D7.4 REMOTE-ACCESS INTERFACE

Olov Engwall[1], Joan Baiges[2], Saeed Dabbaghchian[1], Niyazi Cem Degirmenci[1], Örjan Ekeberg [1], Johan Hoffman[1], Johan Jansson[1], Andreas Selamtzis[1], Sten Ternström[1]

[1] School of Computer Science and Communication, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden

[2] *Centre Internacional de Mètodes Numèrics en Enginyeria, C/Gran Capità s/n, Barcelona 08034, Catalonia, Spain*

e-mail: engwall@kth.se, jbaiges@cimne.upc.edu, saeed@kth.se, ncde@kth.se, ekeberg@kth.se, jjan@kth.se, jhoffman@kth.se, selamt@kth.se, stern@kth.se

ABSTRACT

This user guide describes step-by-step how to install and use the software required to run EUNISON simulations as an end user, namely ArtiSynth, and FEniCS and/or FEMUSS. ArtiSynth, with EUNISON-specific additions, is the remote user interface in this chain, as it can be run on the user's own PC to control the biomechanical models of the vocal tract and vocal folds, and to export the vocal tract tube and vocal fold configurations for FSI simulations on a supercomputer. An automatic cavity extraction tool creates regular vocal tract meshes from the geometric configuration of the ArtiSynth models. These vocal tract meshes can be used directly for airflow and acoustic simulations in FEniCS or alternatively in FEMUSS. The exported vocal fold meshes need to be pre-processed in a meshing tool to improve the mesh quality before they can be submitted to FSI simulations, due to the complexity of the laryngeal structures. EUNISON experiments have been made using the commercial software ANSA for this, but open software, such as gmsh, can also be used.

The user guide further gives several illustrated examples on the different steps involved in the simulation chain.

Version	V1
Date	2016-05-25
WP number	7
WP leader	Olov Engwall
WP leader email	engwall@kth.se

Contents

1. Introduction

The EUNISON software comprises three different toolsets, serving different needs in the quest for voice simulation. ArtiSynth, with add-ons provided by EUNISON, is used as the **remote user interface** for the high-performance computing aero-acoustic simulations carried out in either FEniCS or FEMUSS. Remote user interface here signifies that ArtiSynth may be run on the user's own personal computer and that the output from ArtiSynth is designed to be used as input to simulations in FEniCS or FEMUSS, which are run on a remote HPC network, after user login. ArtiSynth is an open-source Java-based framework for 3D biomechanical modeling, in which the muscle activation and/or articulatory setting of the jaw, tongue and larynx can be defined. The output from these simulations are time sequences of geometries for the vocal tract and surface meshes of the vocal folds. After pre-processing, surface meshes suitable for FSI simulations are created, and these can then be used by either FEniCS or FEMUSS, where airflow and acoustic simulations can be performed. The ArtiSynth component can be used on a high-end, fast personal computer, while both FEniCS and FEMUSS require high performance supercomputers to run any full-scale models. The installation and use of ArtiSynth, FEniCS and FEMUSS, together with illustrated EUNISON examples, are described in Sections 2, 3 and 4, respectively.

Figure 1 illustrates the chain of steps involved in the process of controlling the vocal tract and vocal folds models with ArtiSynth and submitting the resulting surface meshes for simulations in FEniCS or alternatively in FEMUSS. This user guide gives introductory, EUNISON specific, examples of use of the three toolsets. For general and more in-depth instructions on the use of the tool sets, please refer to the respective manuals, available from the dedicated project pages of ArtiSynth, FEniCS and FEMUSS (addresses provided in Section 5).

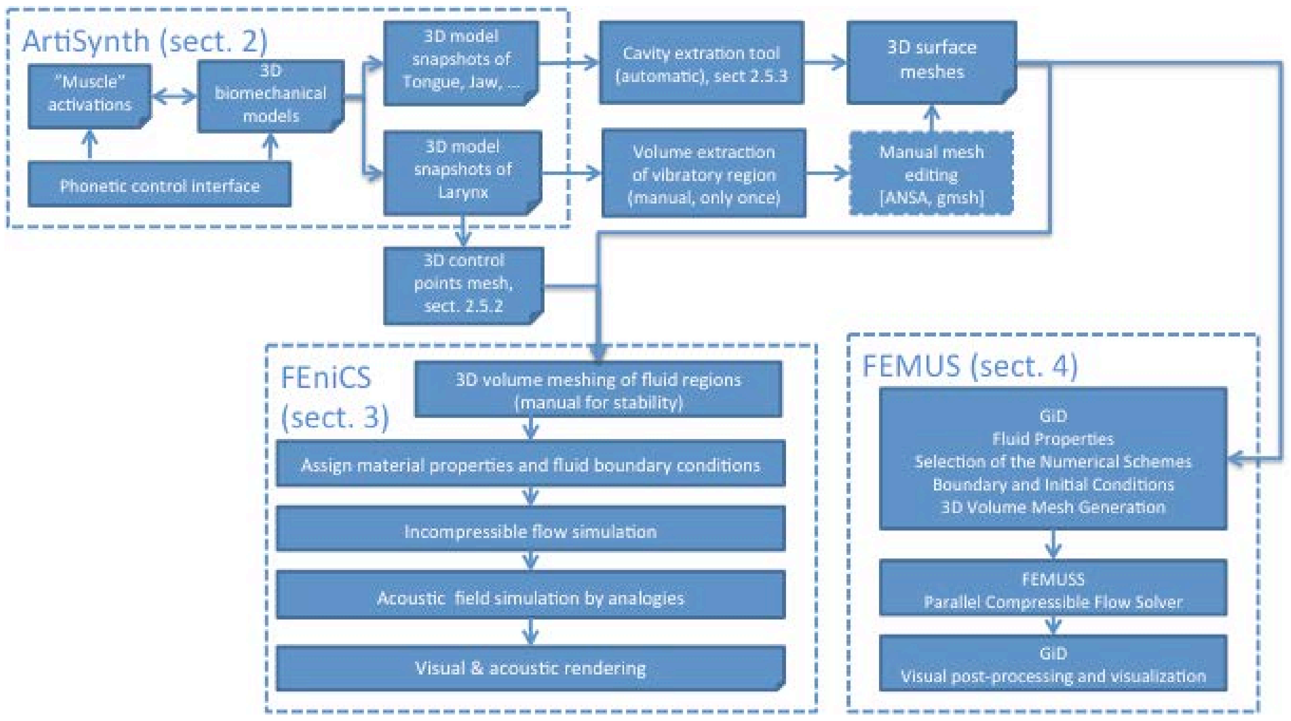

ArtiSynth (sect. 2)
"Muscle" activations
3D biomechanical models
Phonetic control interface
3D model snapshots of Tongue, Jaw, ...
3D model snapshots of Larynx
Cavity extration tool (automatic), sect 2.5.3
3D surface meshes
Volume extraction of vibratory region (manual, only once)
Manual mesh editing [ANSA, gmsh]
3D control points mesh, sect. 2.5.2
FEniCS (sect. 3)
3D volume meshing of fluid regions (manual for stability)
Assign material properties and fluid boundary conditions
Incompressible flow simulation
Acoustic field simulation by analogies
Visual & acoustic rendering
FEMUS (sect. 4)
GiD
Fluid Properties
Selection of the Numerical Schemes
Boundary and Initial Conditions
3D Volume Mesh Generation
FEMUSS
Parallel Compressible Flow Solver
GiD
Visual post-processing and visualization

Figure 1. Sequence of steps of EUNISON simulations when using ArtiSynth in combination with FEniCS or FEMUSS.

2. ArtiSynth

This section explains how to install ArtiSynth and use EUNISON-specific add-ons to run biomechanical simulations and export meshes that after further processing can be used in FEniCS and FEMUSS for simulating fluid dynamical and acoustic phenomena, e.g., the self-oscillation of the vocal folds and the acoustic propagation of waves in the vocal tract.

2.1 Platform

ArtiSynth is Java-based and is therefore in principle independent of the computer platform. In EUNISON it has been run on Windows, Mac OS and Linux. The cavity extraction tool (CET, described in Section 2.4.2) executable was compiled for Windows.

2.2 Requirements

The installation of ArtiSynth requires previous installation of an Integrated Development Environment (IDE) for Java programming, such as Eclipse. This can be done from https://eclipse.org, following the instructions provided there.

In order to run the cavity extraction tool executable CET, the freely available MATLAB Compiler Runtime (MCR) needs to be installed. This can be done by downloading MCR, version R2014a (8.3) from Mathworks at http://www.mathworks.com/products/compiler/mcr/index.html and installing it. We have worked with MATLAB Runtime Version R2014a (8.3) and therefore recommend using this version.

2.3 Installing ArtiSynth

In order to use ArtiSynth for EUNISON-style simulations, you need to install both the ArtiSynth *Core* package and the *Models* package.

2.3.1 Installing ArtiSynth proper

Step 1. Download and install the *Core* package

Refer to http://artisynth.magic.ubc.ca/artisynth/pmwiki.php?n=Documentation.InstallGuide for download and installation instruction. Note that an IDE for Java programming is required (c.f. Section 2.1.2).

Step 2. Request access to the ArtiSynth *Models* Package

Fill in the form on: http://artisynth.magic.ubc.ca/artisynth/pmwiki.php?n=Models.Models

Step 3. Download and install the source code for the *Models*

Follow the instructions provided after filling in the ArtiSynth *Models* form. The address of the *Models* SVN repository is https://svn.artisynth.org/svn/artisynth_models/trunk.

(Step 4. [For the Larynx] Request access to the ArtiSynth *Project* Package – if required)

At the time of writing, the *MoisikEunison* larynx model is included in the *Projects* package of ArtiSynth, rather than the *Models* Package, but Scott Moisik and the ArtiSynth team plan to release the underlying models imminently, and all necessary models will thereafter be placed in the *Models* package. Should access to the *Projects* package still be required at the time of reading, this can achieved by sending an e-mail to the ArtiSynth team (artisynth@ece.ubc.ca) in order to receive the required username/password credentials. The procedure of installation is the same as in the case of *Models*, the only change is the SVN repository address which in the case of *Projects* is https://svn.artisynth.org/svn/artisynth_projects/trunk. Alternatively, the *MoisikEUNISON* model is available in the EUNISON software release bundle.

2.3.2 *Installing EUNISON-specific additional parts of ArtiSynth*

Step 1. Download the ATDandCPC.zip file from the EUNISON website

Step 2. Unzip the file ATDandCPC.zip and include the ATD folder in ArtiSynth

Place the ATD folder in /artisynth_core/src/artisynth/core/
The *Artisynth-To-Dolfin (ATD)* package is used to extract and export surface meshes of the laryngeal structures.

Step 3. Place java-files in the ArtiSynth core

Copy the files `MainFrame.java` and `MenuBarHandler.java` to /artisynth_core/src/artisynth/core/driver/

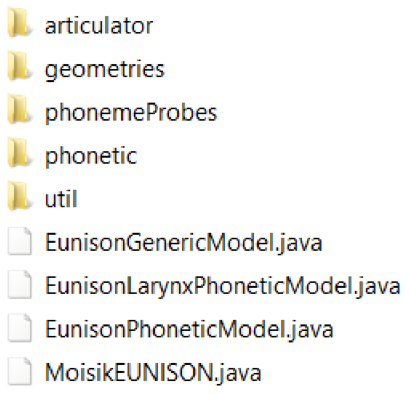
Step 4. Download the cavity extraction tool CET from the EUNISON website[1]

Step 5. Install the cavity extraction code

Follow the included installation wizard.

Step 6. Copy Phonetic Interface files into ArtiSynth models

To use the EUNISON package with ArtiSynth, the Phonetic Interface structure (below) should be copied to INSTALL_DIR/artisynth/models/kth/, where INSTALL_DIR is installation directory of ArtiSynth.


articulator
geometries
phonemeProbes
phonetic
util
EunisonGenericModel.java
EunisonLarynxPhoneticModel.java
EunisonPhoneticModel.java
MoisikEUNISON.java

[1] The cavity extraction tool CET will be realeased as open source on the EUNISON website when the scientific article describing the method has been published. Prior to this, CET is available in the EUNISON software release bundle.

2.4 EUNISON models

2.4.1 *EunisonGenericModel*

The *EunisonGenericModel* copied in Step 6 has the following components:

- *articulator*: folder including several files to define a data structure that has been designed to efficiently represent articulators and their muscles, and markers (EMA coils).
- *geometries*: folder containing all geometries used in EunisonGenericModel.
- *phonemeProbes*: folder storing all data representing muscle activation patterns for phonemes defined in the phonetic interface.
- *phonetic*: folder containing all java files necessary for the phonetic interface.
- *util*: additional utility functions specifically developed for EUNISON.
- *EunisonGenericModel*: This class is based on two existing models in ArtiSynth, *BadinJawHyoidTongue* and *FrankModel2*. The first model lacks the pharynx, larynx and the soft palate, while the latter is a comprehensive orofacial model with a large amount of FEM elements, making it extremely slow for simulations. *EunisonGenericModel* therefore complements *BadinJawHyoidTongue* with the pharynx, larynx, and soft palate from *FrankModel2*, but represented as skin surface meshes, which speeds up the simulations significantly. The pharynx, larynx and soft palate models have further been adapted to improve acoustic simulation results.

EunisonPhoneticModel: This class implements the phonetic interface. It is an extension to EunisonGenericModel showing how users can add the phonetic interface to a model.

EunisonLarynxPhoneticModel: This class combines the phonetic interface to control the larynx in the *MoisikEUNISON* model.

2.4.2 *MoisikEUNISON*

For simulations of laryngeal posturing and vocal fold vibration, *MoisikEUNISON* is used, which is a variant of Scott Moisik's model of the larynx (Moisik & Esling, 2014), including detailed laryngeal cartilages, maxilla and mandible, folds, extrinsic and intrinsic laryngeal muscles. Since the model includes 10 rigid bodies and over 80 different muscles it is computationally heavy to run together with the upper vocal tract structures in *EunisonGenericModel* on a personal computer, and the two models are therefore controlled separately in the user interface and the output meshes from the vocal tract and vocal folds are subsequently merged for FSI simulations.

2.5 Using ArtiSynth for EUNISON simulations

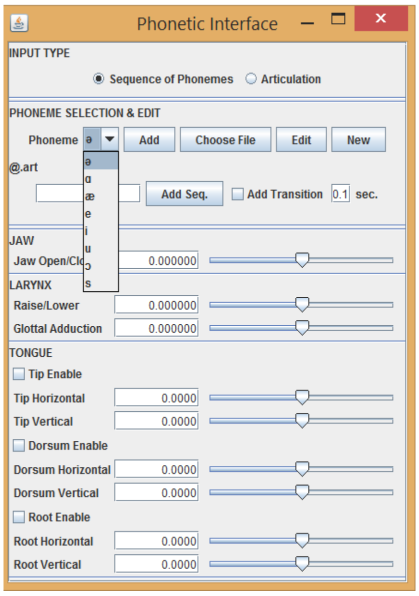
2.5.1 *Procedure*

The procedure for using ArtiSynth for EUNISON simulations is summarized in Figure 2 and the step-by-step description, below. Users interact with the phonetic interface, thereby creating muscle activation tracks that control the biomechanical models. The phonetic interface and the control of the biomechanical models are described further in deliverables D7.3 and D7.2, respectively. Here, we give only the information that is necessary to run the simulations. Through a cavity extraction tool CET, a vocal tract tube mesh is created from the anatomical structures bounding the airway. This surface mesh can be used for FSI simulations in FEniCS (c.f. Section 3) or FEMUSS (c.f. Section 4) after assigning material properties and fluid boundary conditions.

Step 1. Phonetic or Articulatory control

Step 2. Check Muscle excitations

Step 3. Add MeshProbe for mesh exportation


Phonetic Interface

INPUT TYPE

[✓ Sequence of Phonemes] Articulation

PHONEME SELECTION & EDIT

Phoneme: ə

Add | Choose File | Edit | New

@.art

Add Seq. | Add Transition: 0.1 sec.

JAW

Jaw Open/Close: 0.000000

LARYNX

Raise/Lower: 0.000000

Glottal Adduction: 0.000000

TONGUE

Tip Enable

Tip Horizontal: 0.0000

Tip Vertical: 0.0000

Dorsum Enable

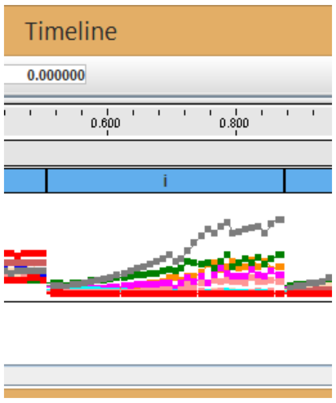
Dorsum Horizontal: 0.0000

Dorsum Vertical: 0.0000

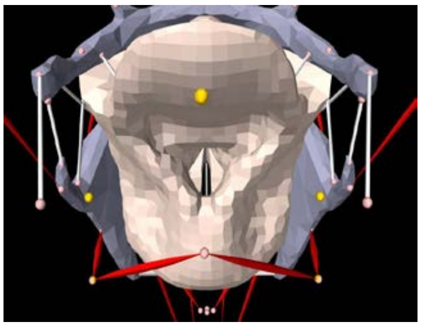
Root Enable

Root Horizontal: 0.0000

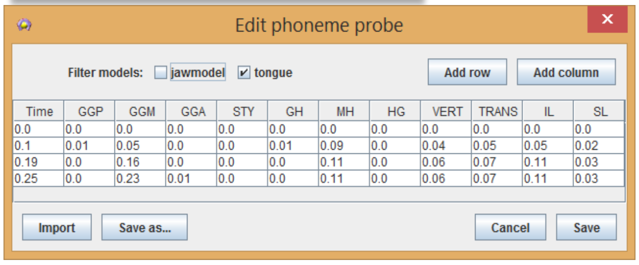
Root Vertical: 0.0000



Step 4. Run simulation to generate articulatory configuration



Step 1(b). Edit or define New muscle activation pattern.

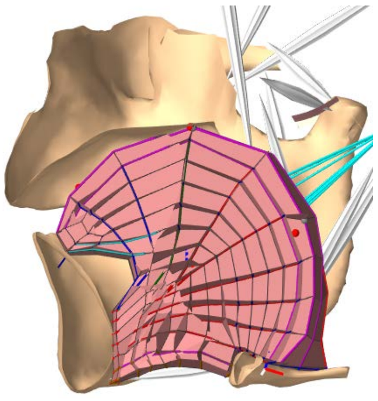

Edit phoneme probe

Filter models: jawmodel [✓ tongue]

Add row | Add column

Time	GGP	GGM	GGA	STY	GH	MH	HG	VERT	TRANS	IL	SL
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.01	0.05	0.0	0.0	0.01	0.09	0.0	0.04	0.05	0.05	0.02
0.19	0.0	0.16	0.0	0.0	0.0	0.11	0.0	0.06	0.07	0.11	0.03
0.25	0.0	0.23	0.01	0.0	0.0	0.11	0.0	0.06	0.07	0.11	0.03

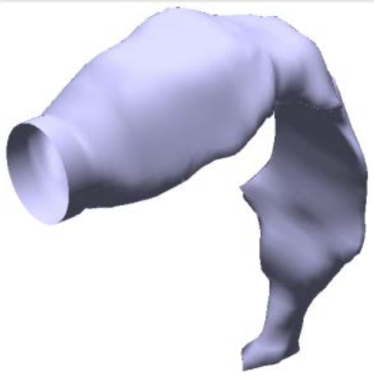
Import | Save as... | Cancel | Save

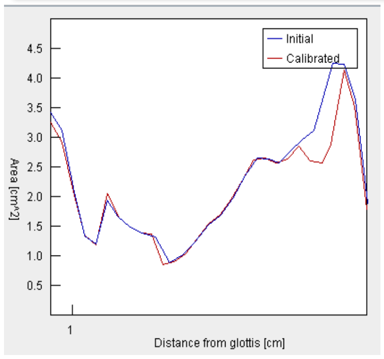


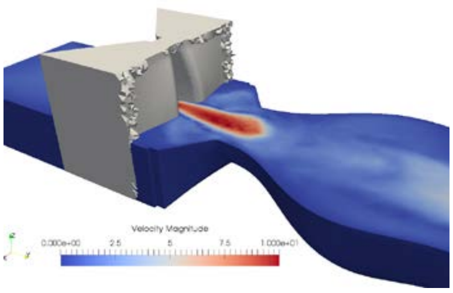
Step 5. Run cavity extraction algorithm (c.f. Section 2.3.2)

Step 6 (option) Area function calculation & visualization

Step 8. Export meshes for 3D FEM FSI simulations







Step 7 (option) Acoustic preview with 1D synthesis

Figure 2. Sequence of steps when using ArtiSynth for EUNISON simulations.

EUNISON is supported by the Future and Emerging Technologies (FET) programme within the ICT theme of the Seventh Framework Programme for Research of the European Commission

Step 1. Define muscle activations using the phonetic interface

(a) [For the tongue and jaw] Select a sequence of phonemes from the drop-down menu in PHONEME SELECTION & EDIT and press Add to activate the corresponding muscle excitations.

(b) [For the tongue and jaw] Edit an existing muscle activation pattern by pressing Edit or define a new pattern by pressing New in PHONEME SELECTION & EDIT. A phoneme probe interface will appear, allowing to set activation for the muscles of the model at different instances in time.

(c) Select vertical and horizontal positions of the target points of the JAW, LARYNX and TONGUE.
Note that for computational reasons, the larynx model is handled separately from the vocal tract model, and the *MoisikEUNISON* larynx model hence needs to be loaded to control the larynx.

(d) [For the larynx] load the appropriate muscle activations using Input probes, accessed through [File → Load probes…] and choose
AdductionAbductionProbes.txt for a gesture of adduction/abduction of the vocal folds
AdductionAndPitchModificationProbes.txt for adduction and pitch manipulation.
Set the time-step of the simulation to 0.001 (sec) by typing the value in the "step" box.

Step 2. Display, and edit, muscle activations

By selecting [View → Show timeline], the Muscle Activations for the chosen gesture appear in the ArtiSynth Timeline window, where the muscle activations may be edited.

Step 3. Add MeshOutputProbe to the model.

The probe enables exportation of meshes, as described further in Sections 2.5.2 and 2.5.3.

Step 4. Run simulation

Click Play in the Main ArtiSynth window to activate the biomechanical muscle models.

Step 5. Export meshes

(a) [For the vocal tract] Run the cavity extraction tool CET that creates vocal tract tube meshes from the extracted surface meshes of the surrounding articulators, as described further in Section 2.5.3.

(b) [For the vocal folds] Select compilation ControlPoint file, as described further in Section 2.5.2.

Step 6 (optional). Area function calculations

As an alternative, or pre-check step, to 3D FSI simulations, the vocal tract tube may also be used to calculated area functions.

Step 7 (optional). 1D acoustic synthesis preview

Acoustic effects of changes in user settings can be previewed using, a one-dimensional speech synthesizer taking the calculated area function as input, before the vocal tract model is submitted to FSI simulations. Available alternative that have been tested in EUNISON are the java-based JASS, which is connected to ArtiSynth, and VTAR, a Matlab-based tool available from http://www.isr.umd.edu/Labs/SCL/vtar/ or the Mathworks file exchange central.

Step 8. Submit vocal tract tube meshes for aero-acoustic simulations

The created vocal tract tube and vocal folds meshes may be used for simulations in FEniCS or FEMUSS, as described in Sections 3 and 4.

2.5.2 *Exporting vocal folds meshes*

The dynamic extraction of the surface mesh (Step 3 above) of the larynx is performed by the EUNISON contributed *Artisynth-To-Dolfin (ATD)* package in ArtiSynth, with DOLFIN .xml files as output. DOLFIN is the computational back-end of FEniCS, and the DOLFIN .xml format is chosen in order for the meshes to be directly usable in FEniCS simulations. The two extraction procedures of the vocal tract and vocal folds could in principle be executed simultaneously, but for computational efficiency they are executed separately.

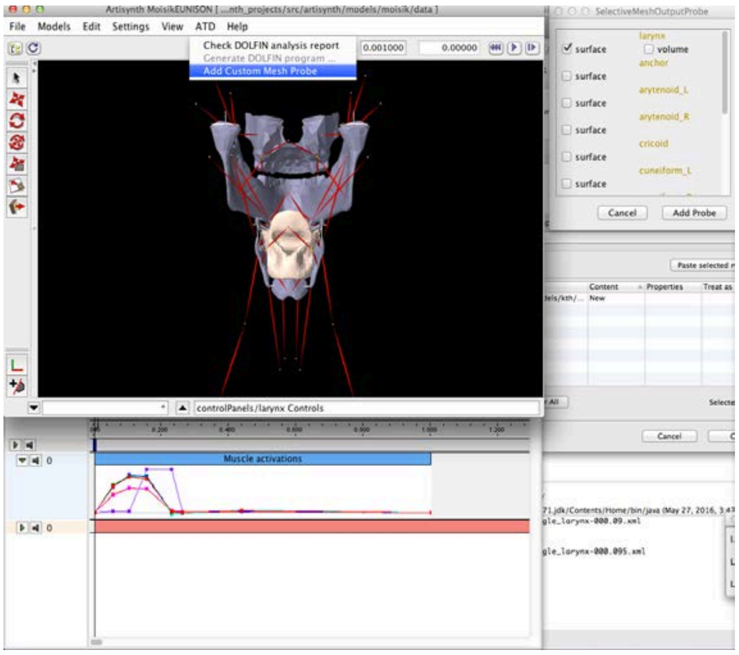
The ATD package can be used with any ArtiSynth model, but we here exemplify with the *MoisikEUNISON* model. The package extracts surface or volume meshes in every time step of an ArtiSynth simulation, by inserting a custom-made output probe (*Selective mesh output probe*). The meshes are exported to the folder `/artisynth_core/selective_mesh_probe_output` with the filename format *Elementtype_Name-Time.xml* where:

Elementtype is mesh element type (triangle for surfaces and tetrahedron for volume meshes).
Name is the ArtiSynth name of the mesh object.
Time is the time in seconds when the snapshot of the mesh was taken.

This means that e.g., the file `triangle_larynx-000.07.xml` corresponds to a snapshot of the surface mesh of the larynx at 0.07 seconds.

The ATD package is accessed through the *ATD* Menu bar of ArtiSynth, and then selecting *Add Custom Mesh Probe*. A popup window named *SelectiveMeshOutputProbe* appears, where the user can select which meshes are to be dynamically exported. This procedure is illustrated to the right.

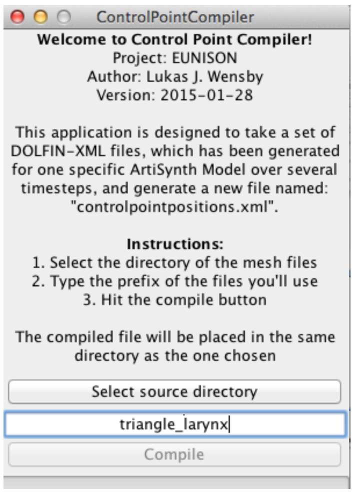
In the case of the *MoisikEUNISON* model, the surface mesh of the larynx is chosen for extraction. In order to dynamically extract the larynx mesh, we select *surface* for the larynx mesh in the *SelectiveMeshOutputProbe* window (accessed as described above).



For step 5 above, the *ControlPointCompiler.jar* application is used to compile the separate .xml files into one Control Point File.

The user should select the folder where the probe files are placed with *Select source directory* and then specifying the desired filename before compiling.

The resulting ControlPointPositions.xml file is placed in the same folder as the input mesh files and may be directly used to control the motion of the larynx mesh for the simulations in FEniCS.


ControlPointCompiler

Welcome to Control Point Compiler!
Project: EUNISON
Author: Lukas J. Wensby
Version: 2015-01-28

This application is designed to take a set of DOLFIN-XML files, which has been generated for one specific ArtiSynth Model over several timesteps, and generate a new file named: "controlpointpositions.xml".

Instructions:
1. Select the directory of the mesh files
2. Type the prefix of the files you'll use
3. Hit the compile button

The compiled file will be placed in the same directory as the one chosen

Select source directory
triangle_larynx
Compile

2.5.3 Exporting vocal tract meshes

Export of vocal tract meshes is based on Steps 3 and 5 above. MeshOutputProbe is an extension of OutputProbe, which supports export of mesh components as wavefront (.obj) file format. Each component needs an individual probe, so as to export both tongue and jaw geometries, two probes are needed. The user can set the time step and the folder to which the files will be exported, as shown in the example below, where one tongue surface mesh is exported every 10 ms between 150 ms and 1 s.

```
MeshOutputProbe probe = new MeshOutputProbe();
probe.setName("tongue surface mesh");
probe.setMesh(tongue.getSurfaceMesh ());
probe.setStartTime(0.15);
probe.setStopTime(1.0);
probe.setUpdateInterval(0.01);
probe.setPath ("C:\geom\"); // Use this folder to export files
probe.setFile("tongue","%d","ms"); //how to name files, e.g. tongue250ms.obj
probe.setModel(this);
probe.setActive(true);  // default to inactive
probe.setRigidBody(true);
 addOutputProbe(probe);
```

After running the ArtiSynth simulations, run the cavity extraction tool from command window in the folder in which the bounding geometry meshes are placed (`C:\geom\` for the example above):

```
> ce.exe
```

The tool assumes that the surrounding geometries are in the current folder. If another folder should be used, this may be set by the address of geometries, e.g., `add = 'C:\geom\'`.

The following default settings are used:

Static geometries: `staticMeshesName={'maxilla';'softPalate';'pharynx';'larynx'};`

Scale of static geometries: `staticMeshesScale = [1 1 1 1];`

Dynamic geometries: `dynamicMeshesName = {'jaw';'tongue'};`

Scale of dynamic geometries: `dynamicMeshesScale = [1 1];`

Number of planes in a semi-polar grid: `planeNum = [20 30 20];`

Resolution of geometries: `resolution = 1e-3;`

The division of geometries into static and dynamic helps to decrease the number of required computations, since the intersection of the static geometries with the semi-polar grid used for cavity extraction can be performed just once at the beginning, while it needs to be repeated at every time step for the dynamic geometries. *Scale* is used to scale the corresponding geometries (if this is required e.g., for speaker adaptation purposes) before any processing and may be used if the units of the models are in meters or millimetres. The semi-polar grid is defined by the number of planes in the horizontal, polar and vertical sections, given in the *planeNum* vector. Resolution is a parameter used in geometrical processing of the vocal tract boundary to identify if two points or vertices should be considered as coinciding and be merged.

3. FEniCS

We here describe the FEniCS-HPC EUNISON distribution v0.1, which is the software distribution of the FEniCS-HPC branch in the FEniCS project (http://fenicsproject.org) for the EUNISON project. The distribution includes automatic installation scripts for the Beskow supercomputer at KTH, which should be easily modifiable for other environments.

3.1 Installing FEniCS

3.1.1 Platform

Running FEniCS requires access to HPC hardware. The instructions below assume that the Beskow supercomputer at KTH is used.

3.1.2 Requirements

In order to visualize the output from the simulations, the open-source, multi-platform data analysis and visualization application ParaView is needed. ParaView can be downloaded from http://www.paraview.org, and installed using the instructions provided with the application.

3.1.3 Installation of FEniCS proper

As installation scripts have been created, the installation procedure of FEniCS is very simple:

Step 1. Login to Beskow

Refer to https://www.pdc.kth.se/resources/software/login-1 for instructions

Step 2. Go to the fenics-hpc distribution directory

```
cd fenics-hpc.dist-*
```

Step 3. Build

```
source build_beskow.sh
```

use: `source init_beskow.sh` for just setting up the environment

Step 4. Basic test

In order to perform a basic test of the installation, the following sequence can be run:

Step 4-1. Change to the test directory

```
cd unicorn-minimal
```

Step 4-2. Parallel build with 4 processes

```
make -j 4
```

Step 4-3. Change to the simulation directory

```
cd sim01
```

Step 4-4. Copy the just-compiled program

```
cp ../unicorn .
```

Step 4-5. Run the adaptive simulation

```
sbatch job-adaptive.script
```

Step 4-6. extract and print the key output values of the simulation

```
sh output.sh
```

Step 4-7. Visualize in ParaView

copy the *.pvd, *.pvtu and *.vtu files to your local computer and open the *.pvd files with ParaView.

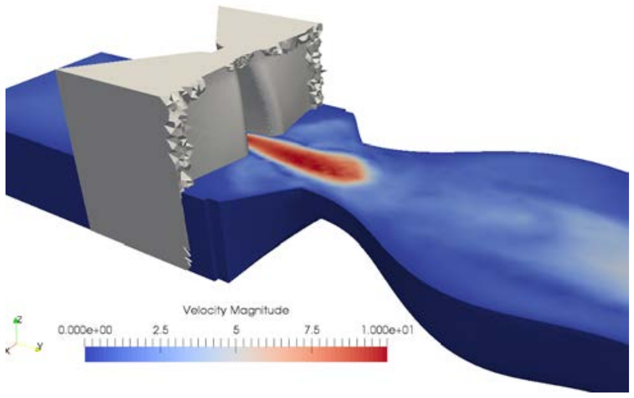
3.2 Using FEniCS for EUNISON simulations

FEniCS can take the vocal tract tube surface meshes and the vocal fold control point meshes as input and perform incompressible flow simulations after having created 3D volume meshes of the structures and assigning material properties. At the time of writing, manual adaptation of the 3D volume meshing of the vocal folds region is still required for stability, due to the complexity of the geometries.

We here exemplify the use of FEniCS with three different cases, with increasing complexity. First the case where the flow velocity and pressure through simplified vocal folds modelled as rubber blocks is demonstrated, then corresponding flow velocity through an anatomically more realistic model, and finally the further propagation of the air pressure wave through the vocal tract. These cases have been studied within the EUNISON project and are described further from a scientific point of view in deliverables D2.1, D2.3 and D2.4.

3.2.1 Vocal folds simplified geometry test

Following the below steps will display the test case with numerical simulations of the airflow through the Erlangen vocal fold replica, as described in EUNISON deliverable D2.1:

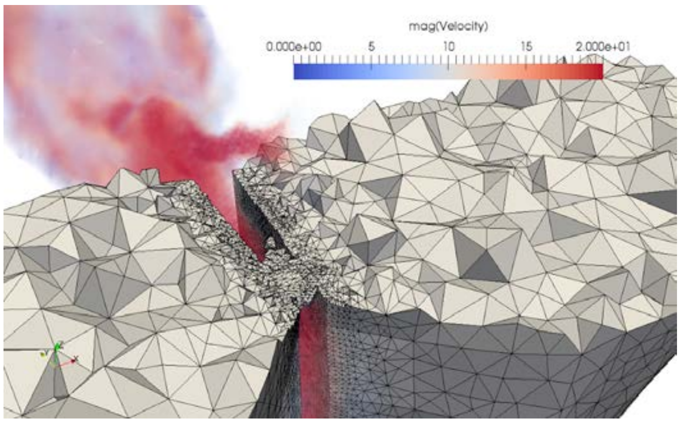


Step 1. Change to the example directory
```
cd unicorn-eunison-vf-simple
```
Step 2. Parallel build with 4 processes
```
make -j 4
```
Step 3. Change to the simulation directory
```
cd sim01
```
Step 4. Copy the just-compiled program
```
cp ../unicorn .
```
Step 5. Run the adaptive simulation
```
sbatch job-eunison.script
```

Step 6. Visualize in ParaView
copy the *.pvd, *.pvtu and *.vtu files to the local computer and open the *.pvd files with ParaView.

3.2.2 EUNISON vocal folds realistic geometry test

The second example will illustrate airflow velocity through the realistic vocal folds model described in Sections 2.4.2 and 2.5, after the pre-processing indicated by Figure 1.



Step 1. Change to the example directory
```
cd unicorn-eunison-vf-real
```
Step 2. Parallel build with 4 processes
```
make -j 4
```
Step 3. Change to the simulation directory
```
cd sim01
```
Step 4. Copy the just-compiled program
```
cp ../unicorn .
```
Step 5. Run the adaptive simulation
```
sbatch job-eunison.script
```

Step 6. Visualize in ParaView
copy the *.pvd, *.pvtu and *.vtu files to the local computer and open the *.pvd files with ParaView.

3.2.3 EUNISON vocal folds and aero-acoustic analogy simplified geometry test

The third example combines the simplified vocal folds and a simplified, straight vocal tract to investigate the velocity and pressure for the aero-acoustic flow in the vocal tract tube.

Step 1. Change to the example directory

```
cd unicorn-eunison-vf-acoustics
```

Step 2. Parallel build with 4 processes

```
make -j 4
```

Step 3. Change to the simulation directory

```
cd sim01
```

Step 4. Copy the just-compiled program

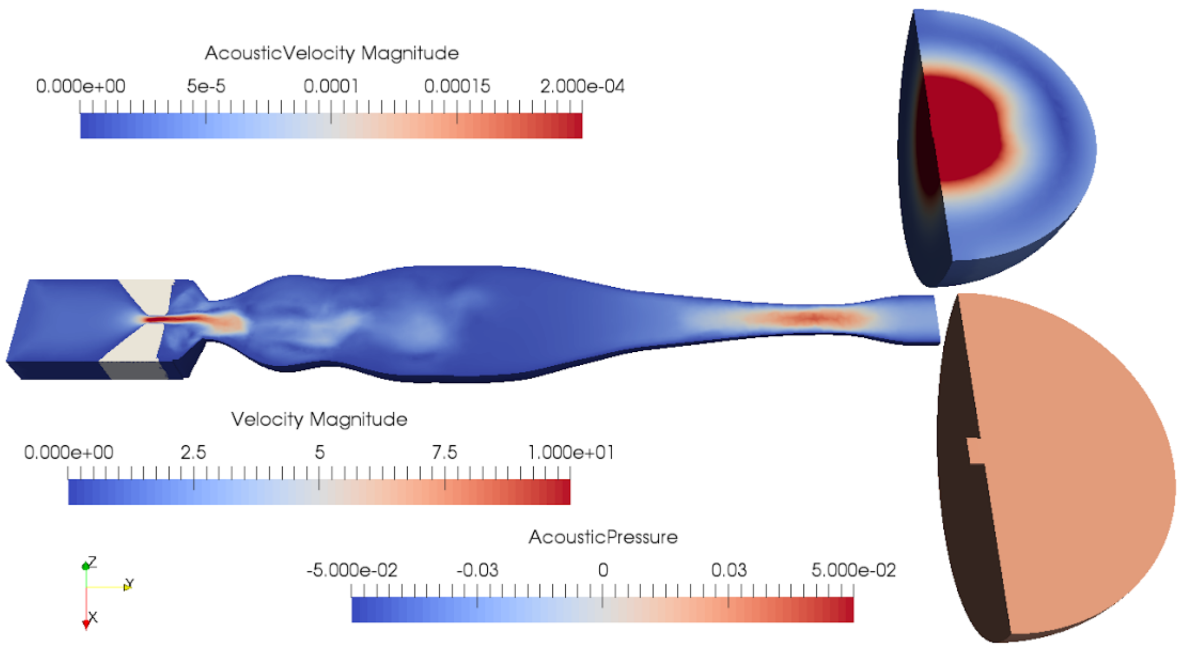
```
cp ../unicorn .
```

Step 5. Run the adaptive simulation

```
sbatch job-eunison.script
```

Step 6. Visualize in ParaView

copy the *.pvd, *.pvtu and *.vtu files to your local computer and open the *.pvd files with ParaView.



4. FEMUSS

This section describes the FEMUSS compressible flow solver with ALE formulation, which may be used as an alternative to FEniCS for the simulations. As for FEniCS, vocal tract surface meshes are used as input

4.1 Installing FEMUSS

Two versions of the code must always be compiled; *Debug version*, which enables tracking runtime code functionalities and *Release version*, which runs without displaying runtime information. GNU compilers can be used to compile FEMUSS code and the required software (OpenMPI and PETSc).

4.1.1 Platform

FEMUSS is supported in LINUX systems and has been tested on UBUNTU 12.04 LTS. Intel compilers required are not available for newer versions of Ubuntu.

4.1.2 Requirements

In order to install the FEMUSS solver package, the following packages need to be installed on the Linux Machine:

1. GNU compilers for C, C++ and FORTRAN. 4.9.3. Latest stable
2. PYTHON 2.7. Latest stable, ubuntu build in.
3. CMAKE 3.4.1. Latest stable.
4. OpenMPI 1.10.0. Latest stable.
5. PETSc 3.5.4. Latest stable.
6. GID 12.0.5. Latest stable.
7. VTK 7.0.0. Latest stable. (Optional)

4.1.3 Step-by-step installation

Step 1. Install GNU compilers

```
sudo apt-get install gcc
sudo apt-get install g++
sudo apt-get install gfortran
sudo apt-get install gcc-multilib
```

Step 2. Update .bashrc

Edit the specified paths of your .bashrc to add the directories where gidpost2.0, PETSc, FEMUSS and others, will be installed. This does not need to be inside the FEMUSS directory.

```
export PATH=/.../.../gid.../:$PATH
export GID_DIR='/.../.../gidpost2.0/unix'
export PETSC_DIR='/.../.../petsc-3.4.3'
export VTK_DIR='/.../.../VTK-6.2.0'
export VTKPOST_DIR='/.../.../vtkpost'
```

Step 3. OpenMPI

Download OpenMPI from http://www.open-mpi.org/software/ompi/
Follow installation instructions provided with the software.

Step 4. PETSc

Download PETSc **Version 3.5.4 (do not download a newer or older version)** from http://www.mcs.anl.gov/petsc/download/index.html

Follow installation instructions provided with the software.

Step 5. GID

Download from http://www.gidhome.com/download
Uncompress the downloaded file and execute directly.

Step 6. GIDpost 2.0

Copy the folder gidpost2.0 provided with the sources of the FEMUSS code to your GID library directory (`GID_DIR` in your `bashrc`)
Run the Makefile in the unix folder

Step 7. VTK (Optional)

Download VTK-X.X.X.tar.gz from http://www.vtk.org/
Follow installation instructions provided with VTK.

Step 8. VTKpost (Optional)

Copy the folder vtkpost in the trunk folder of the FEMUSS Repository to your VTK library directory (VTK_DIR in your bashrc)
Compile the file using the Makefile: make

Step 9. Femuss.Gid (specific FEMUSS data for GID)

Copy the folder Femuss.Gid in the trunk folder of the FEMUSS Repository to the Gid problem type folder.

Step 10. FEMUSS

Open the terminal, change directory to Femuss/Executables/Unix in the trunk or in your branch folder and compile FEMUSS:

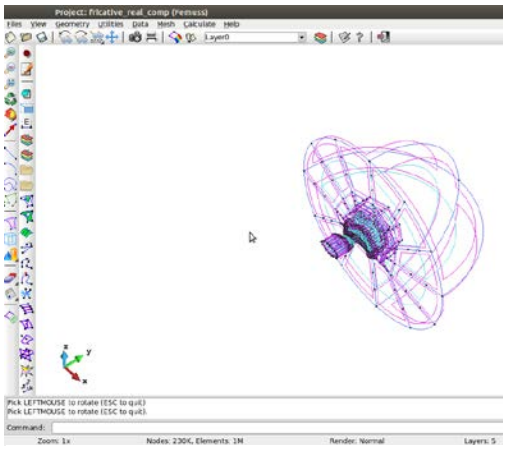
```
make -f make.PFemuss
```

4.2 Using FEMUSS for EUNISON simulations

4.2.1 GID Pre-processing step

Before geometries can be run in FEMUSS, they need to be pre-processed. We exemplify with two GiD files studied in EUNISON. We use the GiD interface for the generation of FEMUSS problem types.

The file *fricative_real_comp.gid*, corresponds to the geometry of the case with a compressible flow solver solution of the realistic fricative presented in EUNISON deliverable D4.3. Opening this file gives the set-up shown to the right.

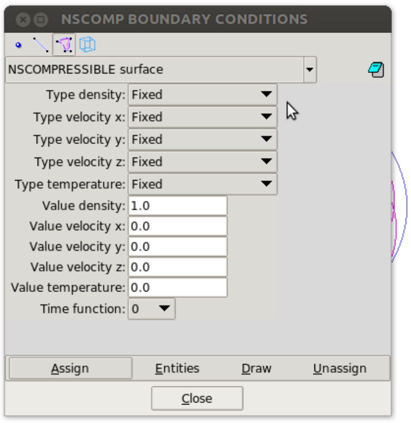

Project: fricative_real_comp (Femuss)
Files View Geometry Utilities Data Mesh Calculate Help
Layer0
z
y
x
Command:

```
Pick LEFTMOUSE to rotate (ESC to quit)
Pick LEFTMOUSE to rotate (ESC to quit).
Zoom: 1x    Nodes: 230K, Elements: 1M    Render: Normal    Layers: 5
```

Data → Conditions → NSCOMP_BOUNDARY_CONDITIONS

is used to modify boundary conditions for the compressible flow solver.

In this dialogue box, boundary conditions can be applied to each surface, including non-slip boundary conditions, inlet conditions and outlet conditions. Initial conditions can be prescribed by using the volume boundary conditions tab.



NSCOMPRESSIBLE surface

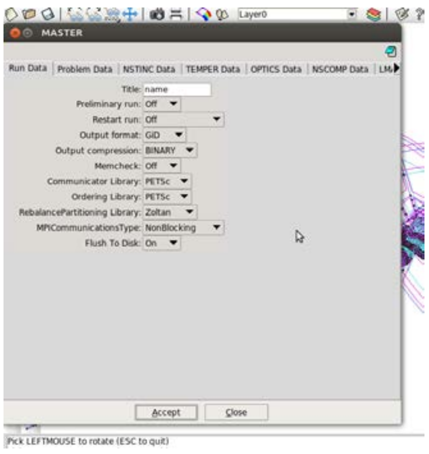
Type density:	Fixed
Type velocity x:	Fixed
Type velocity y:	Fixed
Type velocity z:	Fixed
Type temperature:	Fixed
Value density:	1.0
Value velocity x:	0.0
Value velocity y:	0.0
Value velocity z:	0.0
Value temperature:	0.0
Time function:	0

DATA → MASTER menu

is used to select the runtime, time step, and on modules. In the case of the compressible flow solver, only the NSCOMP module should be turned on. For example

- Run data: Preliminary run: off
- Output format: GiD BINARY
- Problem data: Time initial = 0.0
- Time final = 10.0
- Time step size = 0.1
- NSCOMP Data:

  NSCOMP problem: On
- NSCOMP algorithm: implicit
- All tother modules must be off.

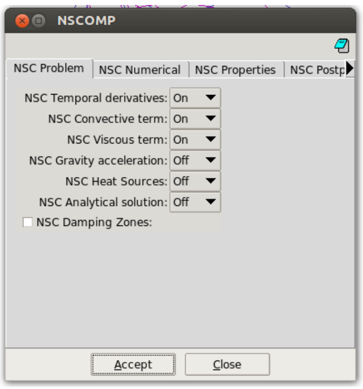


Title:	name
Preliminary run:	Off
Restart run:	Off
Output format:	GiD
Output compression:	BINARY
Memcheck:	Off
Communicator Library:	PETSc
Ordering Library:	PETSc
RebalancePartitioning Library:	Zoltan
MPICommunicationsType:	NonBlocking
Flush To Disk:	On

```
Pick LEFTMOUSE to rotate (ESC to quit)
```

DATA → NSCOMP menu

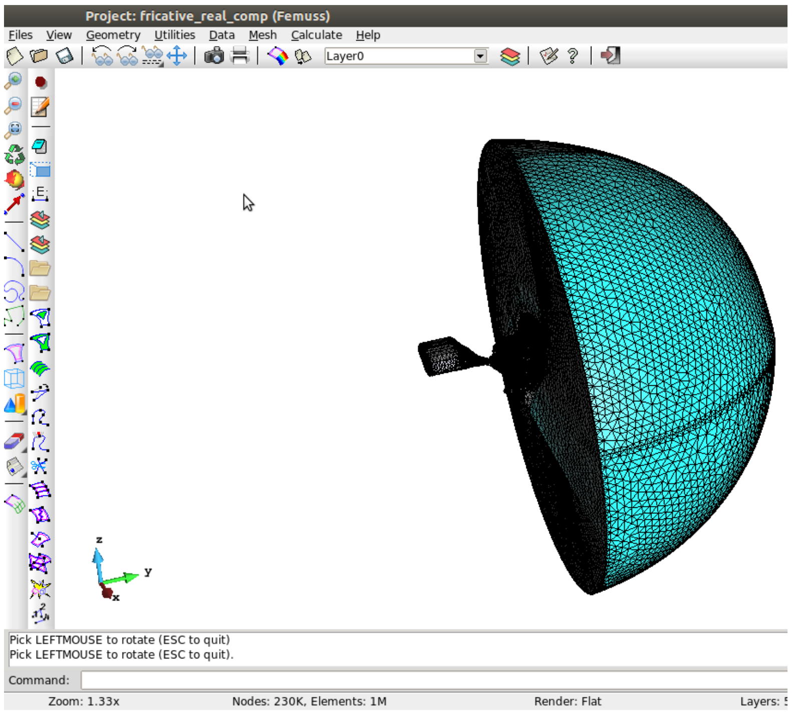
is used to set the parameters for the numerical simulation of the compressible flow and the fluid parameters. This menu also allows adjusting the post-processing options.


NSCOMP

NSC Problem | NSC Numerical | NSC Properties | NSC Postp

NSC Temporal derivatives: On
NSC Convective term: On
NSC Viscous term: On
NSC Gravity acceleration: Off
NSC Heat Sources: Off
NSC Analytical solution: Off
NSC Damping Zones:

The last step of the pre-processing is the Mesh generation, which is done by accessing the
Mesh menu,
where the size of elements may be adjusted (e.g., size: 0.1).


Project: fricative_real_comp (Femuss)

Pick LEFTMOUSE to rotate (ESC to quit)
Pick LEFTMOUSE to rotate (ESC to quit).

Command:

Zoom: 1.33x Nodes: 230K, Elements: 1M Render: Flat Layers: 5

Finally, pressing the Calculate → Calculate button generates the input files for FEMUSS.

4.2.2 Running the case with FEMUSS

Open a terminal in the folder of your problem (You must see the Data and Results folders). From there, launch FEMUSS using the environment variable or the route:

```
(path of the executable../../)Femuss/Executables/Unix/PFemuss.sh casename
```

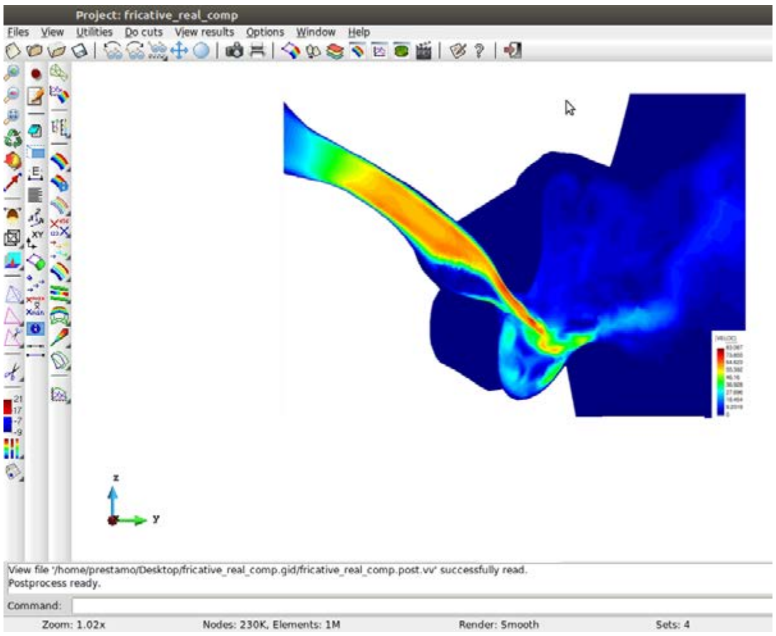
with different options for PFemuss.sh:

default:	Run in release mode.
-p:	Run in profile mode. Valid for Intel compilation of Femuss. A special version of Femuss should be compiled: mpintel; make -f make.PFemuss config profile
-pg:	Run in profile mode. Valid for GNU compilation of Femuss. A special version of Femuss should be compiled: mpignu; make -f make.PFemuss config profile Uses Gprof as profiling tool.
-g:	Run in debug mode.
-o:	Run in release mode.
-t:	Run in debug mode with Totalview.
-to:	Run in release mode with Totalview.
-v:	Run in debug mode with Valgrind.
-vo:	Run in release mode with Valgrind.
-np:	NUMBER. Run in parallel with (NUMEBER) processors.
-nomerge:	Avoids merging the results for plotting when running with several processors.
-mergeonly:	Merges the results for plotting when are obtained with several processors.

4.2.3 Post-process the results in GiD

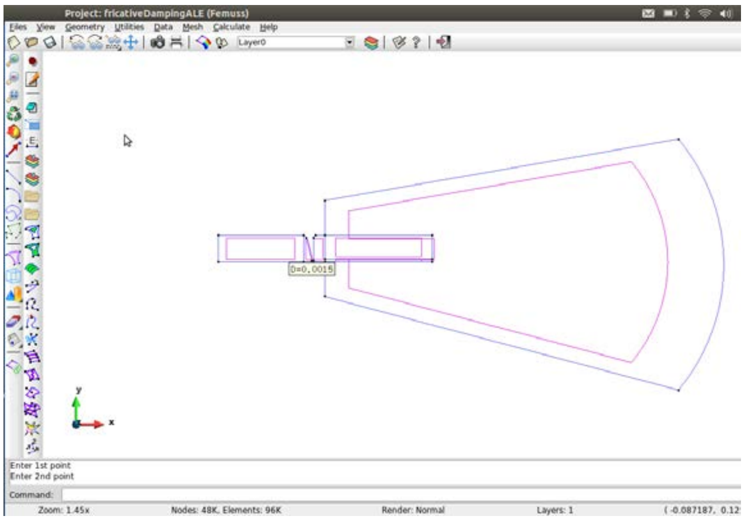
The results may be visualized and post-processed in GiD using
File→Postprocess
View Results → Contourfill → Veloc→ |Veloc|.

Pressure, plot velocity vectors, etc can also be visualized.

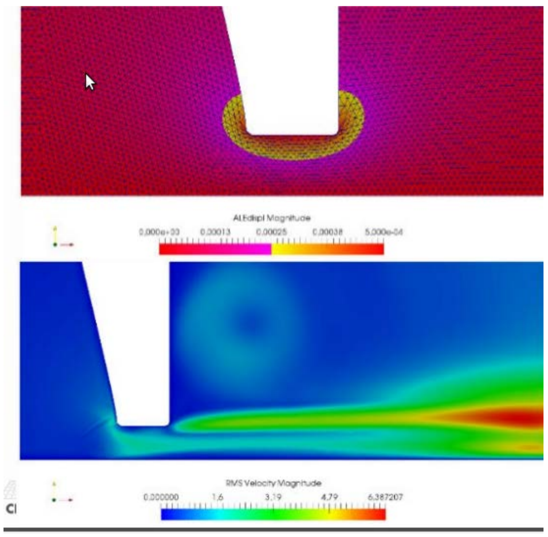

Project: fricative_real_comp
Files View Utilities Do cuts View results Options Window Help
z
y

View file '/home/prestamo/Desktop/fricative_real_comp.gid/fricative_real_comp.post.vv' successfully read.
Postprocess ready.
Command:
Zoom: 1.02x Nodes: 230K, Elements: 1M Render: Smooth Sets: 4

The above instructions can also be used to run or edit the additional EUNISON example case in the *fricativeDampingALE.gid* folder. This case corresponds to the simplified fricative with ALE movement presented in EUNISON deliverable D4.2. The geometry and the resulting flow visualisation are shown below.


Project: fricativeDampingALE (Femuss)
Files View Geometry Utilities Data Mesh Calculate Help
Layer0
D=0.0015
y
x

Enter 1st point
Enter 2nd point
Command:
Zoom: 1.45x Nodes: 48K, Elements: 96K Render: Normal Layers: 1 (-0.087187, 0.12



5. Resources and further reading

Further reading and resources

1. eunion.eu. EUNISON project homepage. Software release, instructions, updates.
2. http://www.artisynth.org. ArtiSynth introduction, download, instructions, examples
3. http://www.fenicsproject.org. FEniCS introduction, download, documentation
4. http://tts.cimne.com/RMOP. FEMUSS introduction, download, manuals
5. Moisik, S. R., & Esling, J. H. (2014). Modeling biomechanical influence of epilaryngeal stricture on the vocal folds: A low-dimensional model of vocal-ventricular coupling. Journal of Speech, Language, and Hearing Research, 57, S687-S704.
6. Deliverable D7.3 PHONETIC INTERFACE. Engwall, O., Dabbaghchian, S., Selamtzis, A., Ternström, S. & Ekeberg, Ö.
7. Deliverable D2.1 FLUID-STRUCTURE INTERACTION MODEL OF VOCAL FOLDS, Degirmenci, N.C., Jansson, J. & Hoffman, J.
8. Deliverable D2.3 INTERFACE TO CONTROL MODELS AND SIMULATION DATA
9. Deliverable D2.4 INCOMPRESSIBLE FLOW MODEL FOR FULL FLUID-STRUCTURE-ACOUSTIC COUPLING. Degirmenci, N.C., Jansson, J. & Hoffman, J., Sánchez-Martín, P., Arnela, M., Guasch, O., Espinoza, H., Codina, R., Selamtzis, A., Dabbaghchian, S., Engwall, O. & Ternström, S.
10. Deliverable D4.2 COMPRESSIBLE FLOW SOLVER, ALE AND SOFT TISSUE COUPLING STRATEGIES. Bayona, C., Pont, A., Baiges, J. & Codina, R.