Tutorial: BwUniCluster 3.0/HoreKa
Hot Radiation Room (parallel)

In this tutorial we will learn about the simulation of radiative heat transfer in OpenFOAM on multiple cores by submitting a parallel job. We will do it using the example hotRadiationRoom from the series of built-in OpenFOAM tutorials.

1. Tutorial Case

First, we should copy the tutorial case from the FOAM_TUTORIALS directory and paste it into our workspace (denoted by a point):

```
$>   cd <workdir>
$>   module purge
$>   module load cae/openfoam/v2112
$>   source $FOAM_INIT
$>   cp -R $FOAM_TUTORIALS/heatTransfer/buoyantSimpleFoam/hotRadiationRoom/ .
$>   cp $FOAM_TUTORIALS/multiphase/compressibleInterFoam/laminar/depthCharge3D/system/decomposeParDict hotRadiationRoom/system
```

Note: The last command copies the dictionary used for domain decomposition from another tutorial. We will modify it for our example.

2. Creating the Mesh

The mesh in the hotRadiationRoom tutorial represents a 10m x 6m x 2m room equipped with a heat source in the corner of its floor.

In order to create the mesh using data from the blockMeshDict-file, we will use the following command:

```
$>   cd hotRadiationRoom
$>   blockMesh
```

Or, better - as it allows to keep the statistics of the grid in a file:

```
$>   blockMesh > screen_output_blockmesh.dat
```

Apart from describing the basic structure of the mesh, the blockMeshDict file also defines boundaries, known as patches, for which later on special boundary conditions will be set. In our case these areas are: floor, ceiling, fixedWalls and box. To view the file, you can use the command:

```
$>   more system/blockMeshDict
```

The mesh is shown in Fig. 1. Figures 2, 3 and 4 show the different boundary patches.

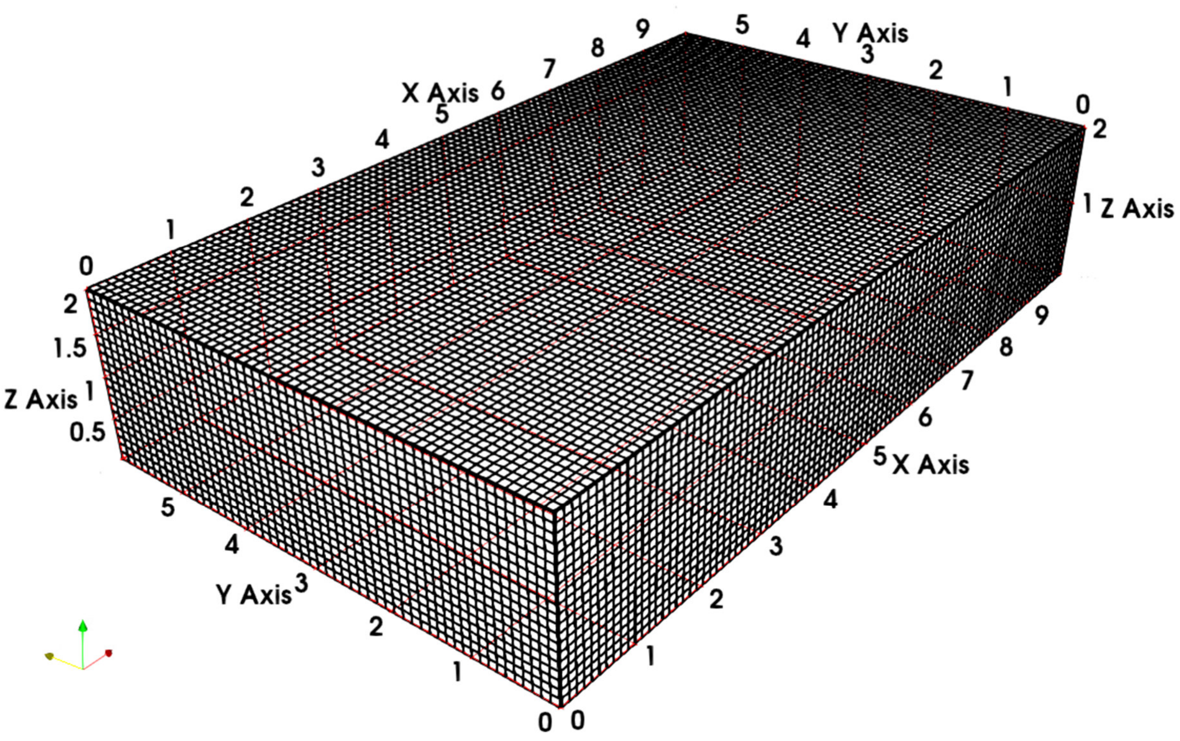


Fig.1 The hotRadiationRoom mesh created using blockMesh

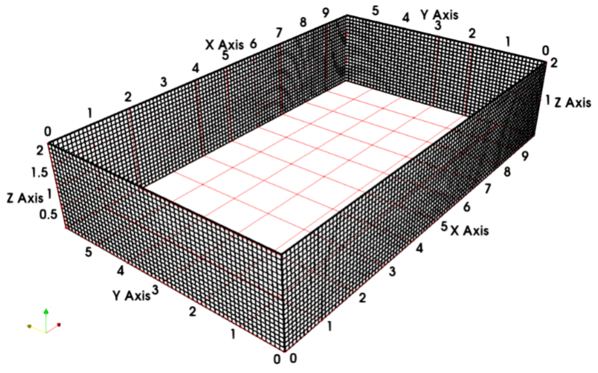


Fig.2 Boundary patch "fixedWalls"

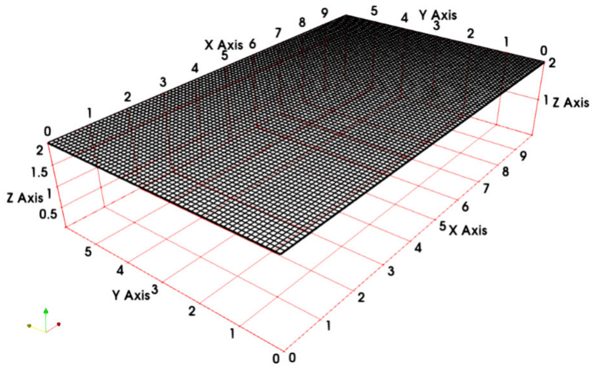


Fig.3 Boundary patch "ceiling"

Fig.4 Boundary patch "floor"

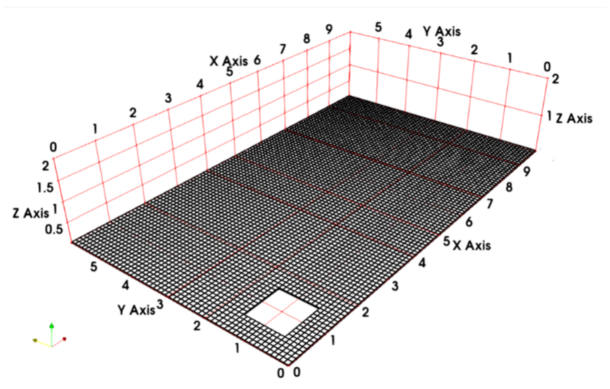
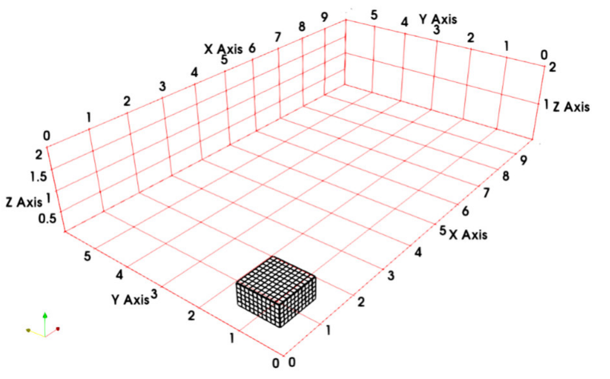


Fig.4 Boundary patch "box"



In this tutorial there will be a heat source with the temperature of 500 K in the box patch:

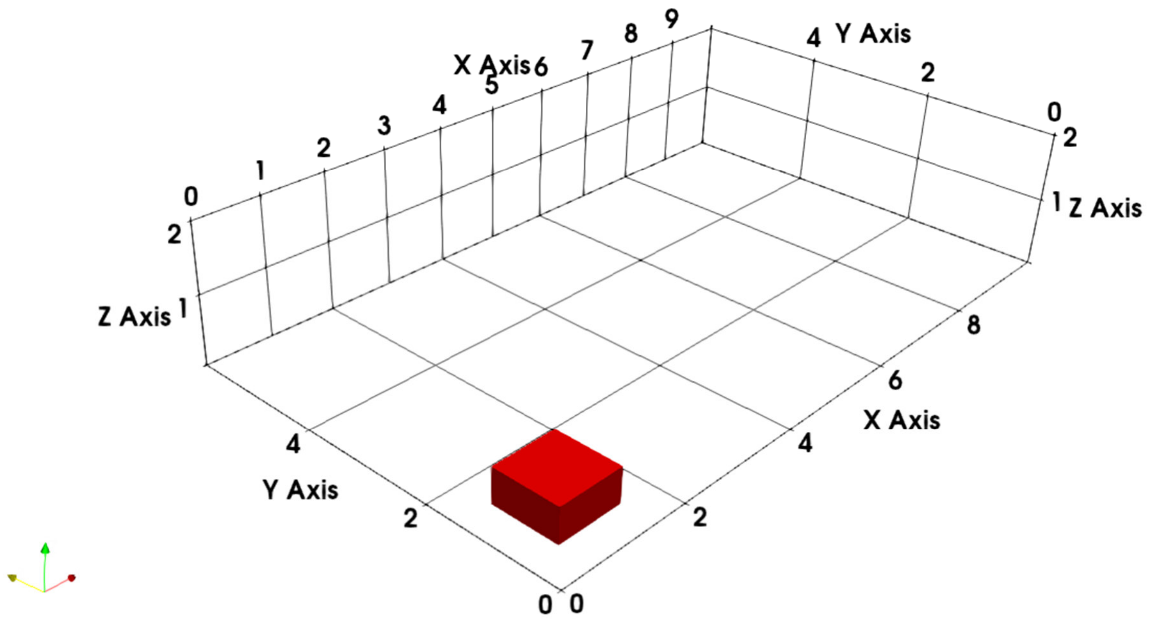


Fig.5 The heat source in the box patch

Boundary conditions are set in the files from the directory „0” which means the starting time (or iteration). To create this directory, we copy it from directory 0.orig:

```
$>   cp -R 0.orig  0
```

The value 500K for this temperature is set towrds the end of the file „0/T”. In the same file the initial wall temperature of 300K is also set.

Since we will be conducting a parallel computation, we also need to decompose our mesh into subdomains. We want to conduct the parallel simulation using 8 cores, therefore our decomposeParDict-file should cut the room-mesh in 8 subdomains and will finally look like this (we need to modify the existing file in directory „system”):

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 8;
method        hierarchical;
coeffs
{
   n          (2 2 2);
}
// ********************************************************************************* //
```

Decomposition of the domain is carried out with the following commnad:

```
$>   decomposePar  >  screen_output_decomposepar.dat
```

This creates a directory for each of the eight processors (cores). Each core has its own grid, initial and boundary conditions in the corresponding files.

Now it is time to submit the job for calculation. Assuming we are conducting the simulation on a multiple nodes the job-submit-file (name e.g. job_submit_dev_ parallel_uc3.sh in directory hotRadiationRoom) should look like this:

```
//////////////////////////////////////////////////////////////////////////////////
#!/bin/bash                          ←————————   header of the file

#SBATCH --partition dev_???????      ←————————   job will be submitted to queue dev_???
#SBATCH --nodes=1                    ←————————   one node will be used
#SBATCH --ntasks=8                   ←————————   ntasks means „the number of cores”  to use
#SBATCH --ntasks-per-node=4          ←————————   this is a bit redundand information
#SBATCH --time=00:30:00              ←————————   the job will run for maximum 30 minutes
#SBATCH --mem=8000mb                 ←————————   the job may use max. 8 Gb
#SBATCH --job-name=hotRadiationRoom      ←————————   this name is given by the user

module purge                             ⎫
module load  cae/openfoam/v2112          ⎬—  loading OpenFOAM on the execute core
source $FOAM_INIT                        ⎭
mpirun -n 8 buoyantSimpleFoam -parallel  ←————————   the name of the OpenFOAM-solver

//////////////////////////////////////////////////////////////////////////////
```

Note!: On HoreKa use for the name of the partition dev_cpuonly. On bwUniCluster3.0 use dev_cpu or dev_cpu_il. Names of partitions can change frequently, always visit the corresponding web-page or use the command "sinfo_t_idle".

We can then submit the job using the name of the job-file with the following command:

```
$>   sbatch job_submit_dev_ parallel_uc3.sh
```

After the simulation has been completed, we can combine the results from different cores (processor* -directories) into one directory using the command:

```
$>   reconstructPar
```

We can also continue without reconstructing. In that case we will be able to visualize results from different processors (decomposed case) as well as the reconstructed case in Paraview.

Note: Following our own recommendations - we did not use the tutorial file „Allrun".

3. **Results**

After the simulation has been completed we can download the results onto our personal computer and visualize them in Paraview. This is the recommended way for visualizing results with small- or medium-sized grids which is the most common case.

We can now examine the steady state of the heat transfer in the room which has been achieved after 935 iterations. The following images depict the central plane of the control room from the top-down view with the box patch located in the bottom left corner:

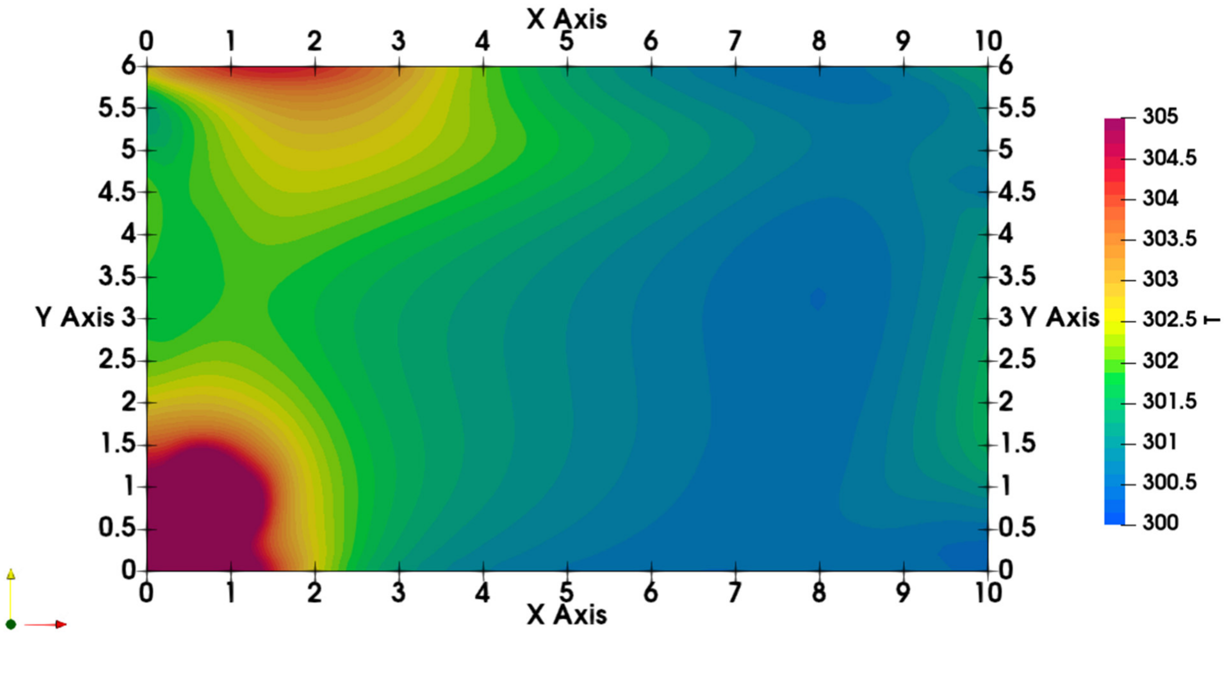


Fig.6 Temperature from the horizontal (top-down) view [K]

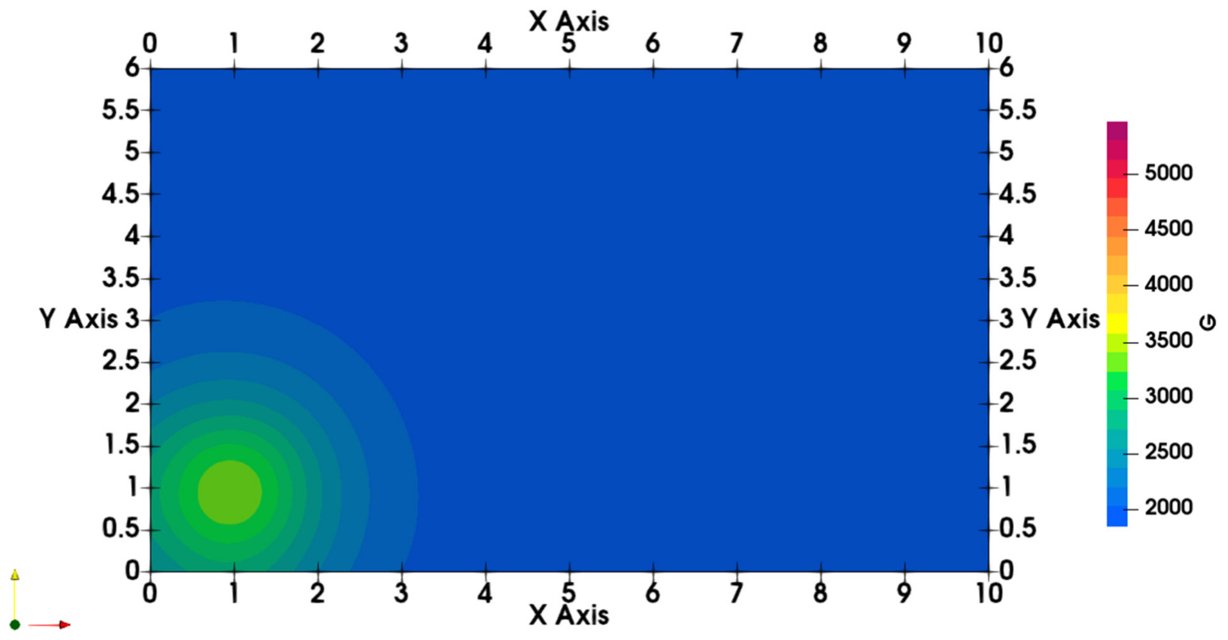


Fig.7 Radiation Intensity „G” from the horizontal (top-down) view [W/m^2]

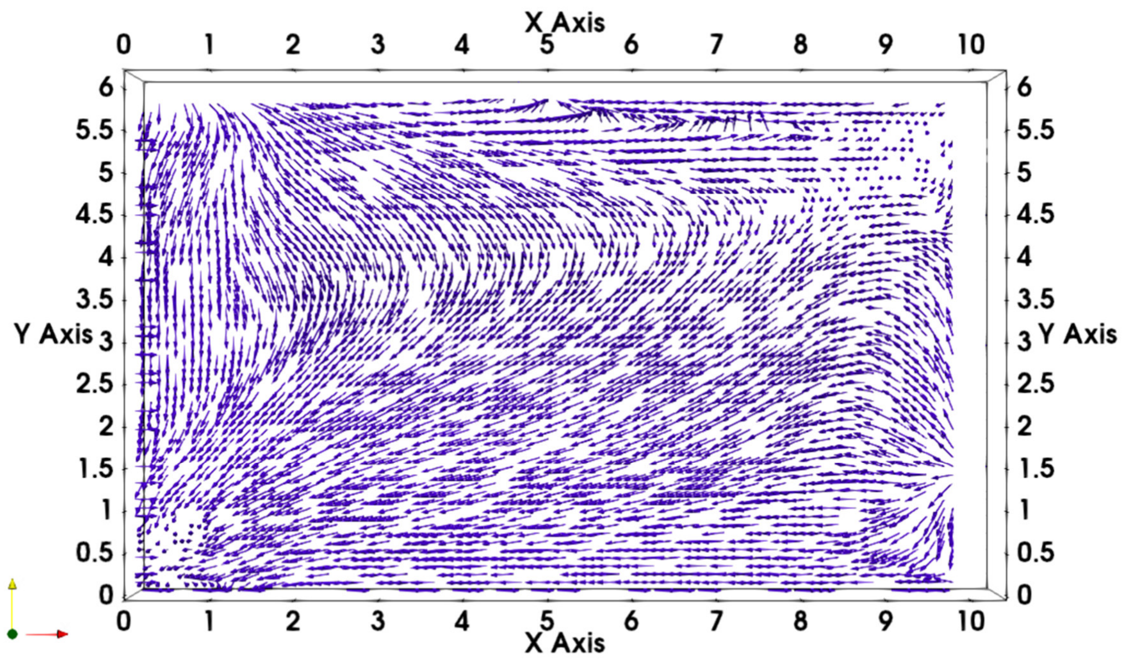


Fig.8 Velocity vectors from the top-down view [m/s]

As one could have expected, the heat transfer caused by radiation is noticeable only above the box patch due to absorption of the medium. The remaining heat transfer as well as the mass transfer take place due to convection.

Another place worth looking at is the Y-normal plane going through the box patch:

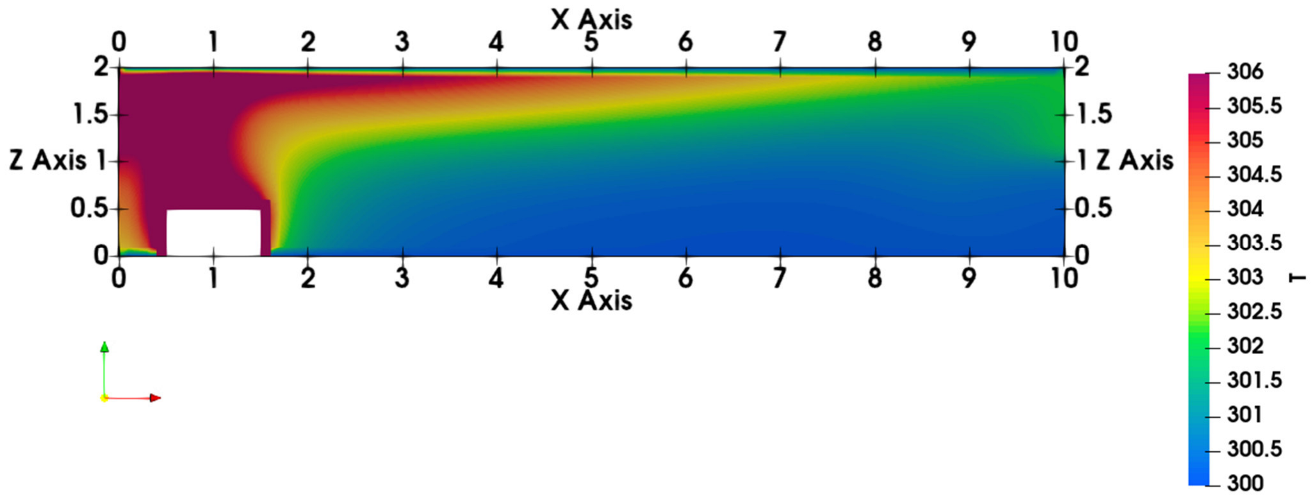


Fig.9 Temperature as viewed from the side [K]

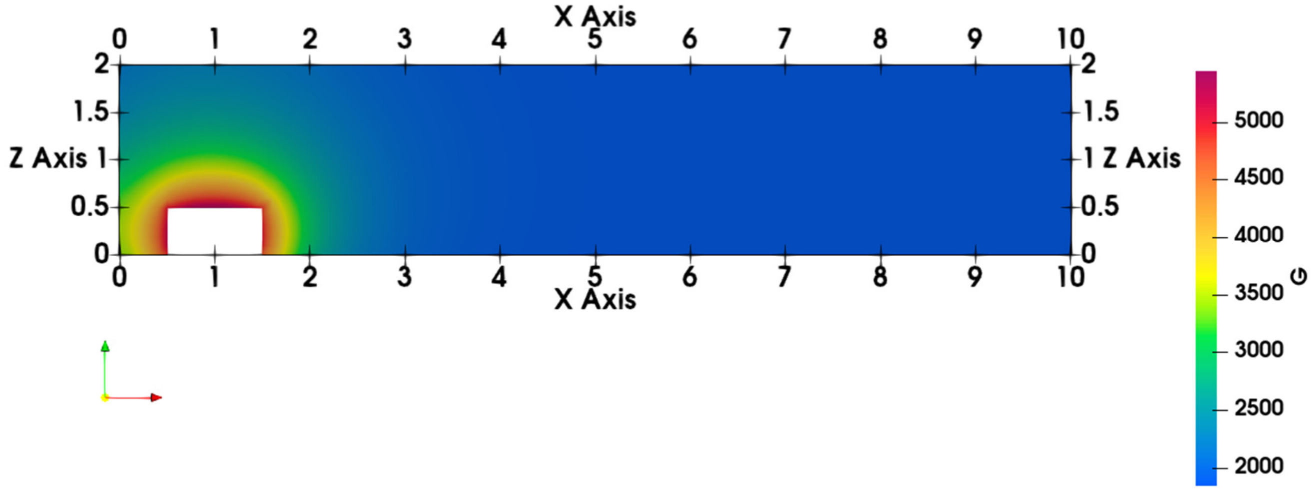


Fig.10 Radiation Intensity „G" as viewed from the side [W/m^3]

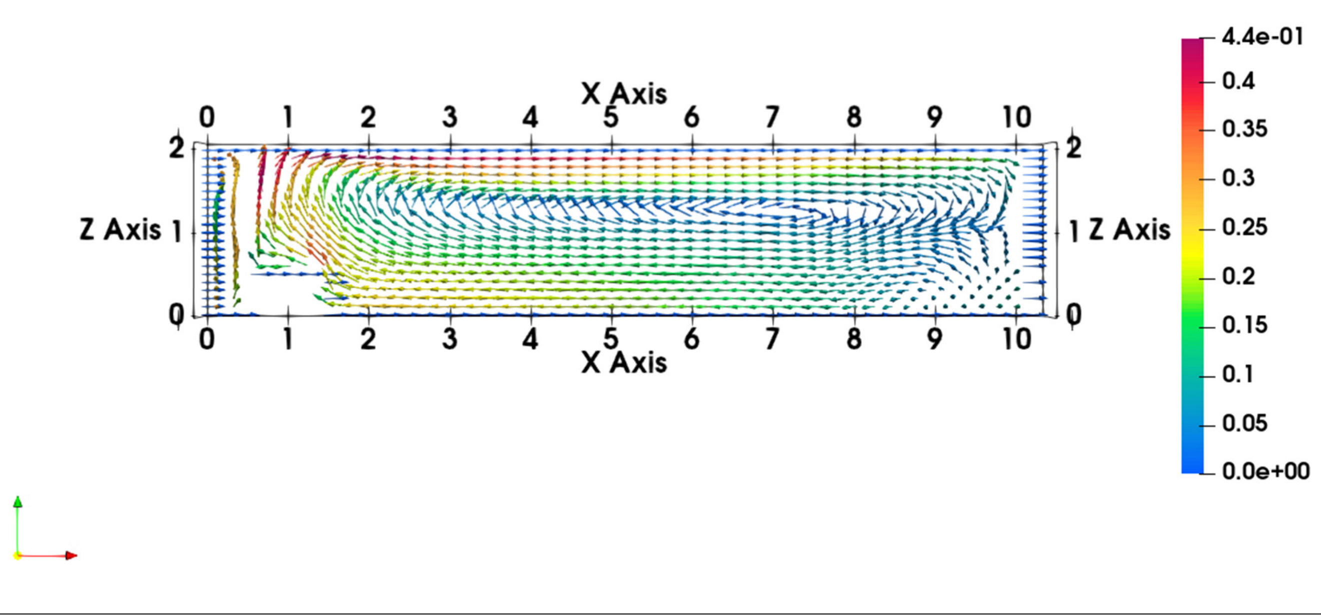


Fig.11 Velocity as viewed from the side [m/s]

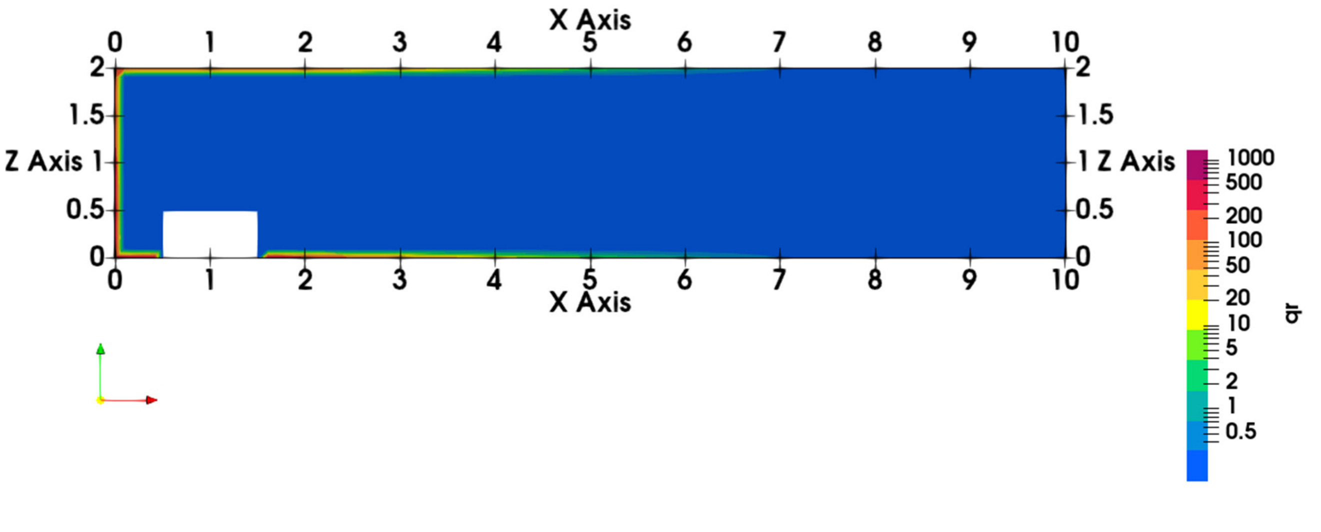


Fig.12 Radiative heat flux „qr” as viewed from the side [W/m^2]

As expected, the radiation coming from the box causes the air arround it to heat up. Then, due to convection, the hot air moves towards the ceiling where it cools down which drives the circulation in the room.

Legend

a	Absorption coefficient	[1/m]
alphat	Turbulent thermal diffusivity	[kg/(m.s)]
epsilon	Turbulent kinetic energy dissipation rate	[J/(kg.s)] or [m^2/s^3]
G	Incident radiation intensity	[W/m^2]
qr	Radiative heat flux	[W/m^2]
phi	Mass flow	[kg/s]
k	Turbulent kinetic energy	[J/kg] or [m^2/s^2]
nut	Turbulent viscosity	[m^2/s]

The names in the leftmost column are the variables in the 0 directory.