Cite as: Setareh, Milad: Acoustic streaming modeling. In Proceedings of CFD with OpenSource Software, 2016, Edited by Nilsson. H.,http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2016

CFD with OpenSource software

A course at Chalmers University of Technology
Taught by Håkan Nilsson

Acoustic streaming modeling

Developed for FOAM-extend-4.0
Requires: swak4Foam

Author:
Milad Setareh
Amirkabir University of
Technology (Tehran Polytechnic)
miladsetareh1989@gmail.com

Peer reviewed by:
Matilda Ronnfors
Håkan Nilsson

Disclaimer: This is a student project work, done as part of a course where OpenFOAM and some other OpenSource software are introduced to the students. Any reader should be aware that it might not be free of errors. Still, it might be useful for someone who would like learn some details similar to the ones presented in the report and in the accompanying files. The material has gone through a review process. The role of the reviewer is to go through the tutorial and make sure that it works, that it is possible to follow, and to some extent correct the writing. The reviewer has no responsibility for the contents.

January 20, 2017

Learning outcomes

The reader will learn:
How to use it:

- How to use the sonicLiquidFoam solver.
- How to use the buoyantBoussinesqSimpleFoam solver.

The theory of it:

- The theory of perturbation method.
- The theory of wave propagation into a fluid.

How it is implemented:

- A detail description of decomposed Navier–Stokes equations by perturbation method.
- A detail description of sonicLiquidFoam source code.

How to modify it:

- Define some new volVectorFields for calculation acoustic force in each cell.
- Use the acoustic force in myBuoyantBoussinesqSimpleFoam solver as source terms.

Contents

1 Acoustic streaming modeling **3**
1.1 Introduction . . . 3
1.2 Mathematical model . . . 3
1.2.1 First order equation . . . 4
1.2.2 Second order equation . . . 4
1.3 Modification of solvers . . . 4
1.3.1 Getting started . . . 4

2 Test solvers and run a case **9**
2.1 Mesh generation and transportProperties . . . 9
2.2 Boundary conditions and initial fields . . . 11
2.3 Property of fluid and simulation parameter . . . 11
2.4 Result (frequency 10MHz) . . . 12
2.4.1 First order . . . 12
2.4.2 Second order . . . 14

Chapter 1

Acoustic streaming modeling

1.1 Introduction

Propagation of ultrasonic waves into fluid generates a flow field which is knows as acoustic streaming. Acoustic streaming is a steady circular flow occurring near the vibrating surface in a high intensity sound field. This phenomenon is regarded with the friction between the vibrating medium in contact with a solid wall.

To reach a steady state solution of acoustic streaming, the Navier–Stokes equation should be solved with a time step at least one order less than period of ultrasonic waves. Obviously, reaching steady state takes too long so it should be applied an adequate method for overcoming this restriction. One of the ways to conquer time consuming of acoustic streaming solution is using perturbation method.

This tutorial describes how to modify the sonicLiquidFoam and buoyantBoussinesqSimpleFoam for acoustic streaming modeling. sonicLiquidFoam is a transient compressible solver for modeling the propagation of wave in a liquid and buoyantBoussinesqSimpleFoam is a steady state solver for buoyant, turbulent flow of incompressible fluids. In order to do this, the sonicLiquidFoam solver is modified to solve only first order Navier–Stokes equation, then calculate some source terms used as source terms in buoyantBoussinesqSimpleFoam solver to solve second order Navier–Stokes equation. By modifying these solvers, we can study acoustic streaming and heat transfer as well.

1.2 Mathematical model

Some methods has been presented for modeling of acoustic streaming: slip velocity [1], perturbation method [2] and direct method [3]. In this report, acoustic streaming is modeled by perturbation method. In this method, the variables in the Navier–Stokes and continuity equations are written as an expansion of equilibrium. The equilibrium value corresponds to zero order, represent constant in time and space, in the absence of acoustic excitation. Therefore, the variables are expanded as:

$$\begin{aligned} \rho &= \rho^0 + \rho^1 + \rho^2 \\ u &= u^0 + u^1 + u^2 \\ p &= p^0 + p^1 + p^2 \end{aligned} \tag{1.1}$$

where (0) refers to zero order, (1) to first order and (2) to second order The components of third and higher orders are neglected, since they are insignificant for describing the acoustic streaming effects. The first order variables represent the damped propagation of the acoustic wave, providing analysis of the instantaneous acoustic flow, while the second order variables describe the average acoustic streaming effects.

1.2.1 First order equation

The first order continuity, momentum and state equations are the following, respectively [2]:

$$\frac{\partial \rho^1}{\partial t} + \rho^0 \nabla \cdot \left(u^1\right) = 0 \tag{1.2}$$

$$\rho^0 \frac{\partial u^1}{\partial t} = -\nabla \cdot p^1 + \mu \nabla^2 u^1 \tag{1.3}$$

$$p^1 = c^2 \rho^1 \tag{1.4}$$

Where ρ^0 represents the equilibrium density, constant in time and space, ρ^1 is the first order density, u^1 is the first order velocity and c is sound speed. The time average driving force will be calculated after solving the first order system for all the required time. The time average value is obtained by integration of the driving force term during one wave period, after the stabilization of the periodic solution, and posterior division by the period duration. This time average driving force, described by Nyborg [5] can be applied as a source term to solve the second order system.

1.2.2 Second order equation

The second order Navier–Stokes system considers the first order solution as known data. It includes the second order terms and describes the mass and body force sources. The time average values determined above allow analyzing the acoustic streaming effects in a large time scale. Therefore, results of the second order system show the mean global flow. The momentum and continuity source terms are described by the following equations [2]:

$$\frac{\partial \rho^2}{\partial t} + \rho^0 \nabla \cdot \left(u^2\right) = \left\langle -\nabla \cdot \left(\rho^1 u^1\right) \right\rangle \tag{1.5}$$

$$\rho^0 \frac{\partial u^2}{\partial t} = -\nabla \cdot p^2 + \mu \nabla^2 u^2 + \left\langle -\rho^1 \frac{\partial u^1}{\partial t} - \rho^0 \left(u^1 \cdot \nabla\right) u^1 \right\rangle \tag{1.6}$$

$$p^2 = c^2 \rho^2 \tag{1.7}$$

The terms in angle brackets represent the non-linear driving source terms. The Source term for the continuity equation is negligible, so it may be ignored.

1.3 Modification of solvers

This section covers the necessary setup needed to modify sonicLiquidFoam and buoyantBoussinesqSimpleFoam. The first order system of equation is solved by using modified sonicLiquidFoam solver and the second order system of equation is solved by adding the source term in buoyantBoussinesqSimpleFoam solver.

1.3.1 Getting started

We will now start to develop our new sonicLiquidFoam and buoyantBoussinesqSimpleFoam solvers in the local solvers directory. Copy the original solvers there, change the corresponding file names, and then compile the solver so that we make sure the basic setup is correct:

```
cd $WM_PROJECT_DIR
cp -r --parents applications/solvers/compressible/sonicLiquidFoam \
$WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/compressible
mv sonicLiquidFoam mySonicLiquidFoam
cd mySonicLiquidFoam
mv sonicLiquidFoam.C mySonicLiquidFoam.C
sed -i 's/sonicLiquidFoam/mySonicLiquidFoam/g' Make/files
sed -i 's/FOAM_APPBIN/FOAM_USER_APPBIN/g' Make/files
wclean
wmake
```

```
cd $WM_PROJECT_DIR
cp -r --parents applications/solvers/heatTransfer/buoyantBoussinesqSimpleFoam \
$WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/heatTransfer
mv buoyantBoussinesqSimpleFoam myBuoyantBoussinesqSimpleFoam
cd myBuoyantBoussinesqSimpleFoam
mv buoyantBoussinesqSimpleFoam.C myBuoyantBoussinesqSimpleFoam.C
sed -i 's/buoyantBoussinesqSimpleFoam/myBuoyantBoussinesqSimpleFoam/g' Make/files
sed -i 's/FOAM_APPBIN/FOAM_USER_APPBIN/g' Make/files
wclean
wmake
```

Once it was compiled successfully, we can start to modify the solvers for our own sake to model acoustic streaming by implementing perturbation method on Navier–Stokes. First, we modify the mySonicLiquidFoam solver. So, we should go to that directory and do all things as below. To create source term field, we should add below lines in the `createFields.H` file.

```
    volVectorField sumAcousticForce
    (
        IOobject
        (
            "sumAcousticForce",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        mesh,
        dimensionedVector("meanAcousticForce",  dimensionSet(0,1,-2,0,0,0,0),
        vector(0,0,0))
    );
```

```
volVectorField meanAcousticForce
    (
        IOobject
        (
            "meanAcousticForce",
```

```
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh,
         dimensionedVector("meanAcousticForce", dimensionSet(0,1,-2,0,0,0,0),
         vector(0,0,0))
    );
```

Also, we change the initial value of ρ to `1+psi*p` in `createFields.H`. For considering kinematic viscosity, it is better to change the word `mu` in `readTransportProperties.H` file to `nu`.

Now we can modify the main mySonicLiquidFoam.C file to implement the first order governing equation. First we declare a `float` variable named `iter` before the time loop and initialize it with zero. Then, we change compressibleCourantNo.H to CourantNo.H everywhere in source code by the following line:

```
sed -i 's/compressibleCourantNo.H/CourantNo.H/g' mySonicLiquidFoam.C
```

The incompressible momentum equation is considered for first order equations,Therefore it is not necessary to solve the density equation and we can remove it. For this purpose, delete the line: `#include "rhoEqn.H"`, located above UEqn in the source code. Now we should implement the first order equations. We can change the momentum and pressure equations line by line, but I think it is better to remove all lines starting with `fvVectorMatrix UEqn` and ended with `rho = rho0 + psi*p`. Then paste the below code instead of removed.

```
        fvVectorMatrix UEqn
        (
            fvm::ddt(U)
          - fvm::laplacian(nu, U)
        );

        solve(UEqn == -fvc::grad(p));

        // --- PISO loop

        for (int corr=0; corr<nCorr; corr++)
        {
            volScalarField rAU(1.0/UEqn.A());
            U = rAU*UEqn.H();

            surfaceScalarField phid
            (
                "phid",
                psi
               *(
                    (fvc::interpolate(U) & mesh.Sf())
                 // + fvc::ddtPhiCorr(rAU, rho, U, phi)
                )
            );
```

```
            phi = (1/psi)*phid;

            fvScalarMatrix pEqn
            (
                fvm::ddt(psi,p)
              + fvc::div(phi)
              //+ fvm::div(phid, p)
              - fvm::laplacian(rAU, p)
            );

            pEqn.solve();

            phi += pEqn.flux();

                #include "continuityErrs.H"

            U -= rAU*fvc::grad(p);
            U.correctBoundaryConditions();
        }
        rho = 1 + psi*p;
        iter=iter+1.0;
```

Some modifications have been done in mySonicLiquidFoam, including the convection term has been removed in the Ueqn part and the PISO loop was used. Also the variable named iter has been added at the last line.

The most important part is the calculation of the driving force. The formula of the momentum driving force has been shown in the previous sections. Now add the following lines at the end of time loop.

```
sumAcousticForce=sumAcousticForce+fvc::div(phi, U)+(rho-1)*fvc::ddt(U);
meanAcousticForce=sumAcousticForce/iter;
```

The momentum driving force is calculated at the first line of the above box. However, we need its time average for the second order equations, so at the second line of the above box the driving force is divided by the number of iterations. When the solution has converged, the `meanAcousticForce` variable expresses the time-averaged driving force for the second order momentum equation.

Now we have completed the implementation new features that the mySonicLiquidFoam solver should have. Type *wmake* in the terminal and compile the new solver. If everything is ok we can go to the second part, which is the modification of the myBuoyantBoussinesqSimpleFoam solver. Now, go to the myBuoyantBoussinesqSimpleFoam solver directory. We just modify `createFields.H` and `UEqn.H` files.

The solver needs to recognize the driving momentum source. First we sgould define the `meanAcousticForce` in the `createFields.H` as follows:

```
volVectorField meanAcousticForce
   (
       IOobject
       (
           "meanAcousticForce",
           runTime.timeName(),
           mesh,
           IOobject::MUST_READ,
```

```
            IOobject::AUTO_WRITE
        ),
        mesh
    );
```

Now it's time to add this source term to the second order momentum equation. The formula of the second order equation is shown in the equatios 1.6. As you can see, the convection term was ignored in that equation due to high order of magnitude but for more accuracy, we consider it. We can ignore it just by removing the `fvm::div(phi, U)` in the `UEqn.H` file.

To import the driving momentum source into the U equation, we should add `meanAcousticForce` in the `UEqn` as shown in the following:

```
    tmp<fvVectorMatrix> UEqn
    (
        fvm::div(phi, U)
      + turbulence->divDevReff(U)
      +meanAcousticForce
    );
```

It is noteworthy that for running second order case by myBuoyantBoussinesqSimpleFoam solver, it is necessary to copy the meanAcousticForce file from the last time directory of first order case and paste it to the start time directory of the second order case. Now we can compile the solver by typing `wmake`. The Following part of the tutorial focuses on how to use these solvers and run a case.

Chapter 2

Test solvers and run a case

In this chapter, a simple case will be run and we will observe the results. To generate pressure wave at the boundary, I considered harmonic pressure boundary condition at one of the walls. The `groovyBC` has been used to implement that boundary condition. `groovyBC` is a library in the swak4Foam package which has many types of boundary conditions [4].

2.1 Mesh generation and transportProperties

This example is based on the icoFoam/cavity tutorial. The lines below, copy the cavity tutorial to the local run directory.

```
mkdir -p $FOAM_RUN/cavity
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity $FOAM_RUN/cavity
cd $FOAM_RUN/cavity
mv cavity cavityFirstOrder
```

Open the blockMeshDict and create the mesh and set the boundary conditions as follows:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.0001;

vertices
(
    (0 0 -0.1)//0
    (10 0 -0.1)//1
```

```
    (10 20 -0.1)//2
    (0 20 -0.1)//3
    (0 0 0.1)//4
    (10 0 0.1)//5
    (10 20 0.1)//6
    (0 20 0.1)//7
    (14 0 -0.1)//8
    (14 20 -0.1)//9
    (14 0 0.1)//10
    (14 20 0.1)//11
    (24 0 -0.1)//12
    (24 20 -0.1)//13
    (24 0 0.1)//14
    (24 20 0.1)//15
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (400 700 1) simpleGrading (1 1 1)
    hex (1 8 9 2 5 10 11 6) (400 700 1) simpleGrading (1 1 1)
    hex (8 12 13 9 10 14 15 11) (400 700 1) simpleGrading (1 1 1)
);

boundary
(
    walls
    {
        type wall;
        faces
      (
        (0 4 7 3)
        (12 13 15 14)
      );
    }
    inlet_w1
    {
        type wall;
        faces
      (
        (3 2 6 7)
      );
    }

    acous
    {
        type wall;
        faces
      (
        (2 9 11 6)
      );
    }

    inlet_w2
    {
```

```
        type wall;
        faces
      (
        (9 13 15 11)
      );
    }
    outlet
    {
        type wall;
        faces
      (
            (0 1 5 4)
            (1 8 10 5)
            (8 12 14 10)
      );
    }
    frontAndBack
    {
        type empty;
        faces
      (
            (0 3 2 1)
            (1 2 9 8)
            (8 9 13 12)
            (4 7 6 5)
            (5 6 11 10)
            (10 11 15 14)
      );
    }
);
```

We should create the `thermodynamicProperties` in the constant directory for first order simulation. we can copy it from one of the tutorial cases in the sonicLiquidFoam directory as follows:

```
cp -r $FOAM_TUTORIALS/compressible/sonicLiquidFoam/decompressionTank/constant/\
thermodynamicProperties $FOAM_RUN/cavity/cavityFirstOrder/constant
```

2.2 Boundary conditions and initial fields

Figure 2.1 shows boundary conditions for the first and second order system of equations.

The initial conditions are set to zero for both the velocity and pressure for the first and second order equations. As shown in the left figure, one of the walls at the top has been considered as a pressure source. This means that a wave is generated at this boundary and propagate into domain. Also, It is assumed that the pressure gradient at the other walls equals to zero and it means that the wave will reflect at these walls.

2.3 Property of fluid and simulation parameter

Table 2.1 shows the fluid properties and some simulation parameters that are used to the simulate first order equations. We should change the values of the parameters in `thermodynamicProperties` as follows:

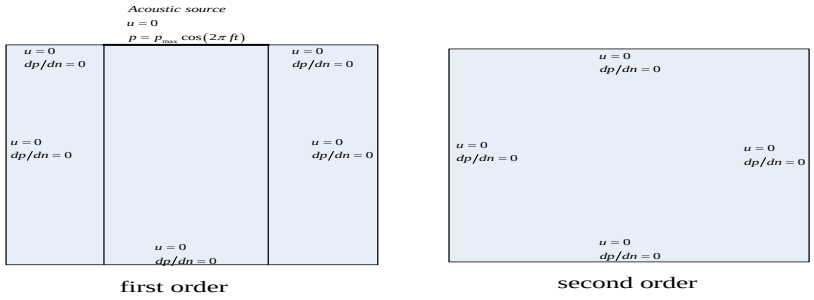


Figure 2.1: Boundary condition for the first and second order equations

Parameter	Value
Sound Speed	1440 m/s
Frequency	10 MHz
Pressure Amplitude	20 kPa
Density	998 kg/m^3
Kinematic viscosity	89 e-08 m^2/s
Time Step–$1^{st} order$	$10^{-9} s$
Simulation time–$1^{st} order$	$\mathrm{O}(10^{-5})\ s$

Table 2.1: Property of fluid and some parameters for numerical modeling

```
rho0            rho0 [ 1 -3 0 0 0 0 0 ] 997;

p0              p0 [ 1 -1 -2 0 0 0 0 ] 100000;

psi             psi [ 0 -2 2 0 0 0 0 ] 4.44e-07;
```

Now, everything is ready to start the simulation. First go to the case directory and use `blockMesh` to create mesh, Then, write `mySonicLiquidFoam` to start running. It has to be added `"libOpenFOAM.so"` and `"libgroovyBC.so"` at the end of `controlDict`. lines below show the code that should be added at the end of `controlDict`.

```
libs (
      "libOpenFOAM.so"
      "libgroovyBC.so"
     );
```

2.4 Result (frequency 10MHz)

2.4.1 First order

To validate the code, A case with the same specifications in [6] was simulated. The value of thermo-physical properties and time step are shown in Table 2.1. Figure 2.2 compares the present results with of those Catarino et al. [6]. The average y-component momentum source is shown for 4 times

(5e-07s, 5e-06s, 8e-06s and 10e-06s) by my code and Catarino et al. [6]. The results show good matching. Results show that the average value of momentum source will be steady so the value of all parameters at final time will be used in the following.

Figure 2.3 shows the comparison between instantaneous pressure contour in the present work and Catarino et al. [6]. The results show good matching between present work and Catarino et al. [6]. The instantaneous pressure in the fluid appears as high and low pressure stripes, as a result of the oscillatory vibration. The propagation of pressure occurs in a semicircular shape with intensity decaying with the distance to the acoustic source. In this case, viscous attenuation has the most important role to attenuate wave intensity so most power of wave is attenuated by the viscosity of fluid before the wave reached to the opposite wall. Also, it is important to note that the number of strips depends on the wave frequency and the speed of wave in the propagation media. In this case, the wavelength is about 0.14 mm and the height of the enclosure is 2 mm therefore we can observe 24 strips by ignoring the viscous attenuation but with considering viscosity we observe less than 24 strips.

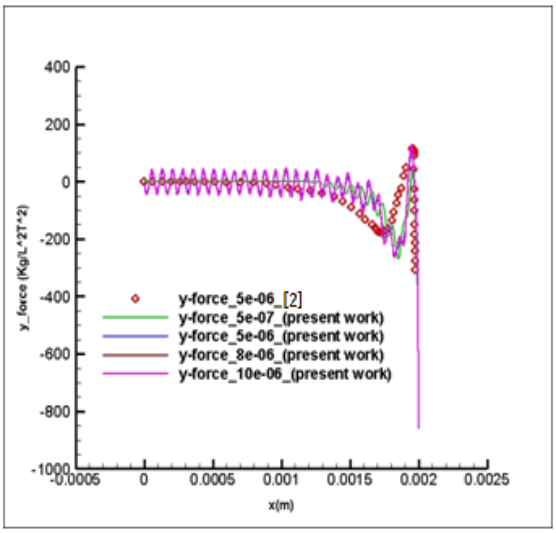
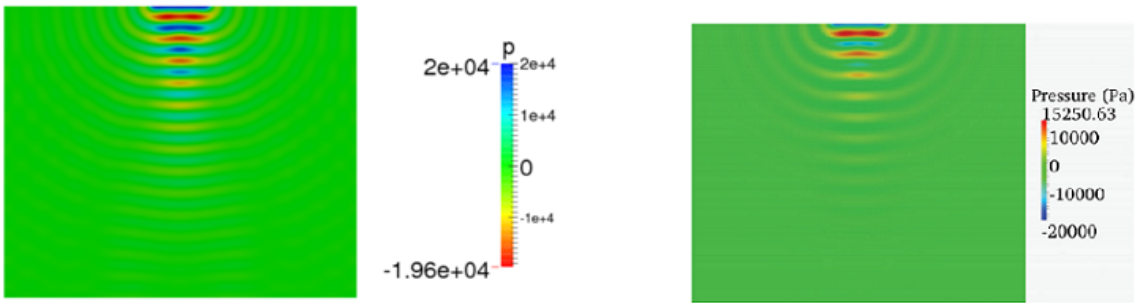


Figure 2.2: average of y-component momentum source at various times



(a) present work (b) Catarino et al. [6]

Figure 2.3: Propagation of instantaneous pressure

2.4.2 Second order

This example is based on the icoFoam/cavity tutorial. Copy the cavity tutorial to the run directory and change its name to cavitySecondOrder as follows:

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity $FOAM_RUN/cavity
cd $FOAM_RUN/cavity
mv cavity cavitySecondOrder
```

Now, change the `blockMeshDict` exactly the same as previous section. Also, we can copy this file from `cavityFirstOrder` and use it for `cavitySecondOrder` case.

```
rm -rf $FOAM_RUN/cavity/cavitySecondOrder/constant/polyMesh
cp -r $FOAM_RUN/cavity/cavityFirstOrder/constant/polyMesh \
$FOAM_RUN/cavity/cavitySecondOrder/constant/polyMesh
```

Now we remove the `transportProperties` in the `cavitySecondOrder` and copy `transportProperties` from one of the tutorial cases in the buoyantBoussinesqSimpleFoam directory. Also, the `g` file(gravitational acceleration) must be in the `constant` directory. By copying and pasting the below lines, both those modification will be done automatically.

```
rm -rf $FOAM_RUN/cavity/cavitySecondOrder/constant/transportProperties
```

```
cp -p $FOAM_TUTORIALS/heatTransfer/buoyantBoussinesqSimpleFoam/hotRoom/constant\
/g $FOAM_RUN/cavity/cavitySecondOrder/constant
cp -p $FOAM_TUTORIALS/heatTransfer/buoyantBoussinesqSimpleFoam/hotRoom/constant/\
transportProperties $FOAM_RUN/cavity/cavitySecondOrder/constant
```

The `buoyantBoussinesqSimpleFoam` solver is used for modeling the buoyant flow with the Boussinesq approximation. Since, we do not want to model this kind of flow, we should assign the gravity (`g`) and `beta` to zero. Now, go to the cavitySecondOrder directory and run the case by typing `buoyantBoussinesqSimpleFoam` in the terminal.

Figure 2.4 shows the acoustic streaming in the domain. As you see, there are two large vortices which cause more fluid mixing and decrease the effect of the boundary layer near solid walls. Therefore, we expect that heat transfer enhances due to acoustic streaming.

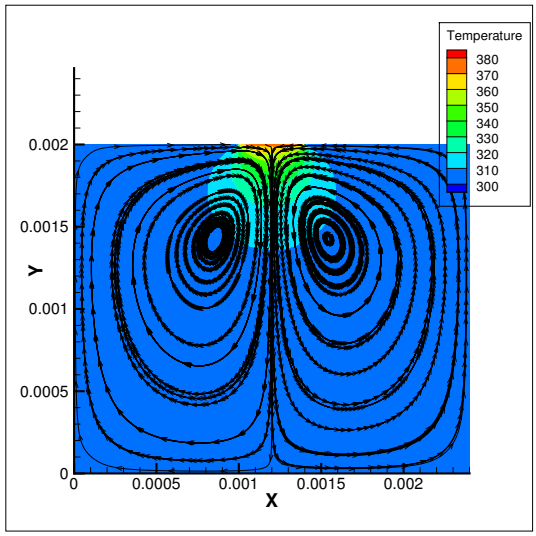


Figure 2.4: Acoustic streaming

Study questions

1. What is the purpose of perturbation method?
2. What is application of sonicLiquidFoam and buoyantBoussinesqSimpleFoam solvers?
3. What is `psi` in `sonicLiquidFoam` solver?
4. How can new volVectorField be defined in a solver?
5. where was acoustic source term used in the solvers?

Bibliography

[1] P. Vainshtein, M. Fichman, C. Gutfinger, Acoustic enhancement of heat transfer between two parallel plates, International Journal of Heat and Mass Transfer, no 38, Issue 10, 1995, Pages 1893-1899

[2] Muller, P.B , Rune, R., Jensen, M.J.H, Bruus, H. A numerical study of microparticle acoustophoresis driven by acoustic radiation forces and streaming-induced drag forces, Lab Chip, Vol 12, no 22, pages 4617-4627, 2012

[3] B. Tajik, A. Abbassi, M. Saffar-Avval, A. Abdullah, H. Mohammad-Abadi, Numerical simulation of acoustic streaming for nonlinear standing ultrasonic wave in water inside axisymmetric enclosure, Engineering Applications of Computational Fluid Mechanics, Vol 6, no 3, pages 366â382, 2012

[4] https://openfoamwiki.net/index.php/Contrib/swak4Foam

[5] W. L. Nyborg Acoustic streaming, Academic Press, San Diego 1998

[6] Catarino, Susana O. and Miranda, Joao M. and Lanceros-Mendez, Senentxu and Minas, Graca, Numerical prediction of acoustic streaming in a microcuvette, The Canadian Journal of Chemical Engineering, Vol, 92, no, 11, 2014