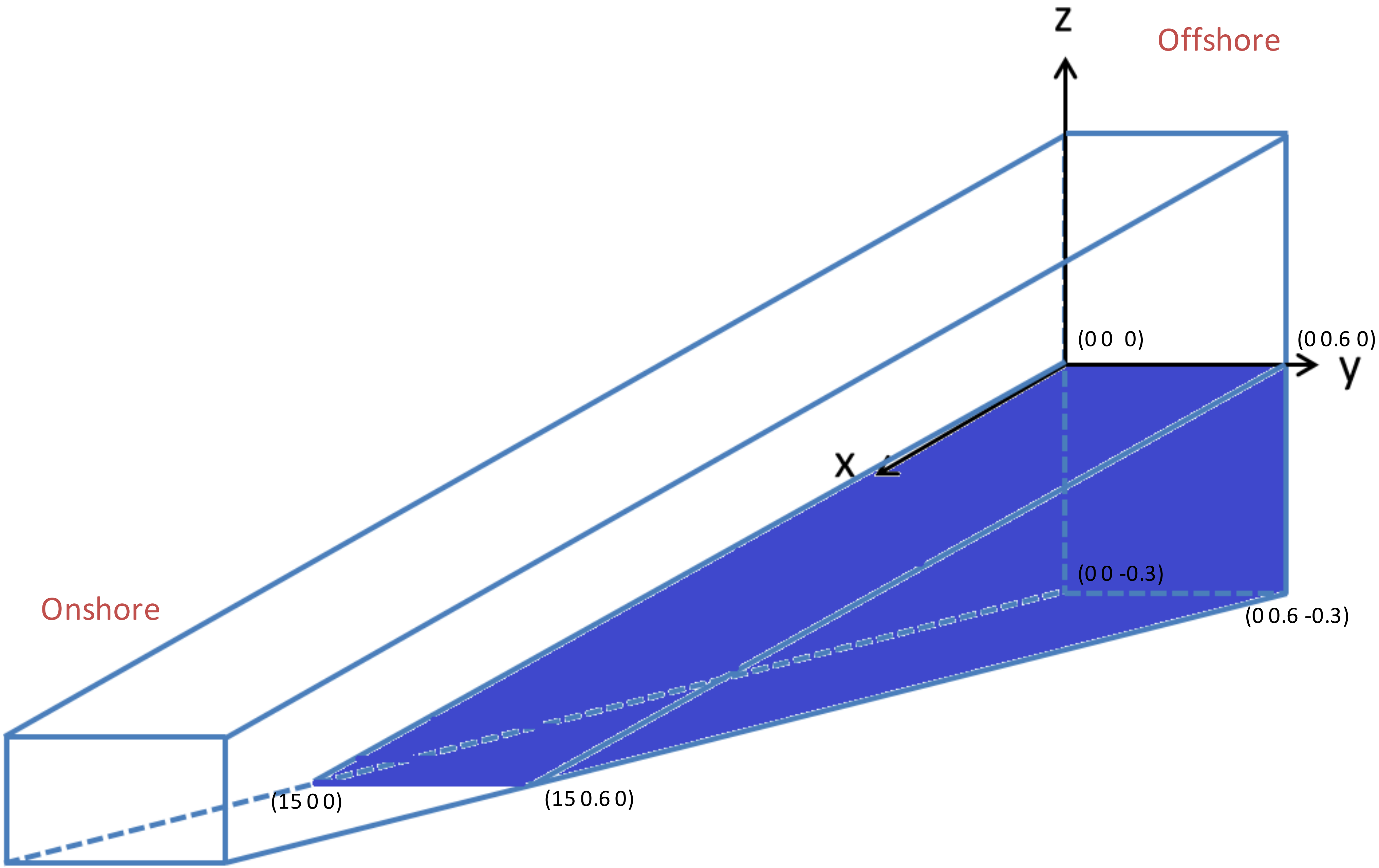
Choose the numerical solvers

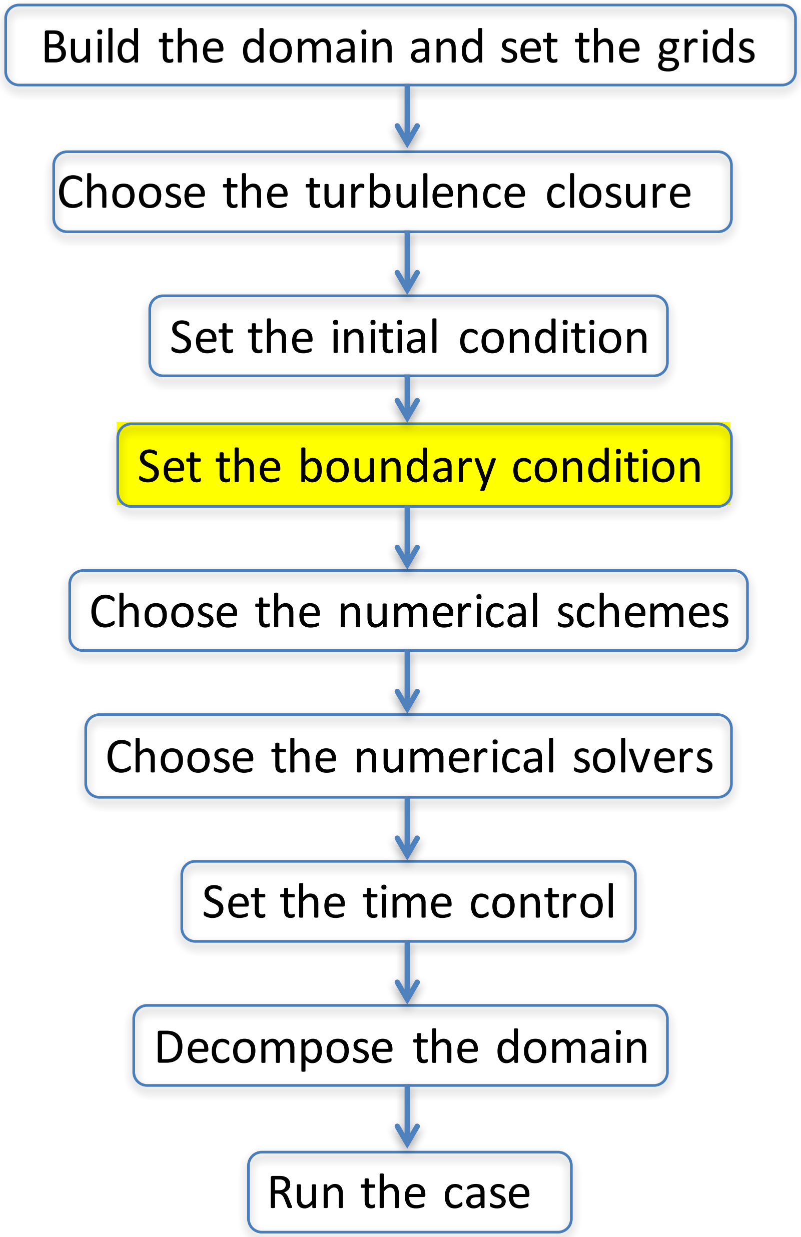
Set the time control

Decompose the domain

Run the case

Set the initial condition





Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

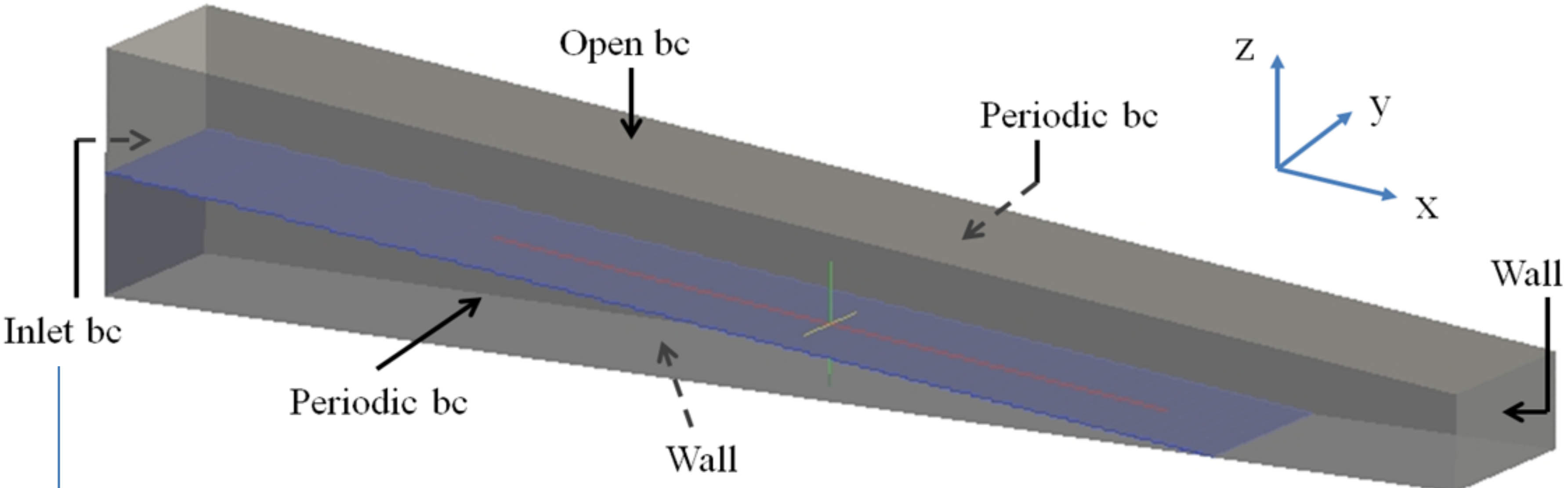
Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

Define the boundaries



(Wall function is used)

Sends in the target solitary wave through *groovyBC*

What is *groovyBC*?

-- A library that can be used to generate arbitrary boundary conditions based on expressions. It is included in the swak4Foam library package.

Link: https://openfoamwiki.net/index.php/Contrib/swak4Foam

Install groovyBC

1. Download swak4Foam library package from

```
svn checkout svn://svn.code.sf.net/p/openfoam-extend/svn/trunk/Breeder_2.0/libraries/swak4Foam/ swak4Foam_2.x
```

2. In the directory of the sources, type

```
wmake all
```

Use groovyBC to send in solitary wave

$$\alpha_1 = \begin{cases} 1, z \le \boxed{\dfrac{H}{\cosh^2\left(atp\left(-ct+xs\right)\right)}} \\ 0, z > \dfrac{H}{\cosh^2\left(atp\left(-ct+xs\right)\right)} \end{cases}$$

Expression of the theoretical surface elevation in Lee et al. [1982]

$$h = 0.3m \quad H = 0.22m \quad f_s = 2.644 \quad g = \left(0,0,-9.81\right) \quad c = \sqrt{gh\left(1+H/h\right)} \quad xs = \frac{hf_s}{\sqrt{H/h}} \quad atp = \sqrt{\frac{0.75H}{h^3}}$$

$$u = \begin{cases} \boxed{\dfrac{\sqrt{gh}H}{\cosh^2\left[atp\left(-ct+xs\right)\right]h}\left[1-\dfrac{0.25H}{\cosh^2\left[atp\left(-ct+xs\right)\right]h}\right]}, z \le \dfrac{H}{\cosh^2\left(atp\left(-ct+xs\right)\right)} \\ 0, z > \dfrac{H}{\cosh^2\left(atp\left(-ct+xs\right)\right)} \end{cases}$$

Expression of the theoretical velocity in Lee et al. [1982]

$$v = 0$$

$$w = \begin{cases} \boxed{\dfrac{-\sqrt{gh}z}{h}\left[1-\dfrac{0.5Hdex}{\cosh^2\left[atp\left(-ct+xs\right)\right]h}\right]}, z \le \dfrac{H}{\cosh^2\left(atp\left(-ct+xs\right)\right)} \\ 0, z > \dfrac{H}{\cosh^2\left(atp\left(-ct+xs\right)\right)} \end{cases}$$

Specify the boundary conditions

alpha.water: α_1 (percentage of water in each cell) in the VOF equation

B: subgrid-scale tensor in LES. $B = 2/3kI + B_{eff}$ is the unit tensor; I is the deviatoric part of the subgrid-scale tensor and is parameterized by subgrid closure

k: subgrid-scale kinetic energy in LES. $k = c_I \Delta^2 \|D\|^2$, $c_I \approx 0.2$, $\|D\|$ is the rate of strain

nuSgs: ν_{sgs}, sub-grid scale viscosity in LES

p_rgh: dynamic pressure

U: velocity

Specify the boundary conditions

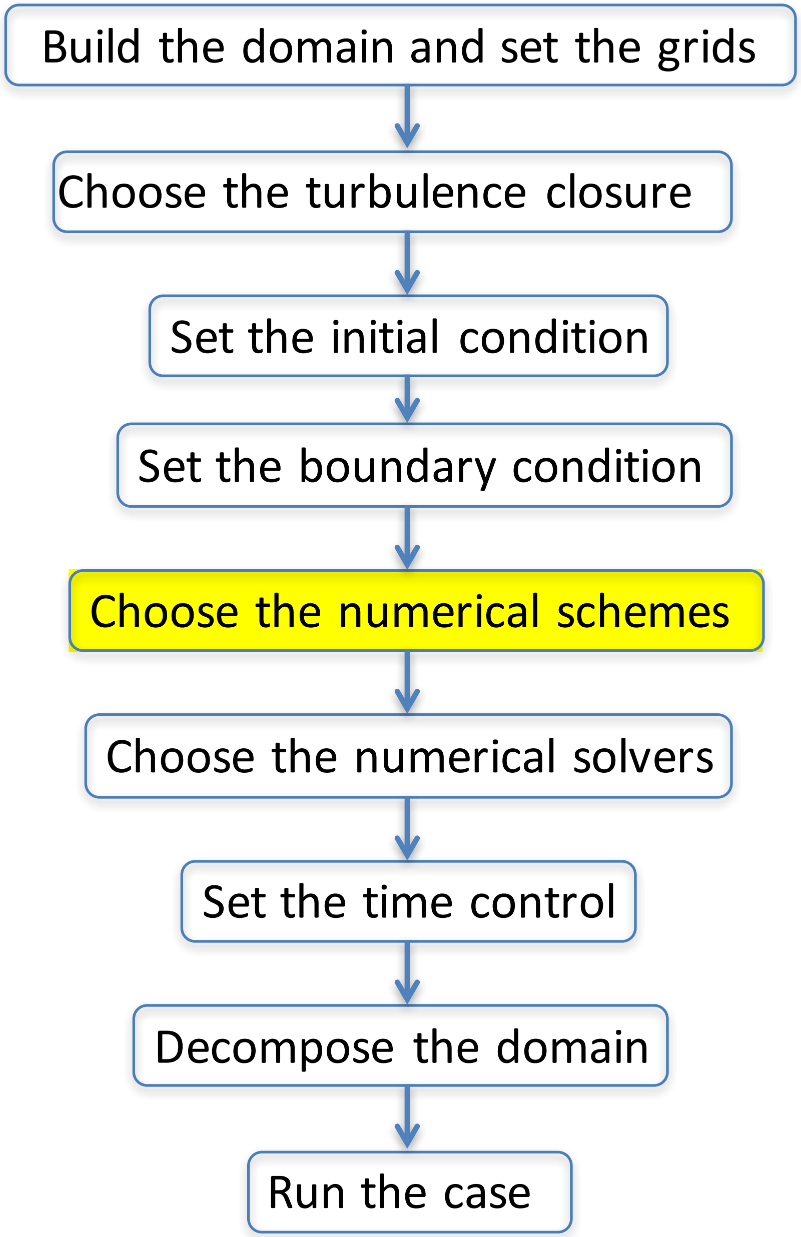
zeroGradient: normal gradient is zero

cyclic: periodic boundary condition

inletOutlet: $\approx$ zeroGradient. But switch to fixedValue (using "inletValue") if the velocity just outside the boundary is flowing into the domain

totalPressure: Total pressure $p_0 = p + 1/2\,\rho\left|U\right|^2$ is fixed; when U changes, p will be adjusted accordingly

pressureInletOutletVelocity: = pressureInletVelocity + inletOut
pressureInletVelocity: When p is known at the inlet, U is evaluated from the flux normal to the path.



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

Numerical schemes

ddtSchemes: first time derivative $(\partial/\partial t)$. “CrankNicholson 1” is the pure 2nd-order Crank-Nicolson scheme

gradSchemes: Gradient ∇. “Gauss linear” means Gauss’ theorem is used when transforming integral over volume into integral over surface; “linear” means central difference scheme (CDS)

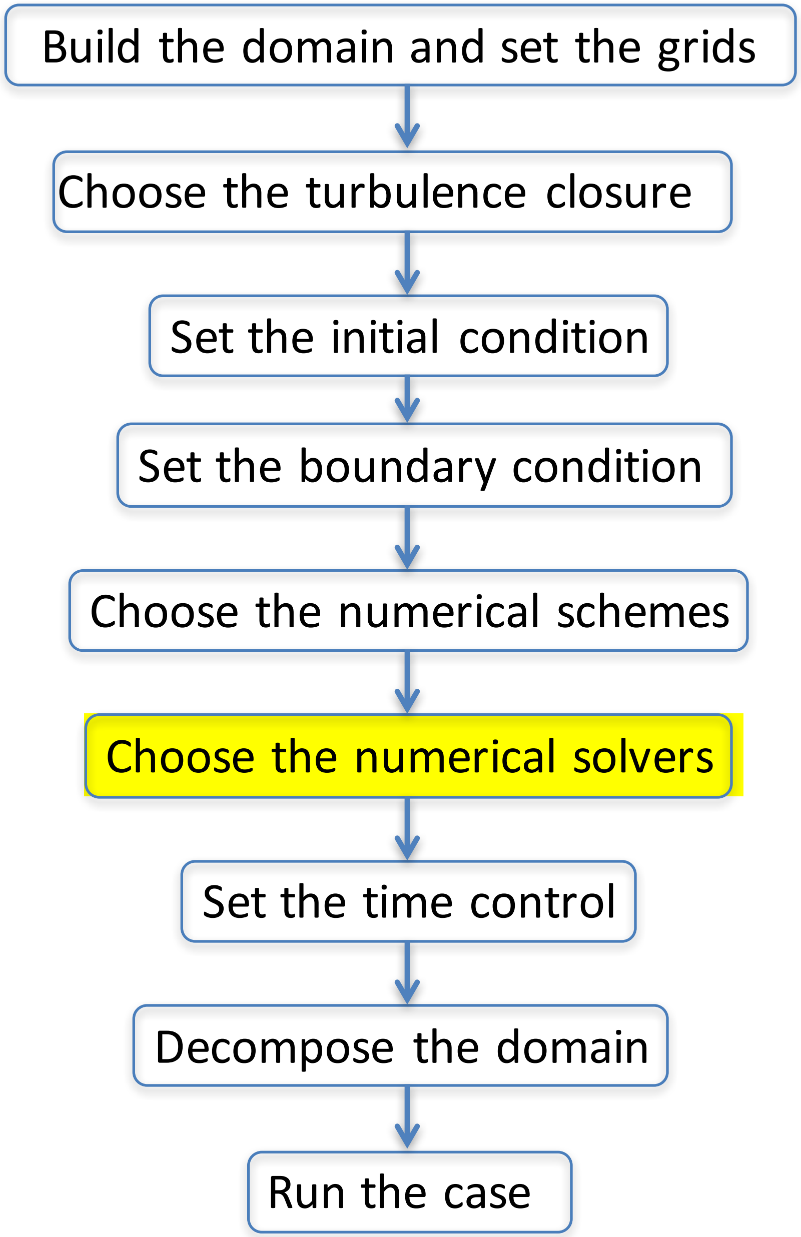
divSchemes: Divergent $\nabla\cdot$. “Gauss limitedLinearV 1” and “Gauss vanLeer” are both TVD schemes with different limiters. “Gauss interfaceCompression” is used for the interface compression term

laplacianSchemes: Laplacian ∇^2. “Gauss linear corrected” is CDS with some correction terms

interpolationSchemes: numerical scheme for the evaluation of face values from the cell center values

snGradSchemes: component of gradient normal to a cell face

fluxRequired: fields which require the generation of a flux



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

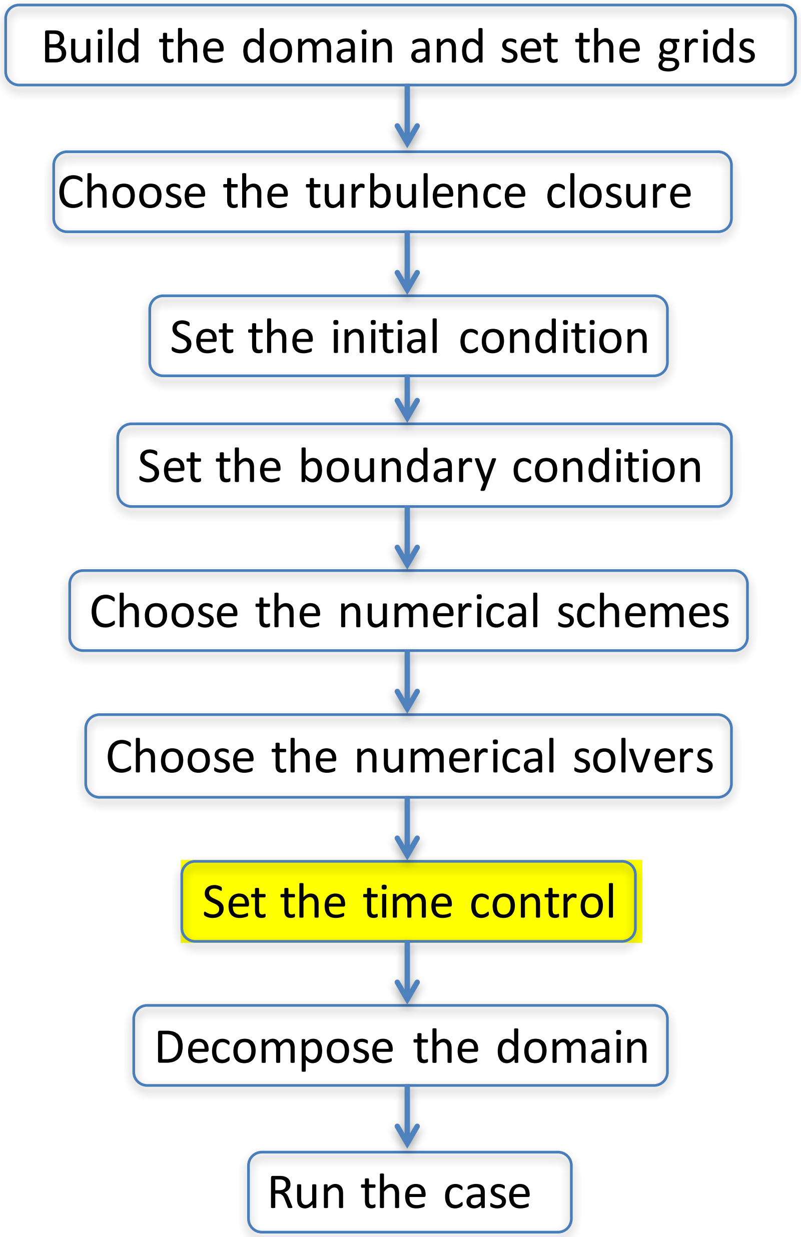
Run the case

Numerical solvers

Solves the Pressure Poisson Equation

Set the solvers for *p_rgh* and *U*

PIMPLE = SIMPLE + PISO



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

Set the time control

startFrom

stopAt

deltaT

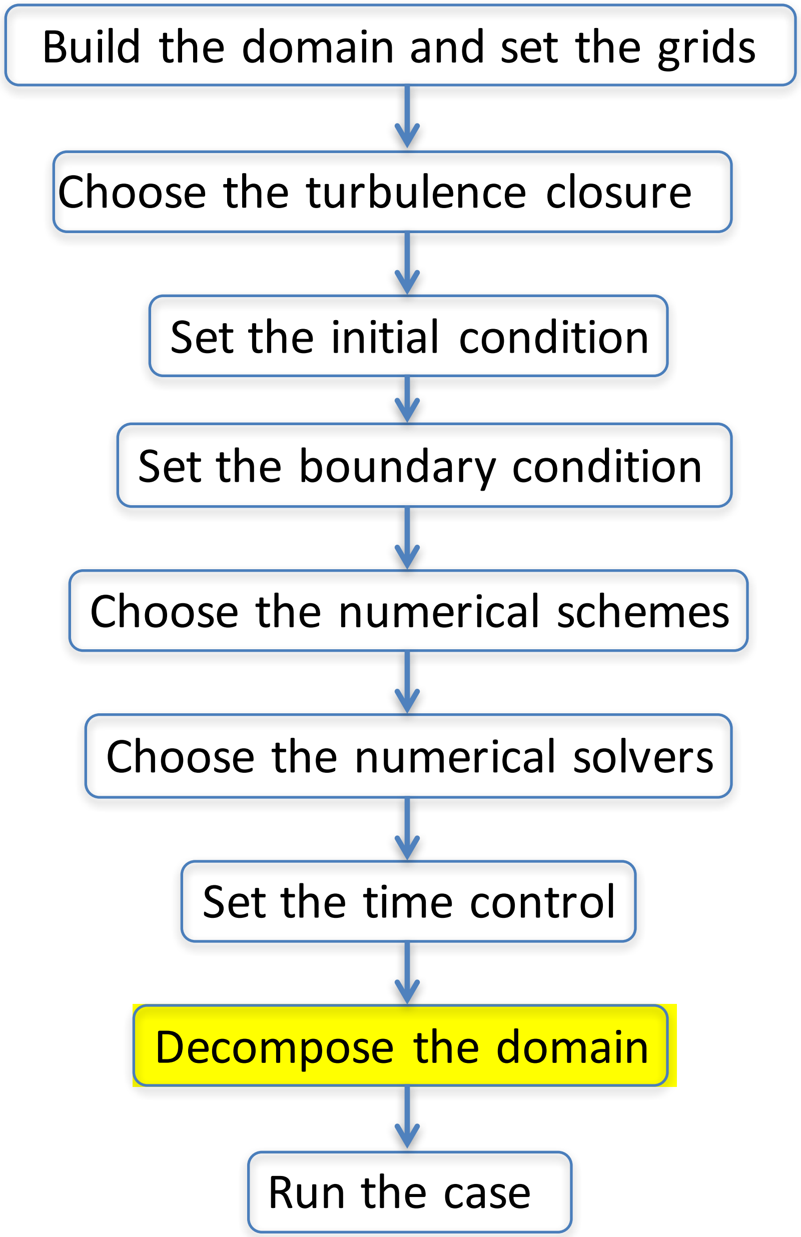
writeControl

adjustTimeStep

maxCo

maxAlphaCo

maxDeltaT



Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

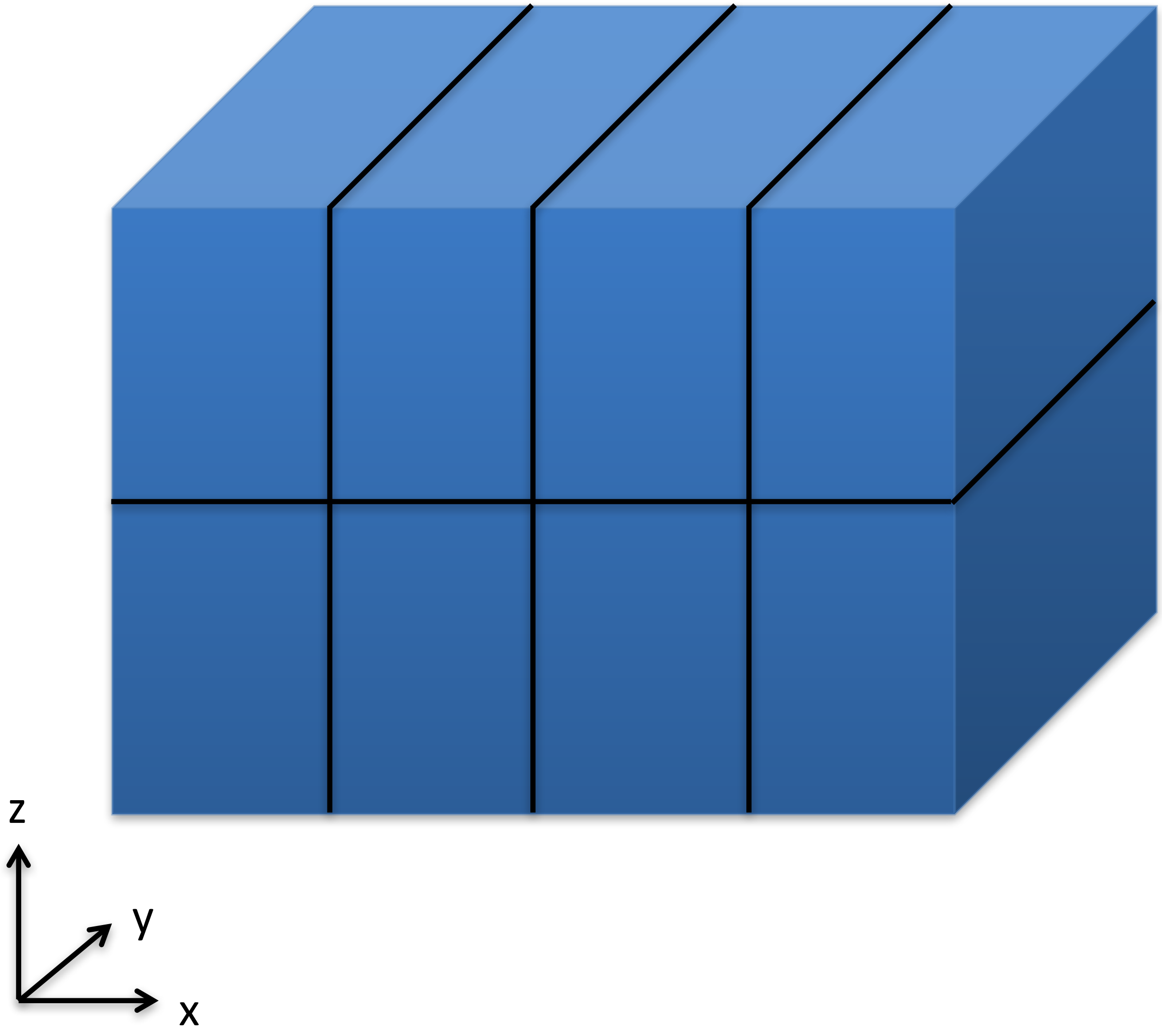
Decompose the domain

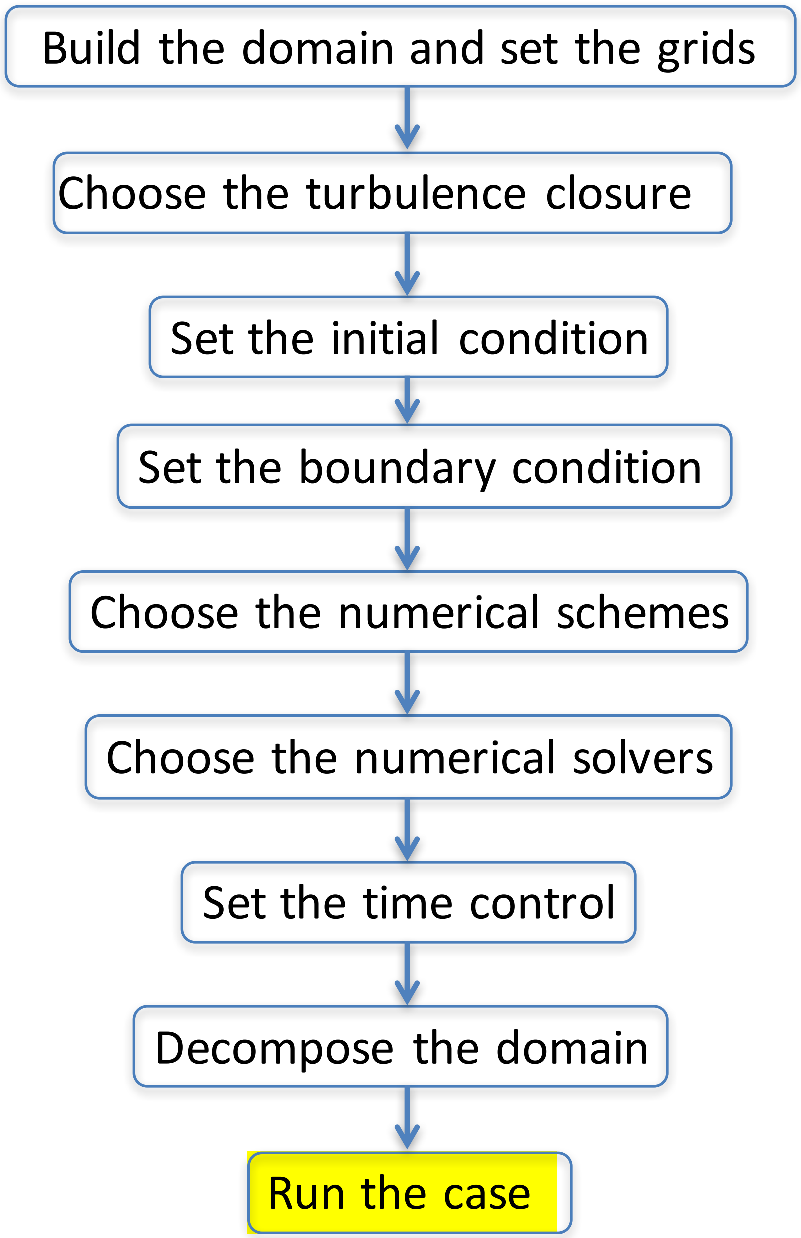
"decomposeParDict"

```
numberOfSubdomains 8;

simpleCoeffs
{
   n             (4 1 2);
   …
}
```

Type
decomposePar





Build the domain and set the grids

Choose the turbulence closure

Set the initial condition

Set the boundary condition

Choose the numerical schemes

Choose the numerical solvers

Set the time control

Decompose the domain

Run the case

Run the case

Type

interFoam