Computational Aeroacoustics Methods with OpenFOAM v. 1812

V. Korchagova, M. Kraposhin, S. Strizhak

Institute for System Programming of Russian Academy of Sciences

Web-laboratory UniCFD

https://unicfd.ru

v.korchagova@ispras.ru

Structure

Theoretical part

1. Introduction
2. Problem statement
3. Available CAA methods
4. Curle's analogy
5. FFWH analogy
6. Boundary Element Method
7. Implementation: `libAcoustics` library

Hands-on part: pulsating sphere

1. Problem statement
2. One-dimensional case
   - Case description
   - How to set up FFWH analogy in OpenFOAM
   - Settings for hybrid CFD/BEM computations
   - Results
3. Basic BEM++ case
4. One-dimensional case
   - Case description
   - Settings in OpenFOAM
   - Results
5. Discussion

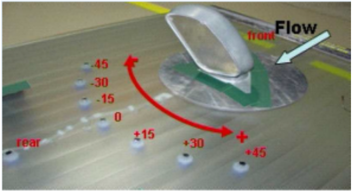
Part I
Theoretical part

Computational Aeroacoustics: issues and methods

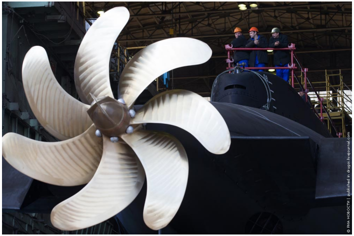
Computational Aeroacoustic applications

Rocket lift-off

Aircfart engines



Flow around wing mirror



Submarine propellers

Sound pressure levels around us

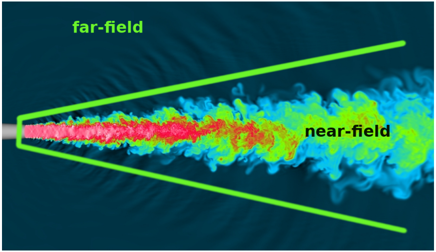
Sound source	SPL, dB
Good recording studio	20
Whisper	30
Average home	40–50
Conversational speech, 1 m	60
Vacuum cleaner	70
Diesel truck (10 m distance)	90
Disco (1 m distance)	100
Chainsaw (1 m distance), rock concert	110
Discomfort threshold, near an air	120
Pain threshold, near an air	130-140
Gun shot	140
Jet engine (1 m distance)	130-160
Rocket lift-off	140-180

Danger zone: $>$140 dB!

Computational complexities

- Complicated structure of flows
- High-frequency acoustic waves
- Requirements for numerical solution:
  - stability;
  - low diffusivity;
  - correct simulation of turbulent structures;
  - correct simulation in the far-field,

  consequently,
- large volume of processed data.



How to solve?

- direct numerical simulation;
- aeroacoustic analogies;
- boundary element methods.

Lighthill's analogy

Hydrodynamic equations

Mass conservation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho U_j)}{\partial x_j} = 0.$$

Momentum conservation:

$$\frac{\partial \rho U_i}{\partial t} + \frac{\partial (\rho U_i U_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}, \quad \tau_{ij} = \mu \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3} \frac{\partial U_j}{\partial x_j} \delta_{ij} \right).$$

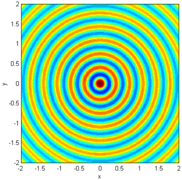
Lighthill's equation

$$\frac{\partial^2 \rho'}{\partial t^2} - c^2 \nabla^2 \rho' = \frac{\partial T_{ij}}{\partial x_i \partial x_j}$$ — quadrupole sources,

$$T_{ij} = \rho U_i U_j + \delta_{ij} \left(p' + c^2 \rho' - \tau_{ij} \right)$$ — Lighthill's stress tensor

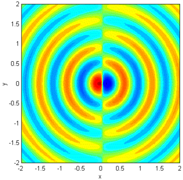
Acoustic sources

Monopole



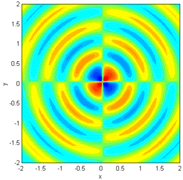
$$\frac{\partial^2 p'}{\partial t^2} + \nabla^2 p' =$$
$$= -f(t)\ \delta(\mathbf{x} - \mathbf{x_0})$$

Dipole



$$\frac{\partial^2 p'}{\partial t^2} + \nabla^2 p' =$$
$$= -\frac{\partial}{\partial x_j}(f_j(t)\ \delta(\mathbf{x}-\mathbf{x_0}))$$

Quadrupole



$$\frac{\partial^2 \rho'}{\partial t^2} - c^2\nabla^2 \rho' =$$
$$= \frac{\partial T_{ij}}{\partial x_i \partial x_j}$$

Curle's analogy[1]

Fundamental result:

$$\rho' = \frac{1}{4\pi c^2}\frac{\partial^2}{\partial x_i\,\partial x_j}\int_V \frac{T_{ij}(\mathbf{y}, t - r/c)}{r}\,d\mathbf{y} + \frac{1}{4\pi c^2}\frac{\partial}{\partial x_i}\int_S \frac{P_i(\mathbf{y}, t - r/c)}{r}\,d\mathbf{y}$$

Simplifications:

- no volumetric sources;
- isentropic flow in the region of interest;
- retarded time is neglected.

Sound pressure computations

$$p' = \frac{1}{4\pi c}\frac{x_i}{r^2}\frac{\partial F_i}{\partial t}, \quad F_i = \int_S P_i(\mathbf{y}, t)\,dS(\mathbf{y})$$

[1] Curle N. The influence of solid boundaries upon aerodynamic sound. Proc. Royal Soc., 1955. 231A. P. 505–514.

Curle's analogy

Computational algorithm

1. Define static solid surface around the source region
2. Compute resultant force F exerted upon the fluid by the surface
3. Compute sound pressure in the observer position

Limitations

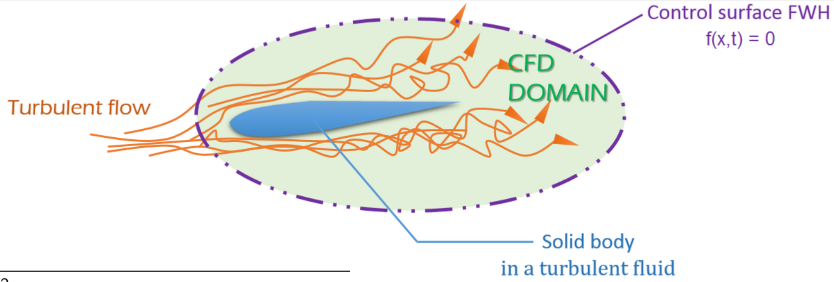
- Low-speed flows
- Static non-deformable surface

Ffowcs-Williams — Hawkings analogy[2]

FFWH equation

$$\frac{\partial^2 \rho' H(f)}{\partial t^2} - c^2 \nabla^2 \rho' H(f) =$$

$$= \frac{\partial T_{ij} H(f)}{\partial x_i \partial x_j} - \frac{\partial}{\partial x_i}\left(\tau_{ij}\delta(f)\frac{\partial f}{\partial x_j}\right) + \frac{\partial}{\partial t}\left(\rho_0 u_i \delta(f)\frac{\partial f}{\partial x_j}\right)$$



[2] J.E. Ffowcs Williams and D.L. Hawkings: Sound generated by turbulence and surfaces in arbitrary motion, Philosophical Transactions of the Royal Society, A264, 1969, 321-342

Ffowcs-Williams — Hawkings analogy

Formulations[3]

Different formulations: Farassat 1, 1A, Q1, Q1A.

Computational algorithm

1. Update surface position around the source region
2. Update distances from observers to surface
3. Compute sound pressure in the observer position using some formulation

[3] Farassat F. Derivation of Formulations 1 and 1A of Farassat. Langley Research Center, Hampton, Virginia. 2007

Boundary Integral Method

Transformation to frequency domain

Wave equation for exterior Dirichlet problem:

$$\frac{\partial^2 p(x,t)}{\partial t^2} - c^2 \nabla^2 p(x,t) = 0, \quad x \in \Omega_e = \mathbb{R}^3 \backslash \overline{\Omega}.$$

Sound pressure is the harmonic function:

$$p(x,t) = \mathrm{Re}\ (u(x) e^{-i\omega t}).$$

Helmholtz equation for complex sound pressure

$$\Delta u(x) + k^2 u(x) = 0 \text{ for } x \in \Omega_e.$$

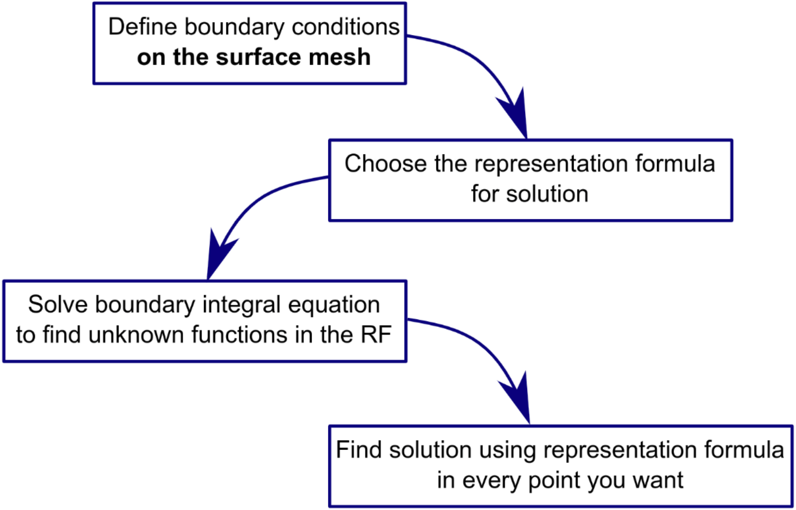
Boundary conditions

$$u = g(x) \text{ for } x \in \Gamma = \partial\Omega;$$

$$\lim_{r \to \infty} r \left(\frac{\partial u(x)}{\partial r} - iku(x) \right) = 0, \quad r = |x|, \quad x \to \infty.$$

Boundary Integral Method

Overview



Boundary Integral Method

Direct and indirect approach: representation formula

Direct approach

$$u(x) = \int\limits_{\partial\Gamma} \frac{\partial G(x,y)}{\partial n_y} g(y)\, dS_y - \int\limits_{\partial\Gamma} \frac{\partial u(y)}{\partial n_y} G(x,y)\, dS_y, \quad x \in \Omega_c.$$

Indirect approach

– via a single layer potential:

$$u(x) = (\tilde{V}_k w)(x) = \int\limits_{\partial\Gamma} G(x,y) w(y)\, dS_y, \quad x \in \Omega_c.$$

– via a double layer potential:

$$u(x) = (W_k v)(x) = \int\limits_{\partial\Gamma} \frac{\partial G(x,y)}{\partial n_y} v(y)\, dS_y, \quad x \in \Omega_c.$$

Boundary Integral Method

Direct and indirect approach: boundary integral equations

Direct approach

$$\int\limits_{\partial\Gamma} G(x,y)t(y)\,dS_y = -\frac{1}{2}g(x) + \int\limits_{\partial\Gamma} \frac{\partial G(x,y)}{\partial n_y} g(y)\,dS_y, \quad x \in \Gamma.$$

Indirect approach

– via a single layer potential:

$$(V_k w)(x) = \int\limits_{\partial\Gamma} G(x,y)w(y)\,dS_y = g(x), \quad x \in \Gamma.$$

– via a double layer potential:

$$\frac{1}{2}v(x) + (K_k v)(x) = \frac{1}{2}v(x) + \int\limits_{\partial\Gamma} \frac{\partial G(x,y)}{\partial n_y} v(y)\,dS_y = g(x), \quad x \in \Gamma.$$

Boundary Integral Method

Combined formulation of BIE[4]

Representation of solution

$$u(x) = (\tilde{V}_k w)(x) - i\eta(W_k w)(x), \quad x \in \Omega_c.$$

Boundary integral equation

$$\left(\frac{1}{2}I + K\right) w(x) - i\eta(V_k w)(x) = g(x), \quad x \in \Gamma.$$

Boundary integral operator is injective for all $k \in \mathbb{R}_+$.

[4] Engleder S., Steinbach O. Modified boundary integral formulations for the Helmholtz equation // Journal of Mathematical Analysis and Applications, 2007. Vol. 331. P. 396–407.

Boundary Integral Method

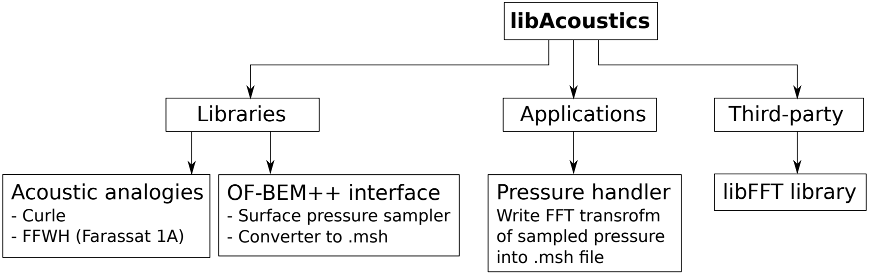
Computational algorithm

1. Define static solid surface around the source region
2. Run CFD simulation with pressure sampling on the surface
3. Transform sample data in the frequency domain
4. Export data to another open-source package
5. Solve boundary integral equations
6. Compute resultant sound pressure in the observer position

BEM++

Solver of elliptic equations with boundary element method.
Use Python 3 interface and C++ kernel.
Actual version: v. 3.3.4.

libAcoustics



Download:

```
git clone https://github.com/unicfdlab/libAcoustics.git
cd libAcoustics
git checkout digitef-dev-1812
```

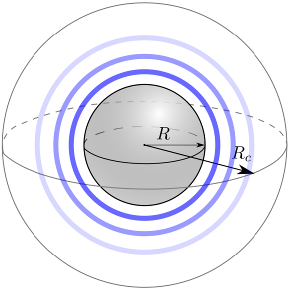
Install:

```
cd Sources
./wmakeAll.sh
```

Part II
Practical part

`libAcoustics` training

Problem statement



Input data

Radius of sphere: $R = 0.1$ m
Velocity oscillations:
$U(R,t) = U_0 \sin(2\pi f t)$
Velocity amplitude: $U_0 = 0.01$ m/s
Frequency: $f = 100$ Hz
Speed of sound: $c = 100$ m/s
Density of gas: $\rho_0 = 14.1855$ kg/m^3

Check

Sound pressure and SPL in points:
$(0,0,1)$, $(0,0,2)$, $(0,0,3)$,
$(0,0,4)$, $(0,0,5)$, $(0,0,10)$.

Analytical solution

$$p(r,t) = \mathrm{Re}\left[\frac{A}{r} e^{-i(\omega t - kr)}\right];$$

here $\omega = 2\pi f$, $A = \rho_0 c U_0 e^{-ikR}$, $k = \omega/c$.

Numerical simulation

Compare techniques

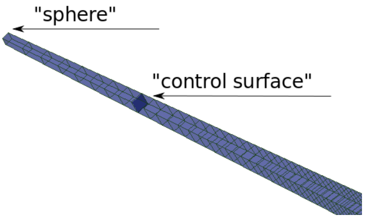
- direct FVM OpenFOAM simulation;
- FFWH using OpenFOAM;
- pure BEM using BEM++;
- CFD/BEM using both OpenFOAM and BEM++.

Tests

- 1D-case in OpenFOAM
- 3D-case in bempp
- 3D-case in OpenFOAM

1D case

Flow domain



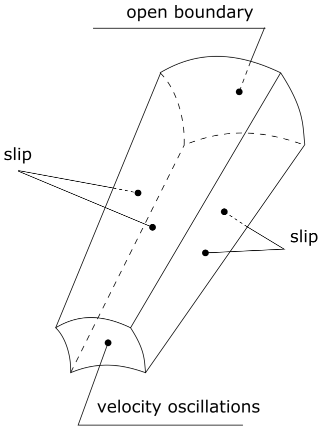
Mesh information

Length: 14.9 m.
Mesh size: 2100 cells.

Numerical parameters

Solver: rhoCentralFoam
Turbulence model: laminar flow

Scheme of boundaries



1D case

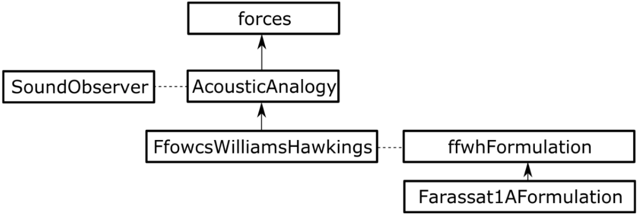
Boundary conditions

Name	T	U	p
sphere	fixedValue 24.8084307131	uniformFixedValue sine	zeroGradient
free	zeroGradient	waveTransmissive	waveTransmissive
cyc	slip	slip	slip

Note

Temperature value was calculated according to sound speed 100 m/s

How to use FFWH in OpenFOAM I



Use **functionObjectAPI** for running

In `system/controlDict`:

```
    functions {
        ffwh1 {
            libs ("libAcoustics.so");
            ...
        }
    }
```

Settings in dictionary I

1. General settings for acoustic analogies:
   - `libs ("libAcoustics.so")` — usage of the libAcoustics library;
   - `log` — logging info;
   - `probeFrequency` — step for acoustic probe;
   - `timeStart`, `timeEnd` — time interval;
   - `c0` — sound speed;
   - `dRef` — domain depth for 2D computations;
   - `pName` — name of pressure field;
   - `pInf` — reference pressure;
   - `rhoName` — name of density field;
   - `rhoInf` — reference density;
   - `CofR` — origin;
   - `writeFft` — true if need write FFT transform;
   - `U0` — reference velocity;
   - list of observers.
2. Special settings for FFWH:

Settings in dictionary II

- `type` — type of acoustic analogy;
- `interpolationScheme` — interpolation scheme for control surface;
- `formulationType` — FFWH formulation
- `surfaces` — list of control surfaces;
- `nonUniformSurfaceMotion` — true if surface can move with mesh
- `Ufwh` — fixed velocity of control surace moving
- `cleanFreq` — step for FFT computations;
- `fixedResponceDelay` — true if responce delay is fixed;
- `responceDelay` — value of responce delay.

Control surface in case: "sphere" patch or small quadrangle placed in the wedge.

Results for 1D case

FFWH

Results are increased: the control surface is just a small part of sphere, computations of surface integral are not correct.

CFD/BEM

Almost zero due to strong 3D statement of BIE. Trick: just set the calculated value of complex pressure to all nodes of 3D sphere (separate geometry) and use this data for BIE computations.

Results for two first microphones

Microphone	Analytical	FFWH	CFD/BEM, 1D	CFD/BEM, 3D
$(0\ 0\ 1)$	≈ 0.7549	≈ 1.08	$\approx 10^{-5}$	≈ 0.73
$(0\ 0\ 2)$	≈ 0.3775	≈ 0.54	$\approx 10^{-5}$	≈ 0.35

How to use BEM++[5]

How to run simple case

- Create a Python script which uses BEM++ Python library to solve Helmholtz equation.
- Run: `python3 scriptname.py`

Structure of bempp basic case folder

- Python 3 script for case.
- Folder `subscripts/`: post-processing, computations of analytical solution.
- Folder `output/`: contains solution, result of post-processing. Appears after successful solution.
- Folder `graph/`: set of gnuplot scripts for data visualization.

Structure of folder is arbitrary.

[5] http://www.bempp.org/

bempp script for simple case I

1. Import Python libraries, bempp Python API, OF–bempp interface:

```
import bempp.api
import numpy as np
from cfdbem.grid import merge
from cfdbem.file_interfaces import FileReader
from bempp.api.linalg import gmres
```

2. Define basic variables:

```
freq        = 100                          # frequency
c       = 100                          # speed of sound
epsilon = 1E-5                         # solution
    accuracy
k       = freq * 2 * np.pi / c  # wave number
muD     = 1.0/k                        # numerical
    constant for CBIE formulation
```

bempp script for simple case II

3. Import mesh from file using OF–bempp interface:

```
reader = FileReader(file_name = "cs.msh")
grid = reader.grid
```

4. Define spaces:

```
piecewise_lin_space   =
   bempp.api.function_space(grid,"P",1)
piecewise_const_space =
   bempp.api.function_space(grid,"DP",0)
```

5. Choose spaces for boundary integral operators:

```
domain_space         = piecewise_lin_space
range_space          = piecewise_lin_space
dual_to_range_space = piecewise_lin_space
```

6. Define boundary integral operators:

bempp script for simple case III

```
identity = bempp.api.operators.
    boundary.sparse.identity(
          domain_space, range_space,
              dual_to_range_space)
dlp = bempp.api.operators.
    boundary.helmholtz.double_layer(
          domain_space, range_space,
              dual_to_range_space,k)
slp = bempp.api.operators.
    boundary.helmholtz.single_layer(
     domain_space, range_space,
         dual_to_range_space,k)
```

7. Define Dirichlet boundary condition using OpenFOAM data:

```
dirichlet_fun =
    bempp.api.GridFunction(piecewise_lin_space,
    coefficients =             reader.node_data)
```

bempp script for simple case IV

8. Write left-hand side of boundary integral equation:

```
lhs = (.5*identity + dlp) - 1j * muD * slp
```

9. Solve BIE with GMRES method:

```
w_fun,info = gmres(lhs, dirichlet_fun,
    tol=epsilon)
```

10. Define function for computations of sound pressure in appropriate points:

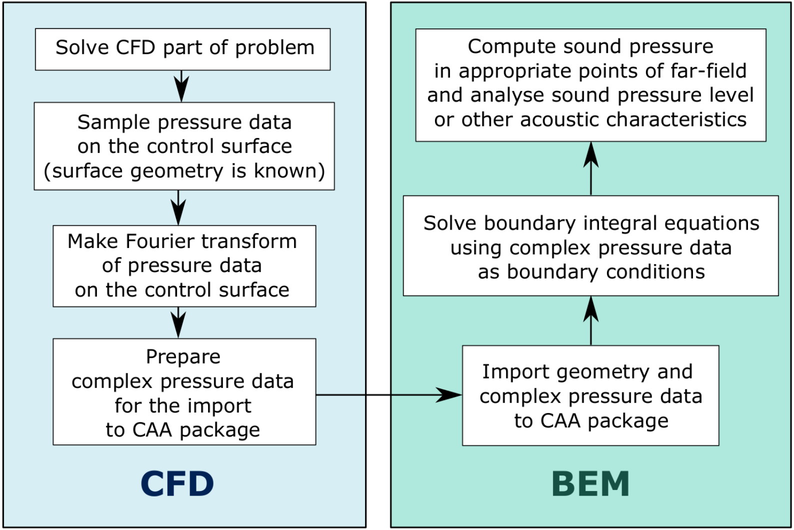
```
def result (points):
    from bempp.api.operators.potential import
        helmholtz as helmholtz_potential

    slp_pot = helmholtz_potential.single_layer(
        piecewise_lin_space, points, k)
```

bempp script for simple case V

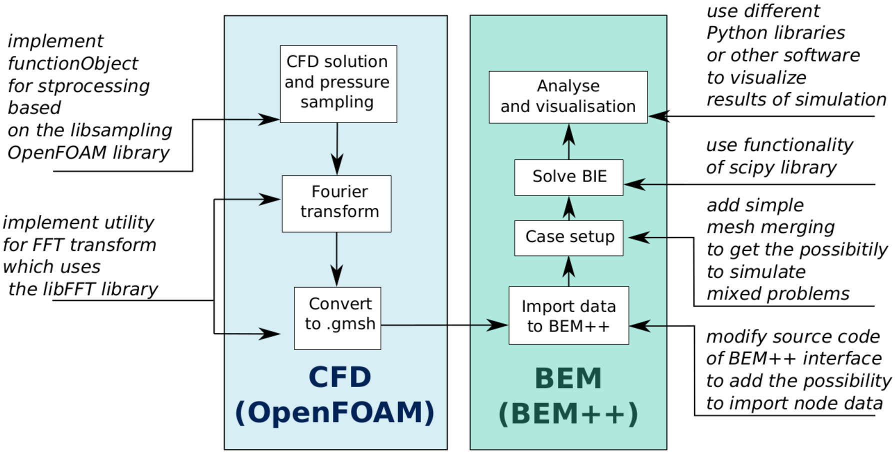
```
dlp_pot = helmholtz_potential.double_layer(
    piecewise_lin_space, points, k)

res =  1j * muD * slp_pot.evaluate(w_fun) -
    dlp_pot.evaluate(w_fun)

return res
```

11. Use standard or user-defined Python 3 functions for data postprocessing.

General scheme of hybrid CFD/BEM technology



CFD/BEM implementation using OpenFOAM and BEM++



Settings in OpenFOAM for CFD/BEM interface I

soundPressureSampler functionObject

- `libs ("libAcoustics.so")` — usage of the libAcoustics library;
- `type` — type of functionObject;
- `writeControl` — what time poinst should be output;
- `outputGeometryFormat` — format for control surface geometry;
- `fields` — what fields should be sampled;
- `pName` — name of pressure field;
- `log` — if true view log info;
- `interpolationScheme` — interpolation scheme for control surface;
- list of control surfaces.

Settings in OpenFOAM for CFD/BEM interface II

surfaceNoise utility

- `outputFormat` — format for data output;
- `inputFileName` — file name of sampled pressure data;
- `maxFrequency` — maximal frequency to be computed.

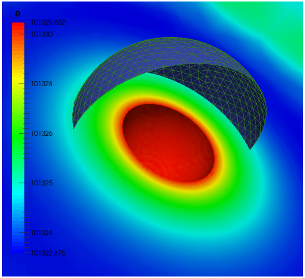
Notes for OF/bempp technology usage

- Set the environment variable before running:

```
export PYTHONPATH=$PYTHONPATH:<libAcoustics_dir>/cfdbem
```

- Only 3D problems.
- Control surface must be triangulated.
- Input format for bempp: only gmsh.
- No separate solver for BIE: all steps for solution are written in one script.

3D case



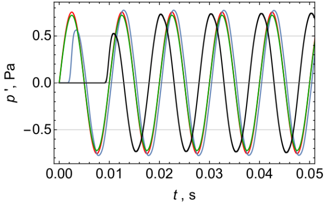
Spherical control surfaces

Radius: 0.2 m.

Mesh resolution:

1. $h_1 = 0.04$ m;
2. $h_2 = 0.02$ m;
3. $h_3 = 0.01$ m;

Due to long computational time all results are saved in the `res/` folder



Microphone (0 0 1)
Control surface 1

Blue color — FFWH
Red color — analytical solution
Green color — CFD/BEM
Black line — FVM

Summary

- We made an overview of CAA methods.
- We have discussed the `libAcoustics` library for CAA computations in OpenFOAM.
- We have known about new open-source code BEM++ for BIE calculations.
- We have known how to make a hierarchical numerical model using OpenFOAM and BEM++.
- We have tested different methods on the test case (pulsating sphere): pure FVM modelling, FFWH acoustic analogy, CFD/CAA hybrid technique.

Thank you for attention!
Some questions?