

OSCILOS_{brass} Technical Report

Alexander MacLaren¹, Aimee S. Morgans and Renaud Gaudron²

Department of Mechanical Engineering, Imperial College London, UK

Version 2.0, 27/04/2021

OSCILOS_{brass} is an acoustic solver for predicting the natural modes of brass wind instruments from their geometry. It was created from the cold flow part of OSCILOS_{long}, an aeroacoustic solver designed to predict thermoacoustic instabilities in gas turbine engine combustors. OSCILOS_{long}, the Open Source Combustion Instability Low-Order Simulator (longitudinal), was developed by Prof. Aimee Morgans et al [1] at Imperial College London. OSCILOS_{brass} supports more flexible specification of geometry, several additional boundary conditions, and new output data formats compared to existing versions of OSCILOS, but the equations used by the solver are the same as those of OSCILOS_{long}.

The code is open source, written in MATLAB[®], and distributed under an open source license. It is highly modular, easy to use, and freely available from <https://www.oscilos.com>.

Contents

1	Getting Started	3
1.1	Running the Code	3
1.2	Quick Examples	4
2	Solver Model	10
2.1	Assumptions	10
2.2	Governing Equations	11
3	Inputs	17
3.1	Configuration	17
3.2	Geometry	26
4	Outputs	31
4.1	Initialisation	31
4.2	Results	35
5	Function Calls to OSCILOS_{brass}()	41
5.1	Function Definition	41
5.2	Code Example	41

¹dam216@imperial.ac.uk

²r.gaudron@imperial.ac.uk

Acknowledgement

This work was supported in part by the European Research Council (ERC) Consolidator Grant AFIRMATIVE (2018–2023). A. MacLaren acknowledges the Mechanical Engineering UROP Bursary 2020 from Imperial College London.

How to Cite OSCILOS_{brass}

When using OSCILOS_{brass}, please cite:

- MacLaren, A., Morgans, A. S. and Gaudron, R. (2021). OSCILOS_brass Technical report.
- MacLaren, A., Morgans, A. S. and Gaudron, R. (2021). OSCILOS_brass - An open-source acoustic solver for brass instruments. 27th International Congress on Sound and Vibration (ICSV27).

Open Source License

Copyright © 2021, Imperial College London

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1 Getting Started

OSCILOS_{brass} is written in MATLAB®, and is available from the OSCILOS website <https://www.oscilos.com>.

1.1 Running the Code

An active installation of MATLAB®, with the Optimisation toolbox installed, is required to run OSCILOS_{brass}. The code was developed on version R2020a.

The open-ended cylinder example in Sec. 1.2.1 runs out-of-the-box - simply run the MATLAB script `./OSCILOS_brass.m`. For the remainder of this document, `./` indicates the OSCILOS_{brass} working directory, e.g. `C:/Users/amacl/Downloads/OSCILOS_brass/`, whose contents are shown in Fig. 1.

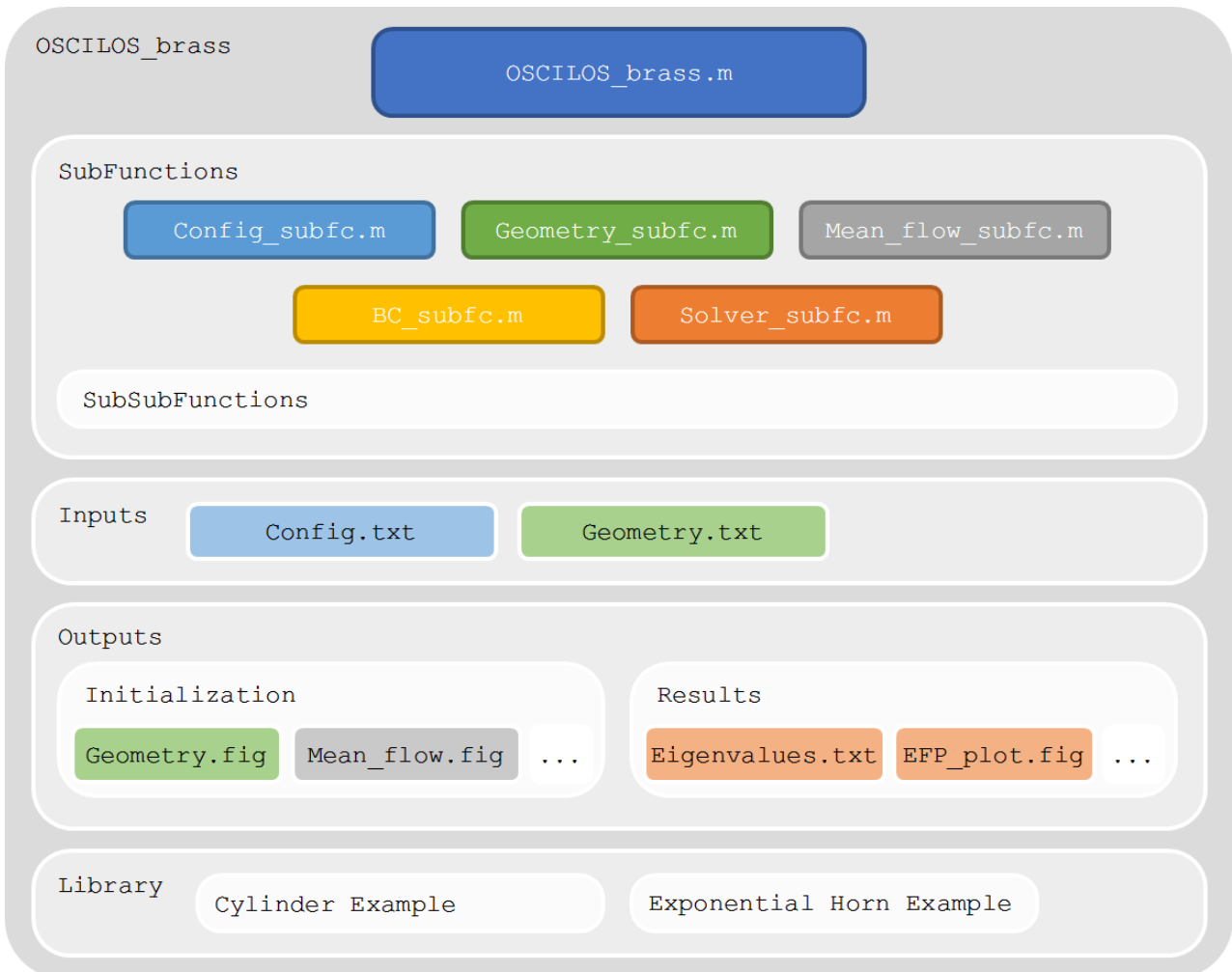


Figure 1: Directory tree and key files used by OSCILOS_{brass}.

Two files are required as **input**, and must be located in `./Inputs` - see Sec. 3.

- The **configuration** file (`Config.txt` by default) specifies the configuration options
- The **geometry** file (`Geometry.txt` by default) defines the geometry to solve

All **output** data is written to `./Outputs` - see Sec. 4.

The top-level `./OSCILOS_brass.m` script runs the 5 subfunctions in the order listed below.

These are located in `./SubFunctions/`.

1. `Config_subfc` - reads and initialises configuration parameters
2. `Geometry_subfc` - reads and initialises the geometry
3. `Mean_flow_subfc` - calculates the flow properties throughout the domain
4. `BC_subfc` - initialises the boundary conditions
5. `Solver_subfc` - solves the equations and outputs the various results

1.2 Quick Examples

A walkthrough of two basic examples located in the `./Library` folder is provided as a quick demonstration of `OSCILOSbrass`.

1.2.1 Open Cylinder (e.g. Boomwhacker[®])

Consider a long, thin cylindrical pipe, open at both ends, sustaining no axial mean flow, subject to some transient excitation. A real-life example is a Boomwhacker[®], the popular plastic tuned-percussion instrument. We wish to establish the resonant modes of the cavity within the pipe based on its geometry.

Initially, we choose to model each end as perfectly open, i.e. a pressure-release condition with reflection coefficient $\tilde{\mathcal{R}} = -1$, and we assume standard atmospheric conditions. The minimum configuration file required to achieve this is as follows:

```
p1          101325.0 % Mean pressure at inlet [Pa]
T1          298.2   % Mean Temperature at inlet [K]
M1_u1      1e-5    % Value of M1 [-] (nonzero but negligible)

inlet_type  1      % Pressure-release inlet boundary condition
outlet_type 1      % Pressure-release outlet boundary condition
```

Configuration options are discussed in detail in Sec. 3.1

The geometry for a simple cylinder of length 1.263 m and radius 0.05 m (corresponding to the length of the C3 boomwhacker) may be specified with the geometry file below:

x [m]	r [m]	SectionIndex	TubeIndex
0	0.05	0	0
0.6315	0.05	0	0
1.263	0.05	0	0

This geometry is visualised in Fig. 2. Geometry specification is discussed in detail in Sec. 3.2.

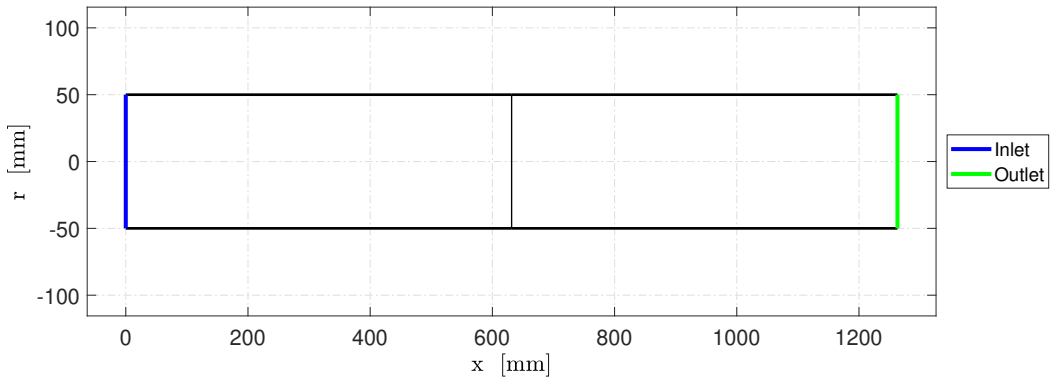


Figure 2: Geometry plot for Cylinder example.

The output file produced by `OSCILOSbrass` is then as follows:

Mode number	Frequency [Hz]	Growthrate [1/s]	EFP [cents]	Note
1	137.05	0.00	+0.00	C#3-19.4¢
2	274.09	0.00	+0.00	C#4-19.4¢
3	411.14	0.00	+0.00	G#4-17.5¢
4	548.18	0.00	+0.00	C#5-19.4¢
5	685.23	0.00	+0.00	F5-33.1¢
6	822.27	0.00	+0.00	G#5-17.5¢
7	959.32	0.00	+0.00	Bb5+49.4¢

The zero growth rate of these modes agrees with analytical expectations - the $\tilde{\mathcal{R}} = -1$ boundary conditions do not account for any loss mechanism so we expect neither amplification

nor attenuation with time. The perfect harmonicity (zero deviation from Equivalent Fundamental Pitch, see Sec. 4.2.3 for more details), implying the modes fall exactly at integer multiples of the fundamental frequency, is also as expected.

The pitch however falls around 80c sharp of the expected C3. Additionally we seek to model the loss mechanisms responsible for faster decay of higher frequencies. One such model is given by Levine and Schwinger [2] for thin open-ended cylinders such as this. Replacing the boundary condition specifications as follows implements a polynomial approximation to the Levine-Schwinger condition:

```
inlet_type 11 % Levine-Schwinger inlet boundary condition
outlet_type 11 % Levine-Schwinger outlet boundary condition
```

In this case the output file becomes:

Mode number	Frequency [Hz]	Growthrate [1/s]	EFP [cents]	Note
1	130.71	-1.88	-2.37	C3-1.3¢
2	261.50	-7.30	-1.88	C4-0.9¢
3	392.43	-15.92	-1.09	G4+1.9¢
4	523.56	-27.34	+0.00	C5+1.0¢
5	654.96	-41.20	+1.34	E5-11.3¢
6	786.66	-57.11	+2.91	G5+5.9¢
7	918.71	-74.76	+4.67	Bb5-25.5¢

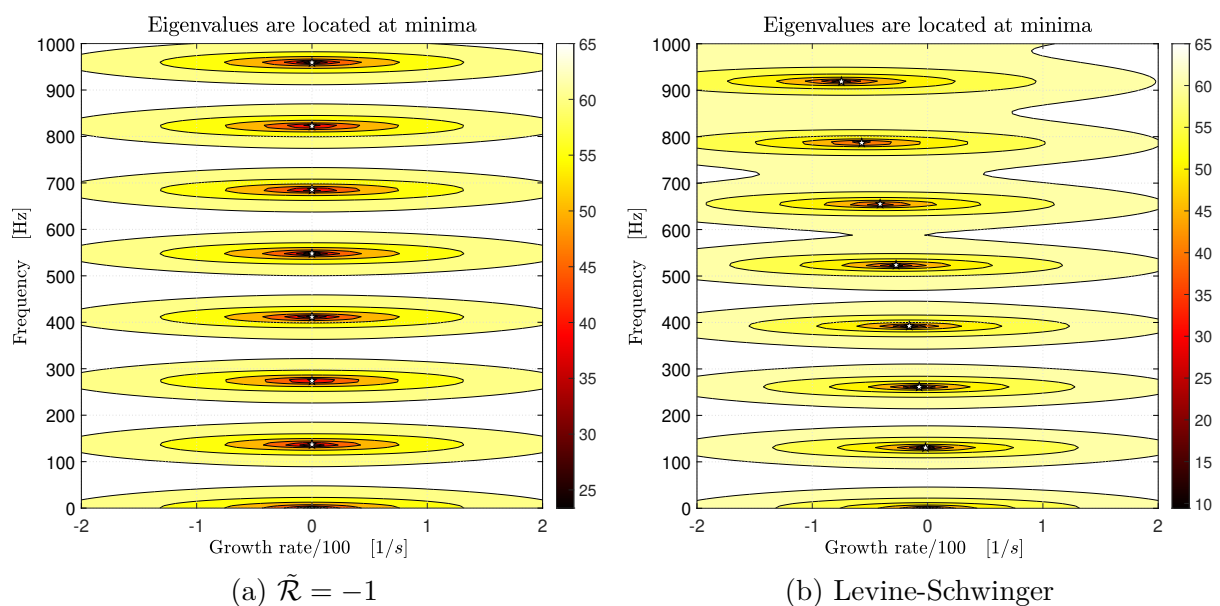
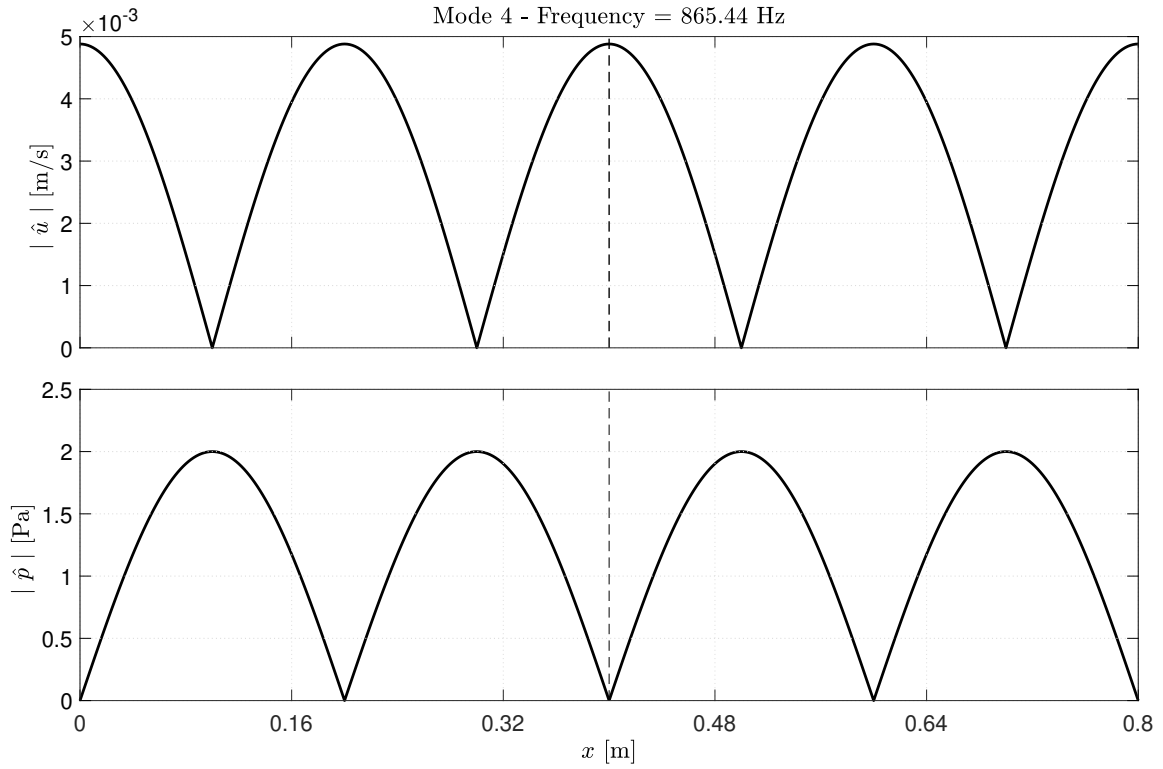
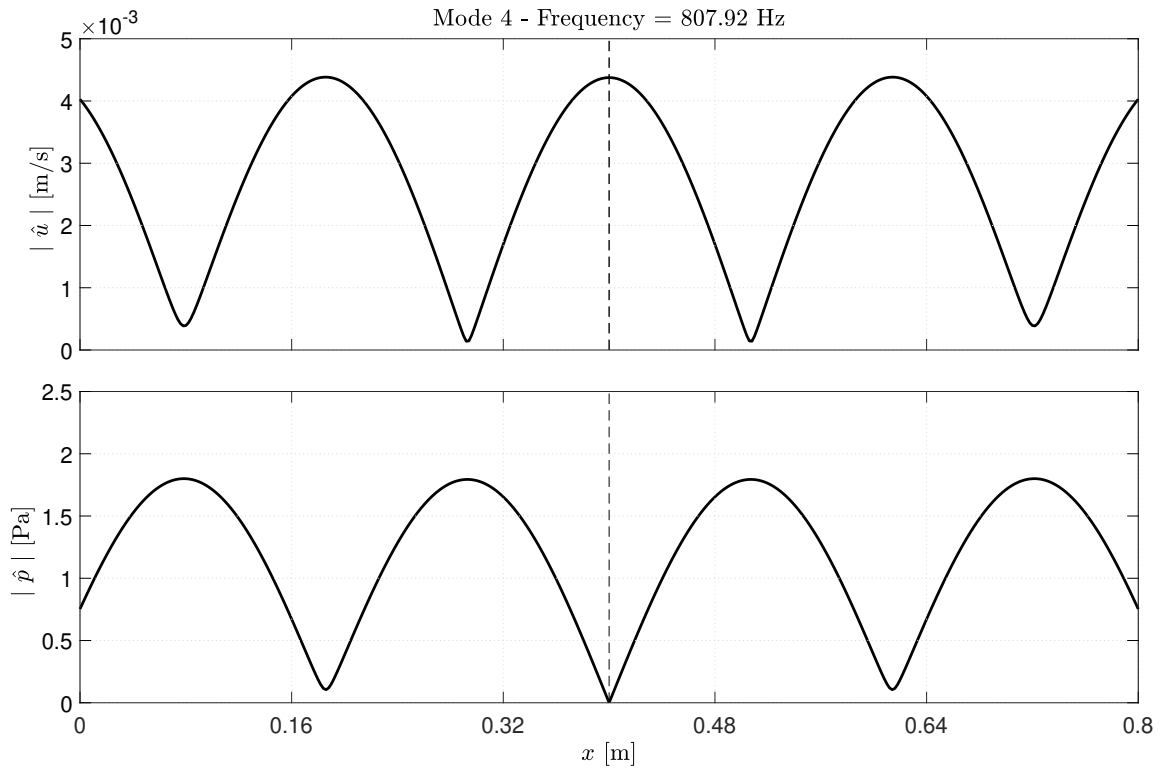


Figure 3: Eigenspace contour plot for Cylinder example.



(a) $\tilde{\mathcal{R}} = -1$



(b) Levine-Schwinger

Figure 4: Mode shapes for Cylinder example.

Therefore for this geometry, introducing the Levine-Schwinger condition lowers the fundamental by 80c to just under C3, and the familiar harmonic series on C is apparent. Inharmonicity is also introduced, due to the frequency dependence of the end correction, which alters each modal frequency to a different extent. The growth rate decreases with frequency on account of the greater radiation efficiency of the open ends at higher frequencies.

The plot of the eigenspace residual for the two cases is shown in Fig. 3.

Mode shapes for the 4th mode, showing the axial distribution of acoustic pressure and velocity magnitudes, are shown in Fig. 4.

1.2.2 Exponential Horn (e.g. trumpet bell approximation)

Backus and Hundley [3] observe that a rudimentary approximation of brass instrument geometries may be achieved using simple cylindrical, conical and exponential-horn shaped sections. Consider a cylinder connected to a diverging exponential horn, with inlet velocity $u_1 = 0.5$ m/s typical of quiet playing. Imposing a closed-end boundary at the inlet (narrow end) and the Levine-Schwinger condition at the outlet (wide end) provides a very basic approximation to the playing conditions of a trumpet. No attempt is made to accurately represent the exact bore dimensions, or the effect of the mouthpiece and lips. We wish to establish the modes of this simple system.

The minimum config file for these conditions is as follows:

```
p1          101325.0  % Mean pressure at inlet [Pa]
T1          298.2    % Mean Temperature at inlet [K]
choice_M1_u1 2      % Specify u1
M1_u1       0.5     % Value of u1 [m/s]

inlet_type  2       % Closed-end inlet BC
outlet_type 11      % Levine-Schwinger outlet BC
```

The geometry file for a cylinder and exponential diverging horn, as shown in Fig. 5, is:

```
x [m]          r [m]          SectionIndex  TubeIndex
0              0.01          0             0
0.5           0.01          0             2
1.2           0.05          0             0
```

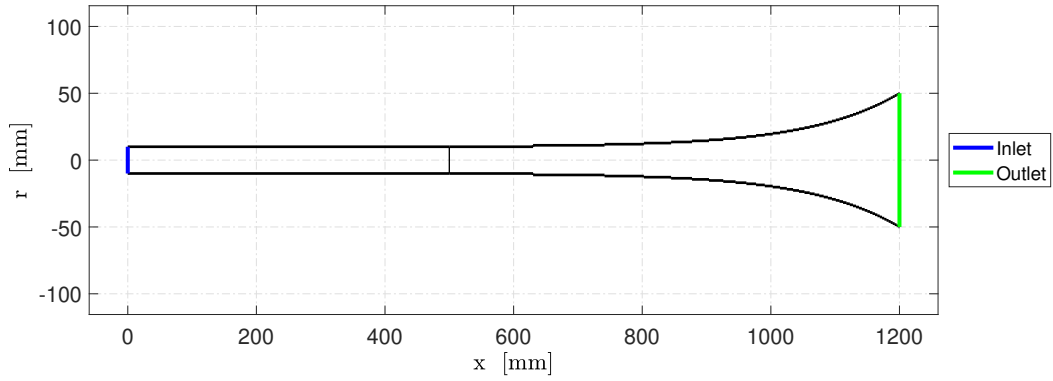



Figure 5: Geometry plot for exponential horn example.

The results produced by $\text{OSCILOS}_{\text{brass}}$ are as follows:

Mode number	Frequency [Hz]	Growtrate [1/s]	EFP [cents]	Note
1	96.72	0.46	-515.63	G2-22.7¢
2	257.71	-0.61	-19.04	C4-26.1¢
3	392.96	-4.79	+9.39	G4+4.3¢
4	521.11	-14.42	+0.00	C5-7.1¢
5	658.11	-24.02	+17.75	E5-3.0¢
6	789.18	-38.43	+16.55	G5+11.4¢
7	933.20	-48.38	+39.87	Bb5+1.6¢

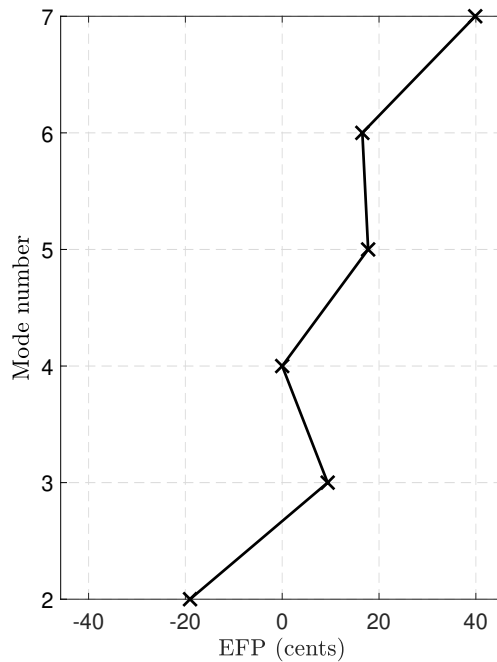


Figure 6: Equivalent Fundamental Pitch deviation plot for exponential horn example.

Note that even with this extremely basic model, characteristic behaviours of the input impedance peaks of trumpets and trombones are evident: a very flat (in pitch) fundamental mode, followed by a reasonably harmonic series of modes, the higher ones with faster attenuation due to better radiation efficiency at the bell.

The inharmonicity plot, visualising the deviation from equivalent fundamental pitch with reference to $1/4$ of the frequency of mode 4, is shown in Fig. 6. The reader is referred to Sec. 4.2.3 for a more detailed discussion of this plot. The inharmonicity pattern is reminiscent of that found from experimental measurements of trombones, e.g. by Braden [4].

2 Solver Model

A summary of the working principles of the solver is provided here. For a more rigorous derivation and discussion of the underlying equations, the reader is referred to the OSCILOS Technical Report [1].

2.1 Assumptions

The OSCILOS_{brass} solver makes the following assumptions:

1. Circular bore cross-section, 1D analysis
2. No heat addition, flames or flame perturbations (likely to hold for many brass instruments). OSCILOS_{lite} and OSCILOS_{long} are capable of modelling these.
3. No entropy waves, heat exchangers, branched Helmholtz resonators or bore perforations (also holds for most brass instruments). OSCILOS_{long} is capable of modelling these.
4. ‘Low’ frequency and ‘narrow’ bore - higher transverse eigenmodes are assumed evanescent. According to Candel and Poinso [5] this is valid on condition that

$$f < \frac{c}{2\pi a} \alpha_{01} \quad (1)$$

for frequency of interest f where c is local sound speed in a bore of radius a , and $\alpha_{01} = 1.8412$ is the 1st root of the first derivative of the first order Bessel function of the first kind, $J'_1(\alpha_{01}) = 0$. For the largest trumpet bell diameter given by Bilbao and Chick [6], the limit of validity is 3.2 kHz, which lies above the 25th mode of a theoretical perfectly harmonic trumpet with equivalent 4th mode.

5. The only stagnation pressure loss in the bore is associated with the expansion of the mean flow as the bore diverges. This does not hold at high sound pressure amplitudes, at which shock waves have been observed, e.g. by Hirschberg et al. [7]. The wavefront-steepening behaviour observed under these conditions cannot be captured by the linear treatment used by this solver in its current implementation.

2.2 Governing Equations

This section outlines the underlying equations of the acoustic wave model, mean flow calculation, and the interpolation of nonlinear geometric profiles.

2.2.1 Acoustic Wave Equations

OSCILOS_{brass} uses a 1D plane-wave ‘transmission-line’ model for acoustic perturbations. Analysis is conducted on a model instrument bore comprising two or more cylindrical elements connected coaxially in sequence, as shown in Fig. 7. The interfaces between the cylinders have axial location x_k where $k = 0, 1, 2, \dots, N$. The cylinders themselves are termed ‘tubes’, of which there are N by definition. The k th tube has inlet at x_{k-1} and outlet at x_k .

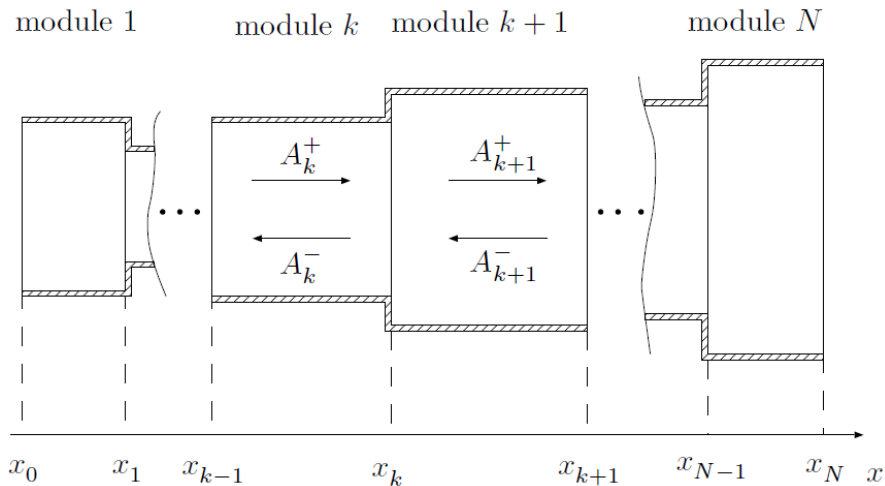


Figure 7: Schematic view of 1D discretised instrument bore model comprising connected cylindrical sections.

The acoustic perturbations are expressed as the sum of downstream and upstream propagating waves with amplitudes A_k^+ and A_k^- respectively.

The pressure p , axial velocity u and density ρ in tube k are therefore given by Eq. 2.

$$p_k(x, t) = \bar{p}_k + p'_k(x, t) = \bar{p}_k + A_k^+(t - \tau_k^+) + A_k^-(t - \tau_k^-) \quad (2a)$$

$$u_k(x, t) = \bar{u}_k + u'_k(x, t) = \bar{u}_k + \frac{1}{\bar{\rho}_k \bar{c}_k} \left[A_k^+(t - \tau_k^+) + A_k^-(t - \tau_k^-) \right] \quad (2b)$$

$$\rho_k(x, t) = \bar{\rho}_k + \rho'_k(x, t) = \bar{\rho}_k + \frac{1}{\bar{c}_k^2} \left[A_k^+(t - \tau_k^+) + A_k^-(t - \tau_k^-) \right] \quad (2c)$$

The convective time delays τ_k^+ and τ_k^- are

$$\tau_k^+ = \frac{x - x_{k-1}}{\bar{c}_k + \bar{u}_k} \quad \tau_k^- = \frac{x_k - x}{\bar{c}_k - \bar{u}_k} \quad (3)$$

2.2.2 Mean Flow Equations

The governing equations implemented by OSCILOS_{brass} are given, in matrix form, by Eq. 4.

$$\mathcal{B}_{k,2} \begin{bmatrix} \tilde{p}_{k+1}(x_k, s) \\ \bar{\rho}_{k+1} \bar{c}_{k+1} \tilde{u}_{k+1}(x_k, s) \\ \tilde{\rho}_{k+1}(x_k, s) \bar{c}_{k+1}^2 \end{bmatrix} = \mathcal{B}_{k,1} \begin{bmatrix} \tilde{p}_k(x_k, s) \\ \bar{\rho}_k \bar{c}_k \tilde{u}_k(x_k, s) \\ \tilde{\rho}_k(x_k, s) \bar{c}_k^2 \end{bmatrix} \quad (4)$$

The superscript $\tilde{}$ denotes the Laplace transform. $s = \sigma + 2\pi i f$, where s is the Laplace variable, σ is the growth rate and f is the frequency.

The flow in converging (area-decreasing) and diverging (area-increasing) sections is modelled distinctly by OSCILOS_{brass}, using different definitions for $\mathcal{B}_{k,1}$ and $\mathcal{B}_{k,2}$. This distinction remains from the solver's application to gas turbine engines. The stagnation pressure loss due to mean flow expansion is not usually considered significant in brass instruments.

Area-increasing

At the area increase interface, the mass and energy flux are unchanged but the momentum flux is increased by the axial force on the walls [8]. The \mathcal{B}_k matrices therefore become

$$\mathcal{B}_{k,1} = \begin{bmatrix} 0 & \frac{\bar{c}_{k+1}}{\bar{c}_k} & \bar{M}_k \frac{\bar{c}_{k+1}}{\bar{c}_k} \\ \Theta_k & 2\bar{M}_k & \bar{M}_k^2 \\ \frac{\gamma}{\gamma-1} \bar{M}_k & \bar{M}_k^2 & -\frac{1}{\gamma-1} \bar{M}_k \end{bmatrix} \quad (5a)$$

$$\mathcal{B}_{k,2} = \Theta_k \begin{bmatrix} 0 & 1 & \bar{M}_{k+1} \\ 1 & 2\bar{M}_{k+1} & \bar{M}_{k+1}^2 \\ \frac{\gamma}{\gamma-1} \frac{\bar{c}_{k+1}}{\bar{c}_k} \bar{M}_{k+1} & \frac{\bar{c}_{k+1}}{\bar{c}_k} \bar{M}_{k+1}^2 & -\frac{\gamma}{\gamma-1} \frac{\bar{c}_{k+1}}{\bar{c}_k} \bar{M}_{k+1} \end{bmatrix} \quad (5b)$$

where $\Theta_k = \frac{S_{k+1}}{S_k}$ denotes the downstream ratio of sectional surface areas, and \bar{M}_k the mean tube mach number.

Area-decreasing

At an area decrease interface, the mass and energy flux are unchanged, and the flow through the interface may be considered isentropic. This leads to

$$\mathcal{B}_{k,1} = \begin{bmatrix} 0 & \frac{\bar{c}_{k+1}}{\bar{c}_k} & \bar{M}_k \frac{\bar{c}_{k+1}}{\bar{c}_k} \\ \frac{1}{\bar{\rho}_k} & 0 & -\frac{1}{\bar{\rho}_k} \\ \frac{\gamma}{\gamma-1} \bar{M}_k & \bar{M}_k^2 & -\frac{1}{\gamma-1} \bar{M}_k \end{bmatrix} \quad (6a)$$

$$\mathcal{B}_{k,2} = \Theta_k \begin{bmatrix} 0 & 1 & \bar{M}_{k+1} \\ \frac{1}{\Theta_k \bar{\rho}_{k+1}} & 0 & -\frac{1}{\Theta_k \bar{\rho}_{k+1}} \\ \frac{\gamma}{\gamma-1} \frac{\bar{c}_{k+1}}{\bar{c}_k} \bar{M}_{k+1} & \frac{\bar{c}_{k+1}}{\bar{c}_k} \bar{M}_{k+1}^2 & -\frac{\gamma}{\gamma-1} \frac{\bar{c}_{k+1}}{\bar{c}_k} \bar{M}_{k+1} \end{bmatrix} \quad (6b)$$

Transmission-line equations

The wave amplitudes at the inlet and outlet of the bore are related by Eq. 7. $\bar{E}(s)$ is an entropy wave strength which in this implementation remains uniformly zero throughout the domain.

$$\begin{bmatrix} \bar{A}_N^+(s) \\ \bar{A}_N^-(s) \\ \bar{E}_N(s) \end{bmatrix} = \mathcal{G}_{1,N-1}(s) \begin{bmatrix} \bar{A}_1^+(s) \\ \bar{A}_1^-(s) \\ \bar{E}_1(s) \end{bmatrix} \quad (7)$$

The $\mathcal{G}_{1,N-1}(s)$ matrix is given by

$$\mathcal{G}_{j,k}(s) = \mathcal{Z}_k(s) \mathcal{Z}_{k-1}(s) \dots \mathcal{Z}_j(s) \quad (8)$$

and the $\mathcal{Z}_k(s)$ matrix is defined as

$$\mathcal{Z}_k(s) = (\mathcal{B}_{k,2} \mathcal{C}_2 \mathcal{D}_{k,2}(s))^{-1} \mathcal{B}_{k,1} \mathcal{C}_1 \mathcal{D}_{k,1}(s) \quad (9a)$$

$$\begin{bmatrix} \bar{A}_{k+1}^+(s) \\ \bar{A}_{k+1}^-(s) \\ \bar{E}_{k+1}(s) \end{bmatrix} = \mathcal{Z}_k(s) \begin{bmatrix} \bar{A}_k^+(s) \\ \bar{A}_k^-(s) \\ \bar{E}_k(s) \end{bmatrix} \quad (9b)$$

where

$$\mathcal{C}_1 = \mathcal{C}_2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \quad (10)$$

$$\mathcal{D}_{k,1}(s) = \begin{bmatrix} e^{-\tau_k^+ s} \\ 1 \\ e^{-\tau_k^- s} \end{bmatrix} \quad \mathcal{D}_{k,2}(s) = \begin{bmatrix} 1 \\ e^{-\tau_{k+1}^- s} \\ 1 \end{bmatrix} \quad (11)$$

$$\tau_k^s = \frac{x - x_{k-1}}{\bar{u}_k} \quad (12)$$

2.2.3 Discretisation Scheme

The ability of OSCILOS_{brass} to interpolate over several analytical profiles, creating a ‘staircase’ of cylindrical sections, is detailed in Sec. 3.2.1. The equations behind the implementation are included here in general form.

Consider an analytical profile described by some shape function $\eta = P(z)$ where η is an arbitrary radial co-ordinate, z an arbitrary axial co-ordinate and P some shape function defined over a given z -range. We seek to discretise radially and axially, mapping η to r and z to x to create a ‘staircase’ profile which achieves:

- Initial radius r_0 at axial location x_0 , and final radius r_N at x_N
- N cylindrical sections (‘steps’ on the staircase)
- Minimal deviation from the analytical profile
- Enclosed revolved volume nearly equivalent to that of the analytical profile

The requirement for the distance between x_0 and x_N to be divided into N cylinders necessitates $N + 1$ interfaces, so x is sampled by $N + 1$ points, as shown in Fig. 8.

In order to achieve minimum deviation from the analytical enclosed volume, we wish the analytical profile to pass through the ‘centre’ of the staircase. Consequently, we require:

- The radius of each tube to fall between the analytical radii at its ends, with the exception of the first and last cylinders, whose radii are fixed by r_0 and r_N respectively
- The first and last cylinders to be ‘half-length’ to avoid excessive deviation at their internal interfaces

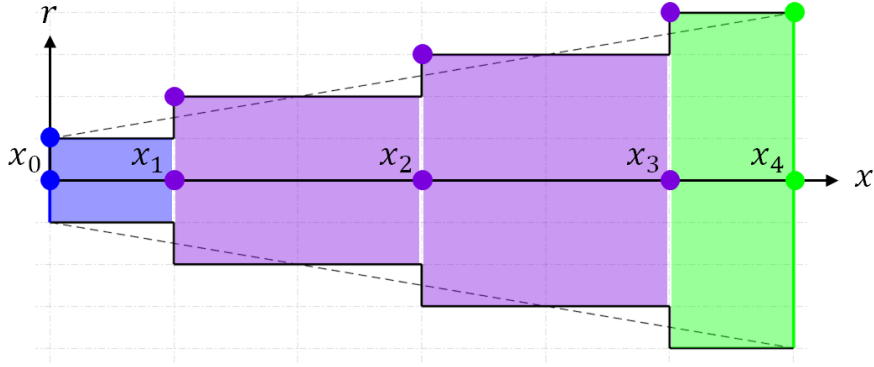


Figure 8: Conical profile with $N = 4$ cylindrical sections showing interface locations x_k .

This is achieved using a finer discretisation in z than in x . The radius function is sampled at axial points intermediate to the interface locations. The z co-ordinate is therefore sampled by $2N - 1$ points, such that there are $N + 1$ interface locations and $N - 2$ intermediate radius sample points, as shown in Fig. 9. The z -range required to define the shape of $P(z)$ is therefore $z_0 \leq z \leq z_{2N-2}$.

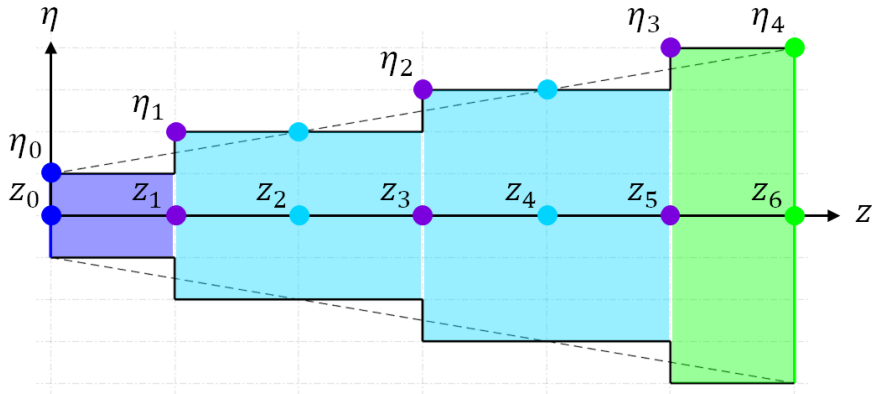


Figure 9: Conical profile with $N = 4$ cylindrical sections showing $2N - 1 = 7$ axial sampling locations z_m , with interfaces located at z_0, z_1, z_3, z_5, z_6 and radii η_k sampled at z_0, z_2, z_4, z_6, z_6 .

Additionally, for nonlinear profiles, we desire the axial samples to have greater density in regions of larger radial gradient $\frac{dP}{dz}$. To achieve this, the z samples are distributed using the inverse of $P(z)$, $P^{-1}(\eta)$. A linearly spaced set of η co-ordinates are passed to the P^{-1} function according to Eq. 13, yielding the vector $[\mathbf{z}]$ whose density is biased by $\frac{dP}{dz}$, as shown in Fig. 10. $\text{linspace}(a, b, n)$ indicates a vector of n linearly spaced points between a and b .

$$[\mathbf{z}] = P^{-1}(\text{linspace}(P(z_0), P(z_{2N-2}), 2N - 1)) \quad (13)$$

The co-ordinates in $[\mathbf{z}]$ are then linearly mapped to x by $\Xi(z)$ as in Eq. 14a, and the radius r is given by $R(z)$ according to Eq. 14b.

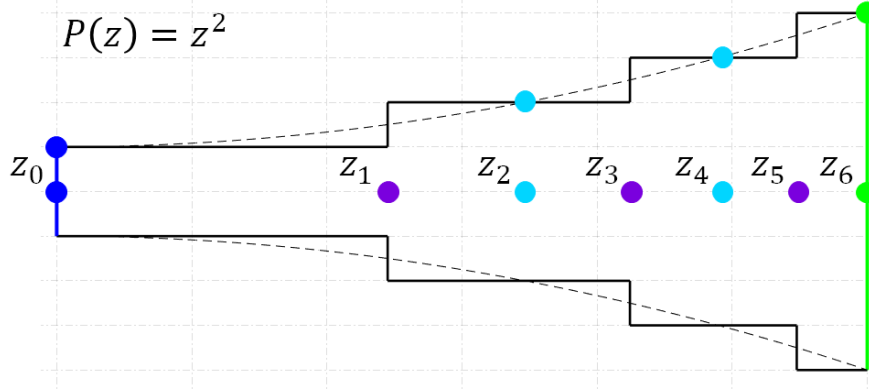


Figure 10: Parabolic profile $P(z) = z^2$, interpolated by $N = 4$ cylindrical sections showing $2N - 1 = 7$ axial sampling locations z_m , spaced according to the square root of a linearly spaced sequence of η values between $P(z_0)$ and $P(z_{2N-2})$.

$$\Xi(z) = x_0 + (x_N - x_0) \frac{z - z_0}{z_{2N-2} - z_0} \quad (14a)$$

$$R(z) = r_0 + (r_N - r_0) \frac{P(z) - P(z_0)}{P(z_{2N-2}) - P(z_0)} \quad (14b)$$

It remains to generate the radial co-ordinates $[\mathbf{x}]$ and axial co-ordinates $[\mathbf{r}]$, as given in Eqs. 15a and 15b respectively. Note that the radial dimension is the same for the final two interfaces, because each describes its downstream cylinder.

$$[\mathbf{x}] = \Xi([z_0, z_1, z_3, z_5, \dots, z_{2N-3}, z_{2N-2}]) \quad (15a)$$

$$[\mathbf{r}] = R([z_0, z_2, z_4, z_6, \dots, z_{2N-2}, z_{2N-2}]) \quad (15b)$$

This discretisation scheme allows the specification of a general shape function $P(z)$. For each shape (e.g. linear, parabola, exponential), only the function $P(z)$, the inverse function $P^{-1}(\eta)$, and the initial and final z values, are required. These are given in the definition of each supported `TubeIndex` in Table. 7.

3 Inputs

OSCILOS_{brass} takes as input a list of configuration settings, and the geometry to solve for. These are each given by human-readable files, `Config.txt` and `Geometry.txt` respectively, located in the `./Inputs/` subfolder. In addition, when calling `OSCILOS_brass()` as a function, configuration parameters may be supplied in `Name, Value` pairs.

3.1 Configuration

The configuration parameters are read from the configuration file, `./Inputs/Config.txt`, by default. The values of these parameters define the boundary conditions used, the mean flow conditions, the scope of the solution, plot options and save locations for the various inputs and outputs.

Each line of the config file (except those for additional boundary condition data described in Sec. 3.1.6, Sec. 3.1.7 and Sec. 3.1.8) should take the form

```
param_name      param_value      [unparsed text]
```

The config file is whitespace-separated (e.g. tab-separated), and parsed line-by-line. Lines which do not begin with a valid `param_name` are ignored.

Each `param_value` has one of four types:

- **integer** - a positive whole number
- **double** - a floating point number saved to double precision
- **string** - a string of characters containing no whitespace
- **logical** - a number or string which represents a boolean value. `TRUE`, `ON` and `1` are interpreted as true, and any other character string is interpreted as false

The `param_value` is read up to and excluding the next whitespace, newline or `%` character. Where the expected `param_value` type is not found, a warning is thrown and the default parameter value is used if it exists. `param_name` and `param_value` are case-insensitive, so `INLET_TYPE` is interpreted as `inlet_type`, and `TRUE` as `true`.

In addition, when `OSCILOSbrass` is invoked by a function call to `OSCILOSbrass`, configuration options may be specified as arguments to `OSCILOS_brass()`, in `Name, Value` argument pairs.

Options passed by arguments override those parsed from the config file, allowing one or more parameters to be updated between successive calls to `OSCILOS_brass()` without any change to the config file. This allows for easy setup of parametric investigative studies seeking to iterate over a particular parameter, such as the example in Sec. 5.

3.1.1 Mean Flow

Brass instruments experience a mean flow, supplied by the player. The Mach numbers achieved under playing conditions however are very small (typically < 0.01) and consequently do not significantly affect the acoustic behaviour of the bore. `OSCILOSbrass` however requires a nonzero mean flow to avoid singularity of the system of flow equations.

Flow conditions at the inlet - pressure, temperature and velocity - are supplied by the Mean Flow parameters in Table 1. Flow velocity is specified either as a Mach number or as a velocity, as selected by `choice_M1_u1`.

If the magnitude of the inlet Mach number is small enough to risk singularity or ill-conditioning of the flow equations, it will be increased to a negligible, but finite, value (currently $1e-5$).

Table 1: Mean Flow Configuration Parameters

Name	Type	Optional	Default	Purpose
<code>p1</code>	double	✗	-	Inlet Pressure [Pa]
<code>T1</code>	double	✗	-	Inlet Temperature [K]
<code>M1_u1</code>	double	✗	-	Inlet Mach number [-] or Velocity [m/s]
<code>choice_M1_u1</code>	integer	✓	1	<code>choice_M1_u1 = 1</code> \implies Mach number <code>choice_M1_u1 = 2</code> \implies Velocity
<code>choice_gamma</code>	integer	✓	1	<i>Not currently supported</i> <code>choice_gamma = 1</code> \implies γ constant <code>choice_gamma = 2</code> \implies $\gamma = \gamma(T)$

3.1.2 Scan Range

The `OSCILOS` solver uses an eigensolver which performs a search for the complex eigenvalues representing the modes of the system by the Laplace variable s . The scope of the search, in terms of frequency ($\text{Im}[s]$) and growth rate ($\text{Re}[s]$), is determined from the scan range configuration parameters shown in Table 2.

Table 2: Scan Range Configuration Parameters

Name	Type	Optional	Default	Purpose
<code>min_freq</code>	double	✓	0	Frequency scan lower bound [Hz]
<code>max_freq</code>	double	✓	1000	Frequency scan upper bound [Hz]
<code>number_freq</code>	integer	✓	20	Number of frequency starting points
<code>min_GR</code>	double	✓	-200	Growth rate scan lower bound [1/s]
<code>max_GR</code>	double	✓	200	Growth rate scan upper bound [1/s]
<code>number_GR</code>	integer	✓	10	Number of growth rate starting points

The search is conducted by MATLAB’s `fsolve()` from seed points spaced linearly throughout frequency and growth rate ranges specified - the number of seed points is configurable using `number_freq` and `number_GR`. If the number of modes is close to or greater than `number_freq`, the solver may miss eigenvalues, and this will be evident on the contour plot, see Sec. 4.2.2.

3.1.3 Outputs and Plots

This section lists the configuration parameters pertaining to outputs and plots. For more detailed discussion of their meaning, see Sec. 4.

Among the various formats in which `OSCILOSbrass` outputs data, there is wide variation in the implications of each for solution time and output file size. The configuration parameters in Table 3 enable the user to disable certain output types to reduce solution time. In particular, the user’s attention is drawn to the eigenvalue contour map, the calculation of which is responsible for a very significant proportion of the solution time if the geometry is complex - setting `contour_plot` to `OFF` may in some cases reduce the runtime by more than half.

3.1.4 Filenames

The name of the various text files for input and output may be specified by the parameters in Table 4. The `config_filename` parameter cannot be specified by the config file, as it locates the file itself, so is only relevant if passed as an argument when `OSCILOSbrass` is called as a function.

Table 3: Output and Plot Configuration Parameters

Name	Type	Optional	Default	Purpose
geom_plot	logical	✓	TRUE	Plot geometry
mf_plot	logical	✓	TRUE	Plot mean flow
bc_plot	logical	✓	TRUE	Plot boundary conditions
contour_plot	logical	✓	TRUE	Plot Eigenspace Contour Map
efp_plot	logical	✓	TRUE	Plot Equivalent Fundamental Pitch
plot_modes	integer	✓	5	No. of mode shapes to plot, 0 to disable
small_plots	logical	✓	FALSE	Reduce plot sizes (for smaller screens)
save_figs	logical	✓	TRUE	Save plots as <code>.fig</code>
save_pdfs	logical	✓	TRUE	Save plots as <code>.pdf</code>
run_name	string	✓		Name of output subfolder. A new folder will be created under <code>./Outputs/</code> . Leave blank to write directly to <code>./Outputs</code> .
warn_ovwrt	logical	✓	TRUE	Bring up dialog box if the folder specified by <code>run_name</code> already exists
cl_out	logical	✓	TRUE	Output progress updates to command line
log_out	logical	✓	TRUE	Output progress updates to log file
fin_msg	logical	✓	TRUE	Display messagebox when finished
no_popups	logical	✓	FALSE	Disable all popups and plots (faster)

Table 4: Text File Name Configuration Parameters

Name	Type	Optional	Default	Purpose
config_filename	string	✓	Config.txt	Name of config file (input)
geom_filename	string	✓	Geometry.txt	Name of geometry file (input)
eig_filename	string	✓	Eigenvalues.txt	Name of mode list file (output)
geom_filename	string	✓	Logfile.txt	Name of log file (output)

3.1.5 Boundary Conditions

OSCILOS_{brass} supports several different boundary condition (BC) types, which are specified using the `inlet_type` and `outlet_type` parameters. The inlet and outlet BCs may each require up to three numerical parameters to define the BC, given by `inlet_param1`, `inlet_param2` and `inlet_param3` for the inlet, and by an equivalent set for the outlet, as detailed in Table 5.

Table 5: Boundary Condition Configuration Parameters

Name	Type	Optional	Default	Purpose
<code>inlet_type</code>	integer	✗	-	Specifies BC type at inlet, see Table 6
<code>inlet_param1</code>	double	✓	0	Parameter 1 for inlet BC, if required
<code>inlet_param2</code>	double	✓	0	Parameter 2 for inlet BC, if required
<code>inlet_param3</code>	double	✓	0	Parameter 3 for inlet BC, if required
<code>outlet_type</code>	integer	✗	-	Specifies BC type at outlet, see Table 6
<code>outlet_param1</code>	double	✓	0	Parameter 1 for outlet BC, if required
<code>outlet_param2</code>	double	✓	0	Parameter 2 for outlet BC, if required
<code>outlet_param3</code>	double	✓	0	Parameter 3 for outlet BC, if required

The number of parameters required and interpretation of their values depends on the BC type selected by `inlet_type` and `outlet_type`. Supported values of the `*let_type` parameters are given in Table 6, where `*let_` refers both to `inlet_` and `outlet_`.

User-defined boundary conditions may be achieved by supplying additional data, which is included in the config file. Note that the format in which these data are required differs from the other configuration options. The order in which the configuration options and the user data appears is immaterial, but each user data block must be contiguous and contain no blank lines.

Table 6: Boundary Condition types

Name	Type	Value	Meaning
*let_type	integer	1	Open condition, $\tilde{\mathcal{R}} = -1$
		2	Closed condition, $\tilde{\mathcal{R}} = 1$
		3	Not implemented <i>Reserved for Choked condition</i>
		4	Time lag condition, $\tilde{\mathcal{R}}(s) = Ae^{-s\tau}$ $A = \text{*let_param1}$ (amplitude) $\tau = \text{*let_param2}/1000$ (time lag in ms)
		5	Phase lag condition, $\tilde{\mathcal{R}} = Ae^{-i\varphi}$ $A = \text{*let_param1}$ (amplitude) $\varphi = \text{*let_param2}$ (phase) $\text{*let_param3} = 0$ for *let_param2 in [radians] $\text{*let_param3} = 1$ for *let_param2 in [degrees]
		6	Polynomial from Coefficients with Time Delay condition, $\tilde{\mathcal{R}}(s) = \frac{b_n s^n + b_{n-1} s^{n-1} + \dots + b_1 s + b_0}{a_m s^m + a_{m-1} s^{m-1} + \dots + a_1 s + a_0} e^{-s\tau}$ $\tau = \text{*let_param1}/1000$ (time lag in ms) $n = \text{*let_param2}$ (numerator order) $m = \text{*let_param3}$ (denominator order) Values of a and b are read separately from the Config file, see Sec. 3.1.6
		7	Not implemented <i>Reserved for Polynomial from User Data with Time Delay condition</i>
		8	Not implemented <i>Reserved for Heat Exchanger condition</i>
		9	Gain and Phase Interpolated from User Data condition $\text{*let_param1} = 0$ for phase in [radians] $\text{*let_param1} = 1$ for phase in [degrees] Gain and Phase data vs Frequency [Hz] are read separately from the Config file, see Sec. 3.1.7
		10	User-defined Gain and Phase functions condition $\text{*let_param1} = 0$ for phase in [radians] $\text{*let_param1} = 1$ for phase in [degrees] Gain and Phase function definitions are read separately from the Config file, see Sec. 3.1.8

Table 6: Boundary Condition types (continued)

11	<p>Unflanged Levine-Schwinger condition [2] using polynomial approximations for R and l by Norris and Sheng [9]</p> $\tilde{\mathcal{R}}(k) = - R e^{-2ikl} \quad k(s) = \frac{\text{Im}[s]}{c}$
12	<p>Flanged Levine-Schwinger condition using polynomial approximations for R and l by Norris and Sheng [9]</p> $\tilde{\mathcal{R}}(k) = - R e^{-2ikl} \quad k(s) = \frac{\text{Im}[s]}{c}$
13	<p>Oscillating Piston Radiating into Hard Tube condition [10] [11]</p> $Z_s = Z_0 + j\omega\rho_0 \sum_{n \geq 1} \frac{J_1^2(k_{0n}a)}{k_{0n}^2 \sqrt{k_{0n}^2 - k_0^2} \cdot J_0^2(k_{0n}a)}$ <p>k_{0n} is the nth root of $J_0'(k_{0n}R) = 0$ $\tilde{\mathcal{R}} = \frac{Z_s \frac{R^2}{a^2} - Z_0}{Z_s \frac{R^2}{a^2} + Z_0}$</p> <p>$R$ = inlet/outlet radius given by geometry $a = R \cdot \star\text{let_param1}$ (piston radius, $\star\text{let_param1}$ is dimensionless)</p>
14	<p>Oscillating Masses Due to Orifice Ahead of Oscillating Piston condition [11] [12]</p> $M_I = 4\pi b^2 \rho_0 R \sum_{m \geq 1} \frac{J_1^2(x_m b) \coth(x_m l)}{(x_m R)^3 J_0^2(x_m R)}$ $M_{II} = 4\pi b^2 \rho_0 R \sum_{m \geq 1} \frac{J_1^2(x_m b)}{(x_m R)^3 J_0^2(x_m R)}$ <p>x_m are the roots of $J_0'(x_m R) = 0$</p> $Z_m = \frac{ik(M_I + M_{II})}{\pi R^2}$ $Z_1 = \frac{i(1 - \cos(kd))}{\sin(kd)} \quad Z_2 = \frac{-i}{\sin(kd)}$ $Z_{piston} = \frac{R^2}{a^2} \left[1 + \frac{j\omega}{c} \sum_{n \geq 1} \frac{J_1^2(k_{0n}a)}{k_{0n}^2 \sqrt{k_{0n}^2 - k_0^2} \cdot J_0^2(k_{0n}a)} \right]$ $Z_t = Z_m + Z_1 + \frac{(Z_1 + Z_{piston})Z_2}{Z_1 + Z_{piston} + Z_2} \quad \tilde{\mathcal{R}} = \frac{Z_t - 1}{Z_t + 1}$ <p>R = inlet/outlet radius given by geometry $a = R \cdot \star\text{let_param1}$ (piston radius) $b = R \cdot \star\text{let_param2}$ (orifice radius) $d = R \cdot \star\text{let_param3}$ (orifice offset) $\star\text{let_param1}$, $\star\text{let_param2}$ and $\star\text{let_param3}$ are dimensionless</p>

3.1.6 Polynomial Coefficients for BC6

Boundary Condition type 6 allows the user to specify the frequency dependence of the reflection coefficient as a ratio of polynomials in the Laplace variable, with additional time lag, according to Eq. 16.

$$\tilde{\mathcal{R}}(s) = \frac{b_n s^n + b_{n-1} s^{n-1} + \dots + b_1 s + b_0}{a_m s^m + a_{m-1} s^{m-1} + \dots + a_1 s + a_0} e^{-s\tau} \quad (16)$$

`*let_param2 = n` and `*let_param3 = m` are required to specify the order of the numerator and denominator respectively. `*let_param1/1000 = τ` .

The coefficients $b_n \dots b_0$ and $a_m \dots a_0$ are specified by two lines in the config file. These must be preceded by the header text `INLET_POLYNOMIAL_COEFFICIENTS` for the inlet, or `OUTLET_POLYNOMIAL_COEFFICIENTS` for the outlet. There may optionally be a line of text between the header and the coefficients for readability. The lines must appear in the order described, as shown:

```
[HEADER]
[TITLE TEXT (optional)]
[NUMERATOR COEFFICIENTS b_0 ... b_n]
[DENOMINATOR COEFFICIENTS a_0 ... a_m]
```

$n + 1$ and $m + 1$ coefficients respectively must be supplied in descending power order, with the highest-order coefficient first. So, for example, to specify the coefficients of 2nd order polynomials at the outlet, with time delay 0.05 ms, the following lines should be included in the config file:

```
outlet_type      6
outlet_param1    0.05
outlet_param2    2
outlet_param3    2

OUTLET_POLYNOMIAL_COEFFICIENTS
s^2              s              1
1e-7            2e-4            0.8
2e-7            1e-4            1.2
```


3.1.7 Gain/Phase User Data for BC9

Boundary Condition type 9 enables the user to specify data pertaining to gain and phase at the boundaries, and the reflection coefficient is constructed by interpolation over this dataset.

Frequency must be specified in [Hz]

*let_param1 = 0 \implies phase given in [radians].

*let_param1 = 1 \implies phase given in [degrees].

The data must be included in the config file in tab-separated format, under the header INLET_GAIN_PHASE_DATA or OUTLET_GAIN_PHASE_DATA, in the format shown:

```
[HEADER]
[TITLE TEXT (optional)]
[DATA line 1 {Frequency} {Gain} {Phase}]
[DATA line 2 {Frequency} {Gain} {Phase}]
...
```

So for example, to specify gain and phase data with phase in degrees for the inlet BC, the config file should include the following lines:

```
inlet_type          9
inlet_param1        1

INLET_GAIN_PHASE_DATA
Frequency [Hz]      Gain [-]          Phase [rad_°]
0                  1                180
200                0.9              170
500                0.75             140
1000               0.5              90
```

The data are interpolated using the 'makima' modified Akima Piecewise Cubic Hermite interpolation scheme [13] by default, as shown in Fig. 18 in Sec. 4.1.3.

3.1.8 Gain/Phase Functions for BC10

Boundary Condition type 10 allows the user to specify reflection coefficient gain and phase as analytical functions of frequency. The functions are specified using the MATLAB syntax for anonymous functions, so may not contain any MATLAB keywords and must each occupy a single line. Consequently the functions must begin with the @ character, and each are read until either the end of the line, or the first % character.

*let_param1 is used to specify the units returned by the phase function.

*let_param1 = 0 \implies phase given in [radians].

*let_param1 = 1 \implies phase given in [degrees].

The functions should take a single argument, equivalent to frequency in [Hz]. The functions should immediately follow a header line reading either INLET_GAIN_PHASE_FUNCTIONS or OUTLET_GAIN_PHASE_FUNCTIONS as appropriate. The gain function should be specified first, then the phase function, with no blank lines between them. The function specification should follow the general format:

```
[HEADER]
[GAIN FUNCTION]
[PHASE FUNCTION]
```

So to specify gain and phase as a function of frequency at the outlet, interpreting the phase in degrees, the config file should include the following lines:

```
outlet_type          10
outlet_param1        1

OUTLET_GAIN_PHASE_FUNCTIONS
@(f) 1 + f.^2./10e5      % Gain
@(f) 135 - 0.5*90*cos(f./1000.*pi) % Phase
```

3.2 Geometry

The geometry file defines the instrument bore geometry, and is located at `./Inputs/Geometry.txt` by default, although a different filename may be specified by the `geom_filename` configuration parameter.

OSCILOS_{brass} models the geometry using cylindrical finite elements. The elements are specified by their circular bounding faces, termed ‘sections’. Each section must be located axially, and its radius specified. Each cylindrical element retains the radius of its upstream section, and extends axially to the next section. The inlet is located at the smallest axial co-ordinate, and the outlet at the largest. The sequence of sections in the file is not significant - the sections are sorted such that their axial co-ordinates are monotonically increasing, before the cylinders are constructed.

The geometry file should be tab-separated with at least four columns. Each line of the file represents a section. The role of each column is outlined below:

1. **Axial position x** - Axial position of the section in [m]
2. **Radius r** - Section radius in [m]
3. **SectionIndex** - Section type (nonzero values are not currently supported by OSCILOS_{brass} but parameter retained for compatibility with other OSCILOS versions)
4. **TubeIndex** - Tube shape - additional sections may be interpolated, approximating other tube shapes - see Sec. 3.2.1 and Table 7

An optional 5th column can also be included to control the level of discretisation when **TubeIndex** is nonzero:

5. **TubeSplit** - Number of smaller tubes into which the original tube is divided. If **TubeSplit** = 0, or if this column is not present in the file, the default value of 50 tubes is used. Where **TubeIndex** = 0, **TubeSplit** is ignored.

The geometry file must begin with a line of column headers, one for each column. A blank line is interpreted as the end of the file. The file should therefore be laid out as follows:

```
x[m]           r[m]           SectionIndex  TubeIndex      [TubeSplit]
{Section 0}
{Section 1}
{...}
```

For example, the cylindrical geometry in Sec. 1.2.1 (shown in Fig. 11) may be specified by the geometry file:

x [m]	r [m]	SectionIndex	TubeIndex
0	0.05	0	0
0.25	0.05	0	0
0.5	0.05	0	0

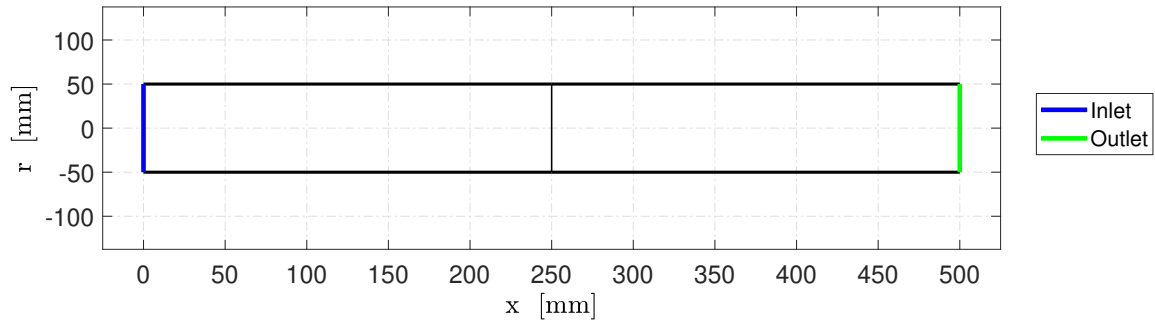


Figure 11: Cylindrical pipe with 3 cylindrical sections.

3.2.1 TubeIndex and TubeSplit

If two consecutive sections have different radius, a radius discontinuity arises at the downstream section. The `TubeIndex` parameter is used to distribute this discontinuity over the whole tube length by dividing it axially into many shorter tubes, which allows a ‘staircase’ approximation to shapes other than cylinders. `OSCILOS` performs this operation internally by interpolating to create more sections within the tube.

A conical profile can be created between two sections of different radius as shown in Fig. 12, using the geometry file:

x [m]	r [m]	SectionIndex	TubeIndex
0	0.05	0	0
0.25	0.05	0	1
0.5	0.1	0	0

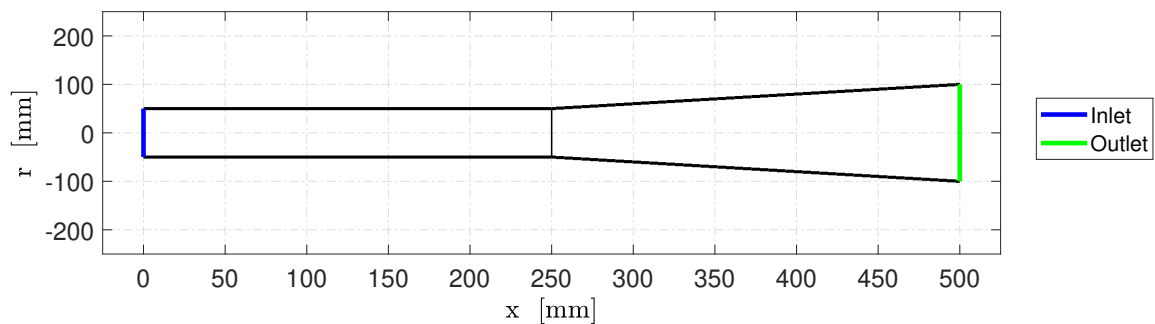


Figure 12: Cylindrical tube ending in conical horn, default interpolation.

The default interpolation can be altered, for example to decrease the number of elements to reduce solution time, as shown in Fig. 13, using the `TubeSplit` column:

x [m]	r [m]	SectionIndex	TubeIndex	TubeSplit
0	0.05	0	0	0
0.25	0.05	0	1	4
0.5	0.1	0	0	0

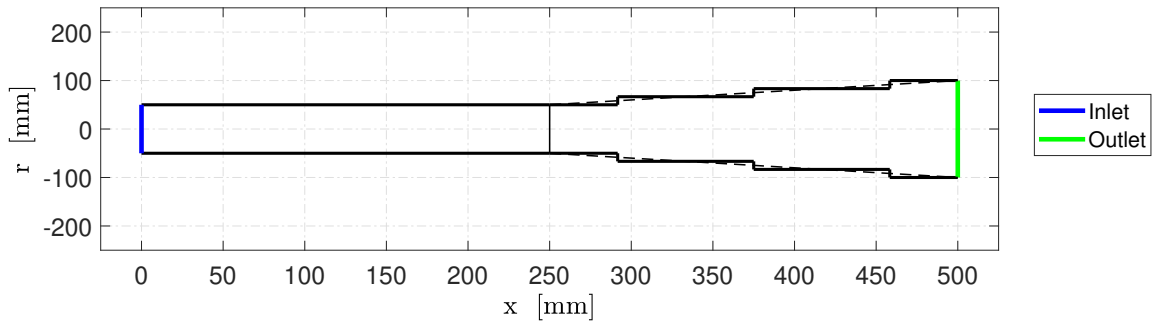


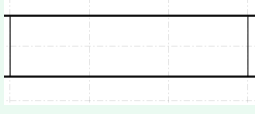

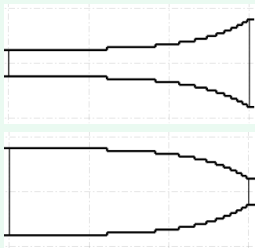
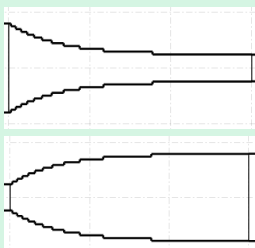
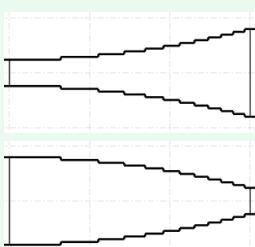
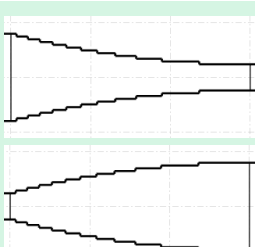

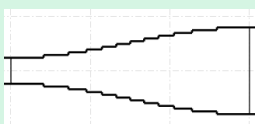
Figure 13: Cylindrical tube ending in conical horn, interpolation defined by `TubeSplit`.

`OSCILOSbrass` supports several interpolated shapes, each with its own `TubeIndex` - these are listed in Table 7. The shapes are interpolated so that the volume enclosed by the ‘staircased’ geometry closely approximates that of the equivalent analytical shape. The number of tubes given by `TubeSplit` are distributed with greater density in regions of greater radial gradient.

3.2.2 Cautionary Notes

- `OSCILOSbrass` requires at least 3 sections (2 tubes) after any tube interpolation is complete - if only 2 sections are given, and the intervening tube is cylindrical (initial section `TubeIndex` = 0, no interpolation), a warning is thrown and a third section is created at the midpoint between the two, with radius the average of the others, and all other parameters equivalent to those of the inlet.
- The line in the file with greatest axial co-ordinate (the last in all of the above examples) is treated as the outlet, and should have `TubeIndex` = 0, because there is no tube after the last section. A warning is thrown if this is not the case.

Table 7: TubeIndex values and corresponding tube shapes

Value	Shape	Example
0	Cylinder, no interpolation, TubeSplit ignored	
1	Cone $P(z) = z$ $P^{-1}(\eta) = \eta$ $x_0 < z < x_N$	
2	Exponential horn, minimum gradient $\left \frac{dP}{dz} \right $ at upstream end $P(z) = e^z$ $P^{-1}(\eta) = \ln(\eta)$ $-5 < z < 0$	
3	Exponential horn, minimum gradient $\left \frac{dP}{dz} \right $ at downstream end $P(z) = e^z$ $P^{-1}(\eta) = \ln(\eta)$ $0 > z > -5$	
4	Parabolic horn, zero gradient $\frac{dP}{dz}$ at upstream end $P(z) = z^2$ $P^{-1}(\eta) = \sqrt{\eta}$ $0 < z < 1$	
5	Parabolic horn, zero gradient $\frac{dP}{dz}$ at downstream end $P(z) = z^2$ $P^{-1}(\eta) = \sqrt{\eta}$ $1 > z > 0$	
6	Sigmoid $P(z) = \frac{1}{1 + e^{-z}}$ $P^{-1}(\eta) = -\ln\left(\frac{1}{\eta} - 1\right)$ $-5 < z < 5$	
7	Sinusoid $P(z) = 1 - \cos(z)$ $P^{-1}(\eta) = \cos^{-1}(1 - \eta)$ $0 < z < \pi$	

4 Outputs

The outputs fall into two categories: initialisation data, and results data. The initialisation outputs are produced before the solution begins, and serve to visualise and record the simulation setup. The results outputs visualise and record the various results yielded by the solver.

4.1 Initialisation

Initialisation data consists of: a record of the configuration parameters used, a copy of the geometry file, and optionally the geometry, mean flow and boundary condition plots. These are saved to `./Outputs/Initialisation/` (or, if `run_name` is specified by the configuration file, `./Outputs/run_name/Initialisation/`).

4.1.1 Geometry

The Geometry plot shows the geometry after any interpolation is complete. The boundaries of the geometry are shown as thick black lines, and the tubes are oriented so that the flow direction is left \rightarrow right. Inlet (left-hand end) and outlet (right-hand end) are shown as thick vertical coloured lines. The internal sections provided in the original geometry file are marked with thin vertical black lines. The ‘staircasing’ created by interpolating over a tube with nonzero `TubeIndex` is left without vertical lines.

The trumpet-like geometry given in Fig. 14 was generated from the following geometry file:

x [m]	r [m]	SectionIndex	TubeIndex	TubeSplit
0	0.02	0	6	0
0.03	0.004	0	1	10
0.1	0.008	0	0	0
0.1	0.01	0	0	0
0.6	0.01	0	2	25
0.9	0.06	0	0	0

The geometry plot is enabled by the `input_plots` configuration option. If `save_figs` and/or `save_pdfs` is enabled, the plot is saved to `./Outputs/Initialisation/Geometry.*` (or `./Outputs/run_name/Initialisation/Geometry.*`, if `run_name` is specified by the config file), where `.*` denotes `.fig` or `.pdf`.

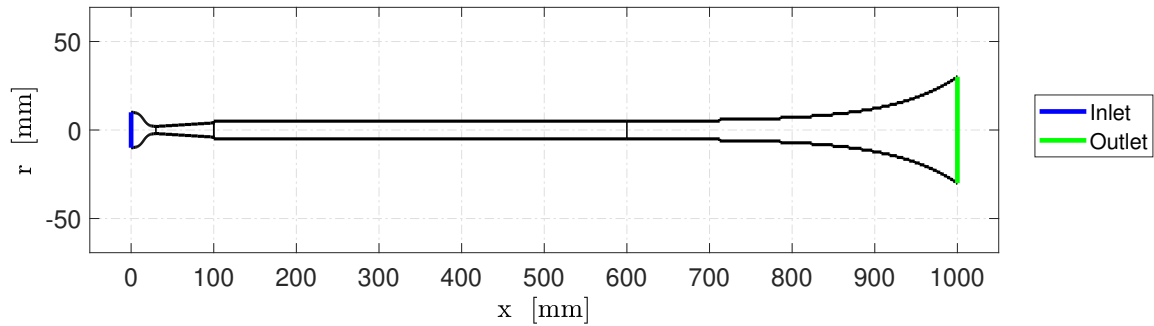


Figure 14: Trumpet-like geometry showing finely and coarsely interpolated sections, and a discontinuity between the mouthpiece backbore and the remainder of the instrument (scale and dimensions not representative of instrument).

4.1.2 Mean Flow

The Mean Flow plot visualises the axial mean velocity and temperature distributions, as shown in Fig. 15. Where a radius discontinuity exists in the geometry file, a gap appears in the mean velocity and temperature curves to reflect the jump condition. Interpolated tubes by contrast are ‘staircased’ in the mean flow plots, so that the effect of the discretisation level on mean flow is evident. The relatively coarse discretisation in Fig. 15 shows the greater influence of step changes at small radii (higher velocities). The finer discretisation in Fig. 16 achieves a smoother variation in mean flow properties.

The mean flow plot is enabled by the `input_plots` configuration option. If `save_figs` and/or `save_pdfs` is enabled, the plot is saved to `./Outputs/Initialisation/Mean_flow.*` (or `./Outputs/run_name/Initialisation/Mean_flow.*`, if `run_name` is specified by the configuration file), where `.*` denotes `.fig` or `.pdf`.

4.1.3 Boundary Conditions

The boundary conditions are formulated as complex reflectances at the inlet and outlet, and specified by the boundary condition configuration parameters (see Sec. 3.1.5). The dependence of these on frequency is visualised over the frequency range specified by the scan range configuration parameters (see Sec. 3.1.2). Gain and phase are visualised for each as shown in Fig. 17.

If the boundary condition is specified by interpolation from data as described in Sec. 3.1.7, the data is also plotted, as shown in Fig. 18.

The boundary conditions plot is enabled by the `input_plots` configuration option. The plot is saved to `./Outputs/Initialisation/Boundary_Conditions.*` (or, if `run_name` is specified

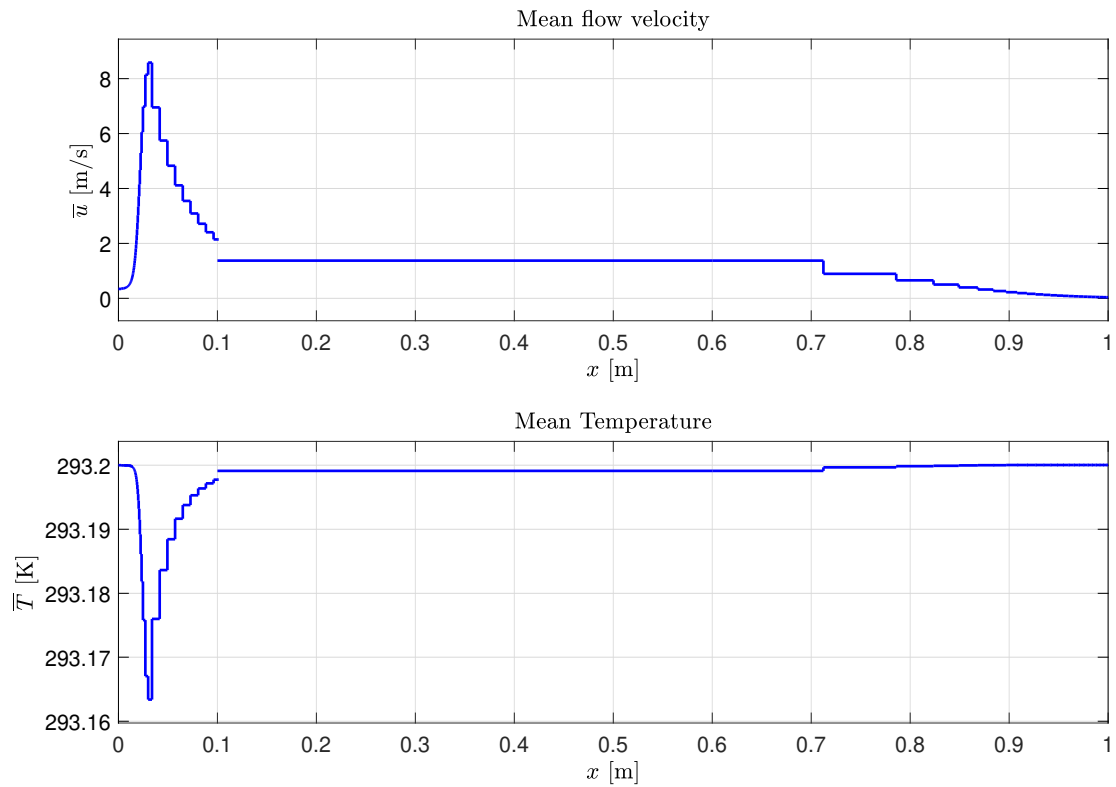


Figure 15: Mean Flow plots for trumpet-like geometry with same discretisation level as in Fig. 14.

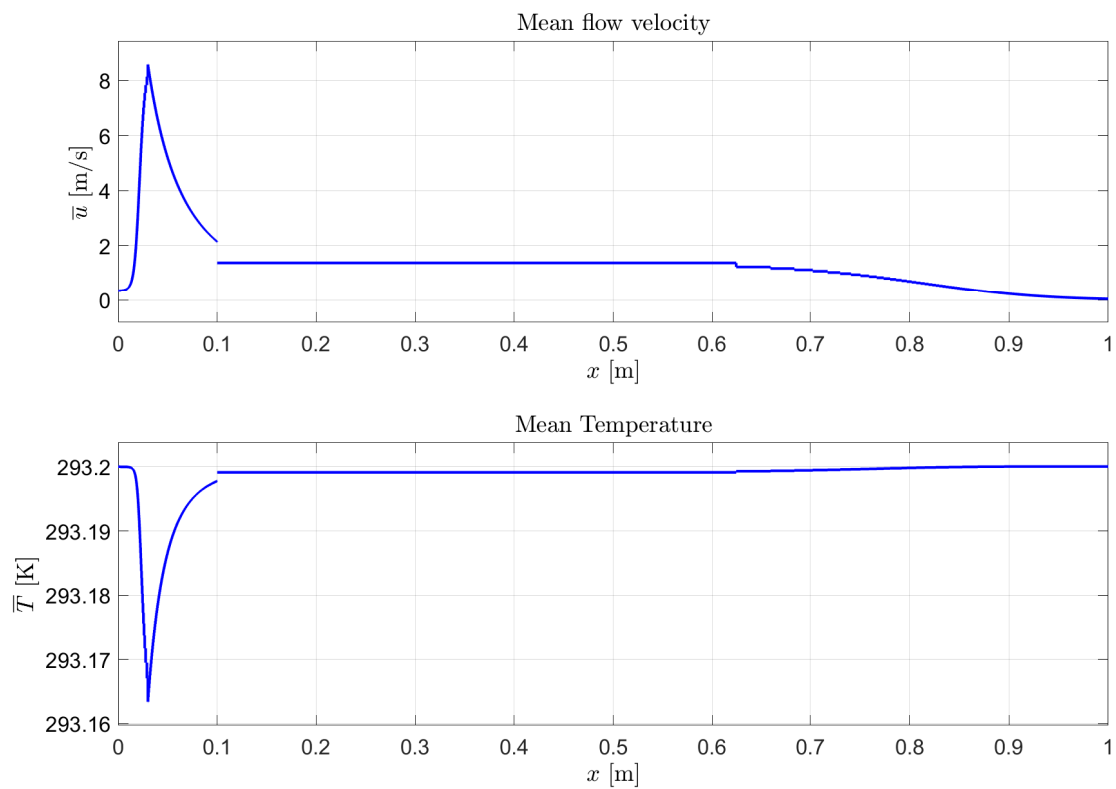


Figure 16: Mean Flow plots for trumpet-like geometry with a finer discretisation level than Fig. 14, with TubeSplit set to 200 for each interpolated shape.

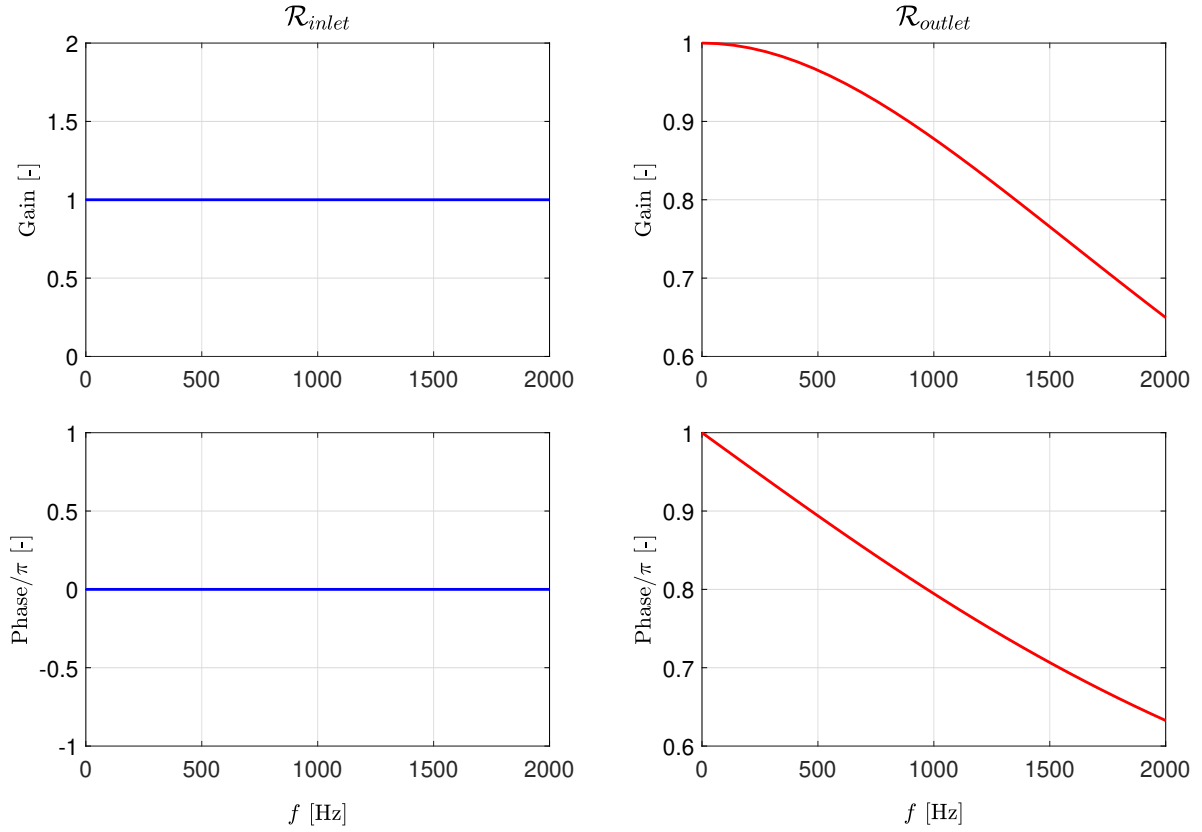


Figure 17: Boundary condition plots: closed-end boundary condition (BC2) at the inlet and Levine-Schwinger condition (BC11) at the outlet.

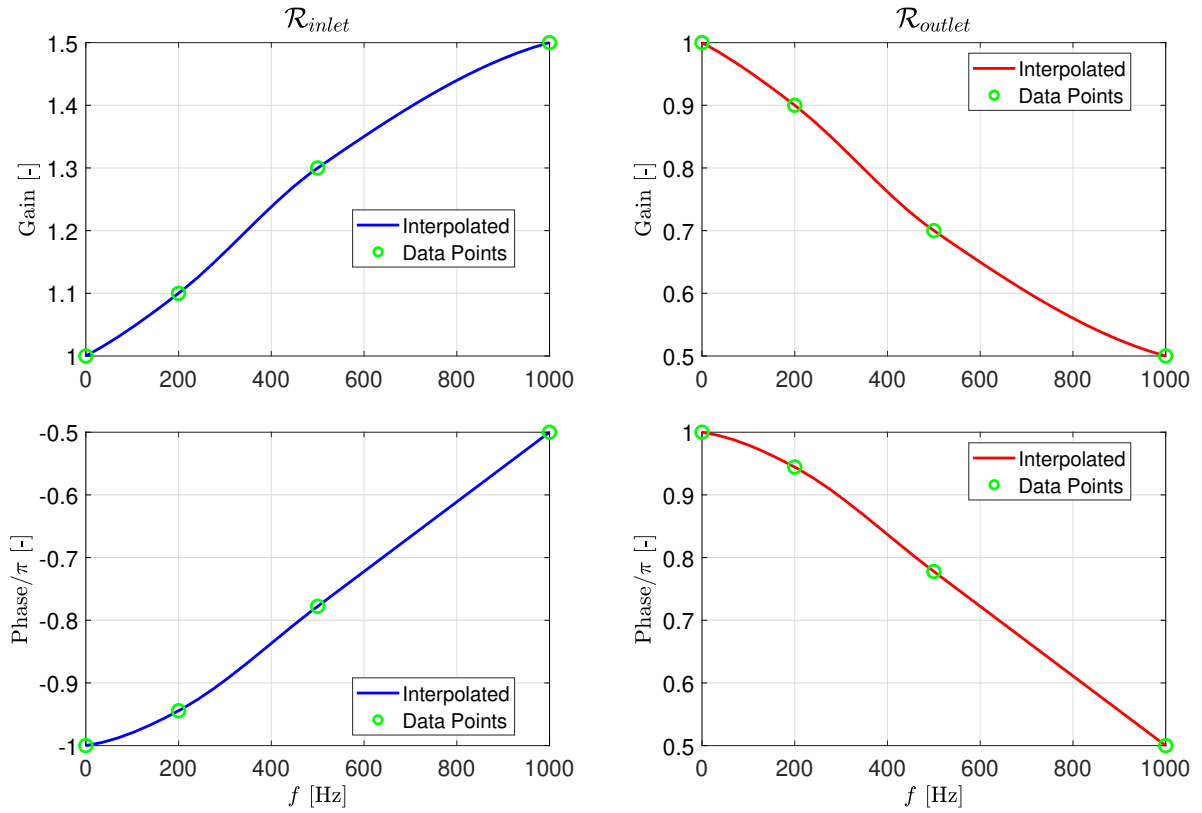


Figure 18: Boundary condition plots defined by user data (BC9) showing interpolations for inlet and outlet.

by the configuration file, `./Outputs/run_name/Initialisation/Boundary_Conditions.*`) if `save_figs` and/or `save_pdfs` is enabled, where `.*` denotes `.fig` or `.pdf`.

4.2 Results

Results data includes: a text file containing the modes found, and optionally a log file, the eigenvalue contour map, the equivalent fundamental pitch plot, and mode shape plots. These are saved to `./Outputs/Results/` (or `./Outputs/run_name/Results/` if `run_name` is specified by the configuration file).

4.2.1 Eigenvalue List File

The eigenvalue list file is the only non-optional output, and contains a summary of the solver results in whitespace-separated, human-readable form. It comprises 5 columns, labelled:

Mode number	Frequency [Hz]	Growth rate [1/s]	EFP [cents]	Note
-------------	----------------	-------------------	-------------	------

The **Mode number** column contains a list of ascending integers describing the modes in order of ascending frequency. For more discussion of **Frequency** and **Growth rate**, see Sec. 4.2.2. For an explanation of equivalent fundamental pitch, or **EFP**, see Sec. 4.2.3. The **Note** column gives the name of the nearest chromatic note to the mode frequency in scientific pitch notation with pitch standard $A4 = 440$ Hz, and the deviation in cents from that note where positive deviation means the mode is sharper.

The eigenvalue list file is saved to `./Outputs/Results/Eigenvalues.txt` (or, if `run_name` is specified by the configuration file, `./Outputs/run_name/Results/Eigenvalues.txt`).

4.2.2 Eigenvalue Contour Map

The eigenvalue contour map (Fig. 19) visualises the residual in the transmission-line calculation. The eigenvalues of the system, and hence the acoustic modes, lie at the minima of this function. The plot represents the Laplace variable space, in the range specified by the scan range configuration parameters, with the real part corresponding to growth rate on the abscissa, and the imaginary part converted to [Hz] on the ordinate. The resolution of the plot (number of residuals calculated) is 10 times the `num_freq` and `num_GR` parameters, in the respective axes.

The eigenvalues previously found by the solver are overlaid on the contour plot as white stars. In some cases, e.g. where `num_freq` is insufficient, the solver may miss one or more eigenvalues.

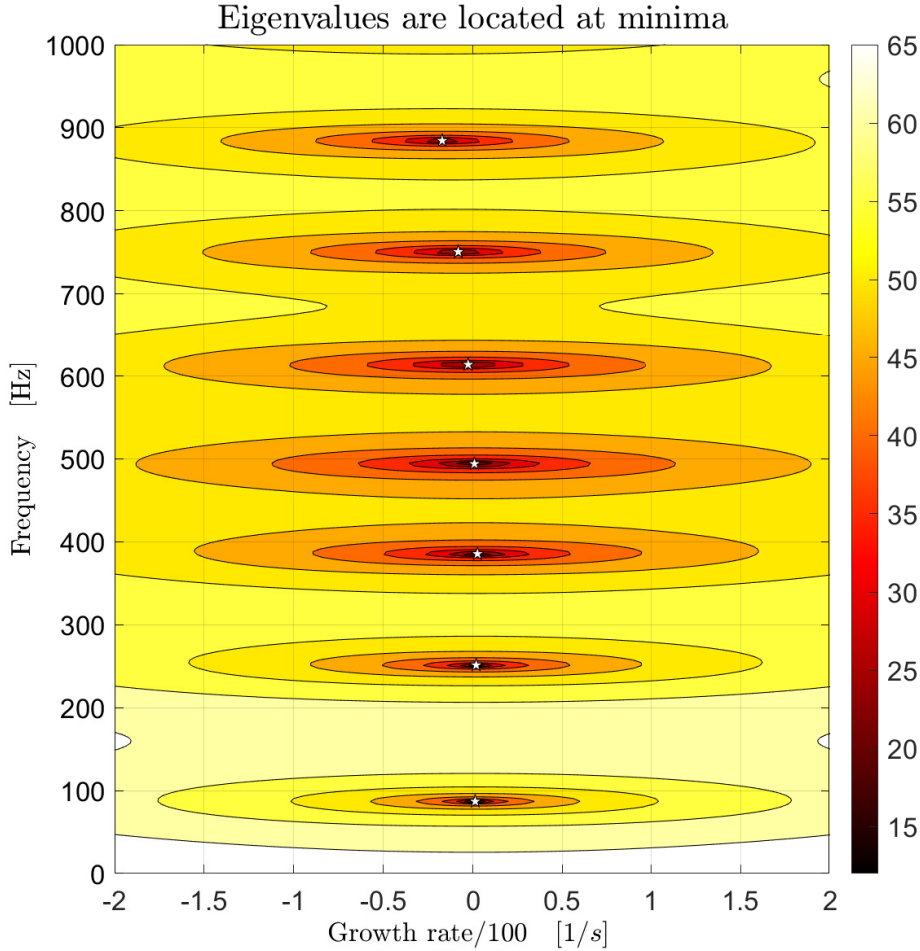


Figure 19: Eigenvalue Contour Map, showing eigenvalues (white stars) located at minima.

If this happens, the contour plot will exhibit minima which have no white star, as illustrated in Fig. 20, indicating that the solver was poorly configured and the calculation should be repeated if accurate mode numbers are desired. In most cases, increasing `num_freq` will solve the problem, but the user may also increase `num_GR` if there is large variation in growth rate.

The eigenvalue contour plot is enabled by the `contour_plot` configuration option. It is saved to `./Outputs/Results/Eigenvalues_map.*` (or, if `run_name` is specified by the configuration file, `./Outputs/run_name/Results/Eigenvalues_map.*`), if `save_figs` and/or `save_pdfs` is enabled, where `.*` denotes `.fig` or `.pdf`.

Calculating the residuals for every point on the eigenvalue plot requires the solver to repeat the transmission line calculation $(10 \times \text{num_freq}) \times (10 \times \text{num_gr})$ times, which for many-sectioned geometries takes considerable computational time. It is recommended to use the contour plot to verify that no eigenvalues were missed for every combination of geometry, boundary condition types and scan range settings. However, if multiple runs on equivalently-shaped

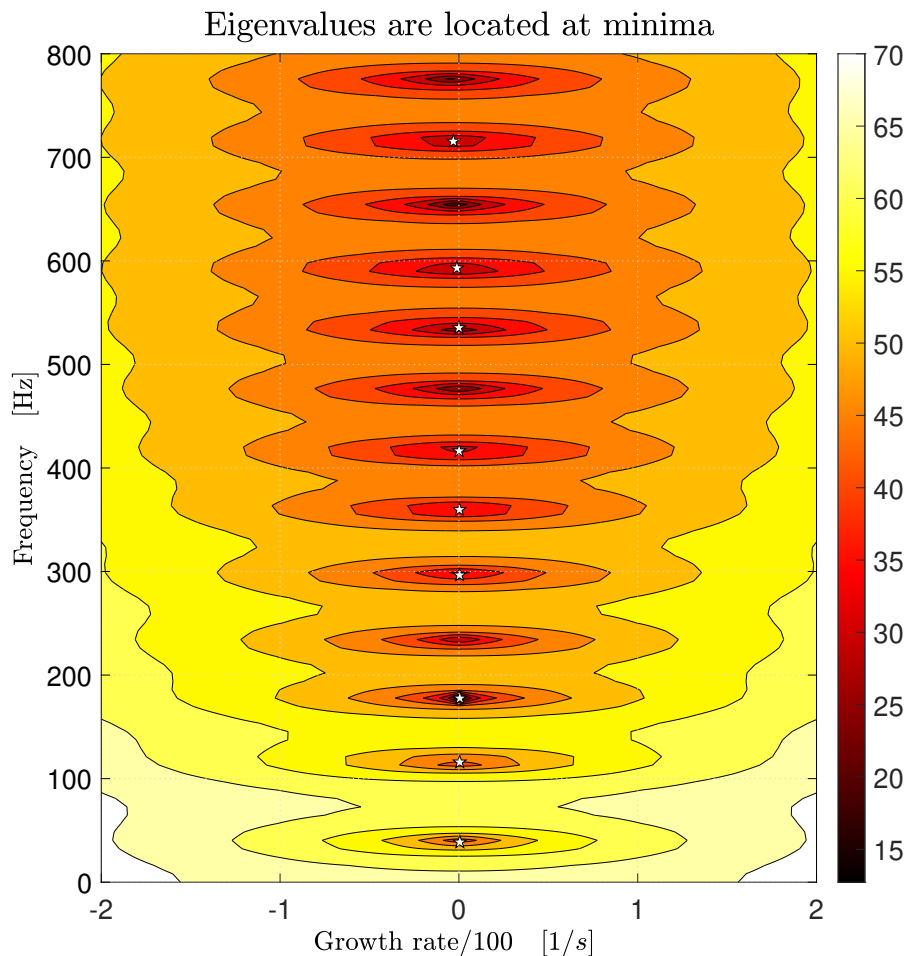


Figure 20: Eigenvalue Contour Map for realistic trombone geometry, created with insufficient `num_freq` such that modes 4, 8, 11 and 13 have been missed.

geometry are to be conducted, the computation time for each iteration may be vastly reduced by setting `contour_plot` to `FALSE`.

4.2.3 Equivalent Fundamental Pitch

Equivalent Fundamental Pitch, as defined by Chick et al [14], allows a quantitative measure of the inharmonicity of an instrument by evaluating the deviation of the ‘equivalent fundamental’ corresponding to each mode, from some reference. It is given by Eq. 17, where f_i is the frequency of the i th mode, and F is the reference fundamental pitch, conventionally taken as $f_4/4$, the note to which brass players often tune their instruments. The unit of this definition is the [cent], equal to $1/100^{\text{th}}$ of a semitone, or a frequency ratio of $^{1200}\sqrt{2}$.

$$\text{EFP}(f_i) = \frac{1200}{\log(2)} \log \left[\frac{f_i}{iF} \right] \quad (17)$$

In musical acoustics literature, EFP is typically plotted for a sequence of modes as in Fig. 21.

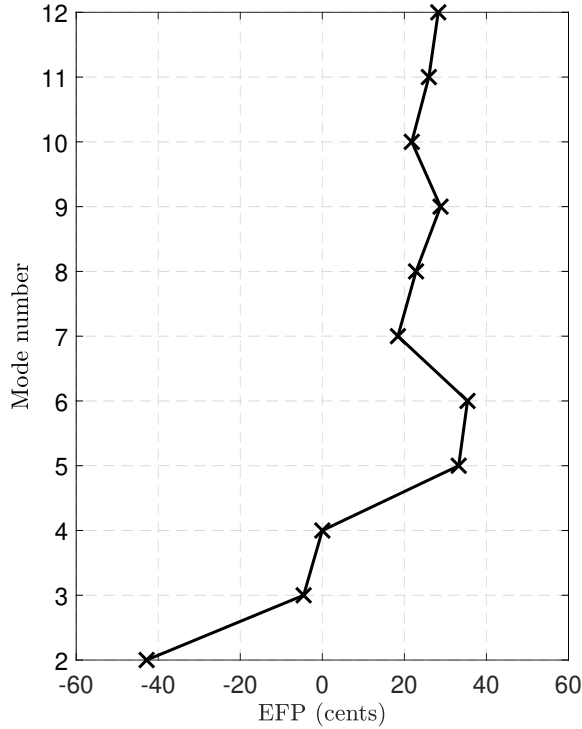


Figure 21: Equivalent Fundamental Pitch (EFP) plot for trombone geometry from Bilbao and Chick [6].

By default, `OSCILOSbrass` takes $F = f_4/4$, as is conventional. If there are fewer than 4 modes found overall, the value $F = f_i/i$ is used where i is the total number of modes. The fundamental mode of most brass instruments is far from the equivalent fundamental of all other modes and is therefore usually excluded from the EFP plot. For geometries whose modes are inherently not harmonically distributed, the EFP plot does not give meaningful information.

If the solver misses eigenvalues, the EFP plot is inaccurate for any modes above those which were missed, because it relies on an accurate mode number i for each frequency. Approaches to rectify this are discussed in Sec. 4.2.2.

4.2.4 Mode Shapes

`OSCILOSbrass` calculates the mode shape, i.e. the axial distribution of acoustic pressure and velocity magnitude, for each mode. These are visualised by the mode shape plots, as in Fig. 22. The number of mode shapes plotted is controlled by the `plot_modes` configuration parameter, and the number of modes found overall within the scan range - whichever is the lowest.

The mode shape plots may be disabled by setting `plot_modes` to 0. If `save_figs` and/or `save_pdfs` is enabled, the mode shape plots are saved to `./Outputs/Results/Mode_n.*` (or,

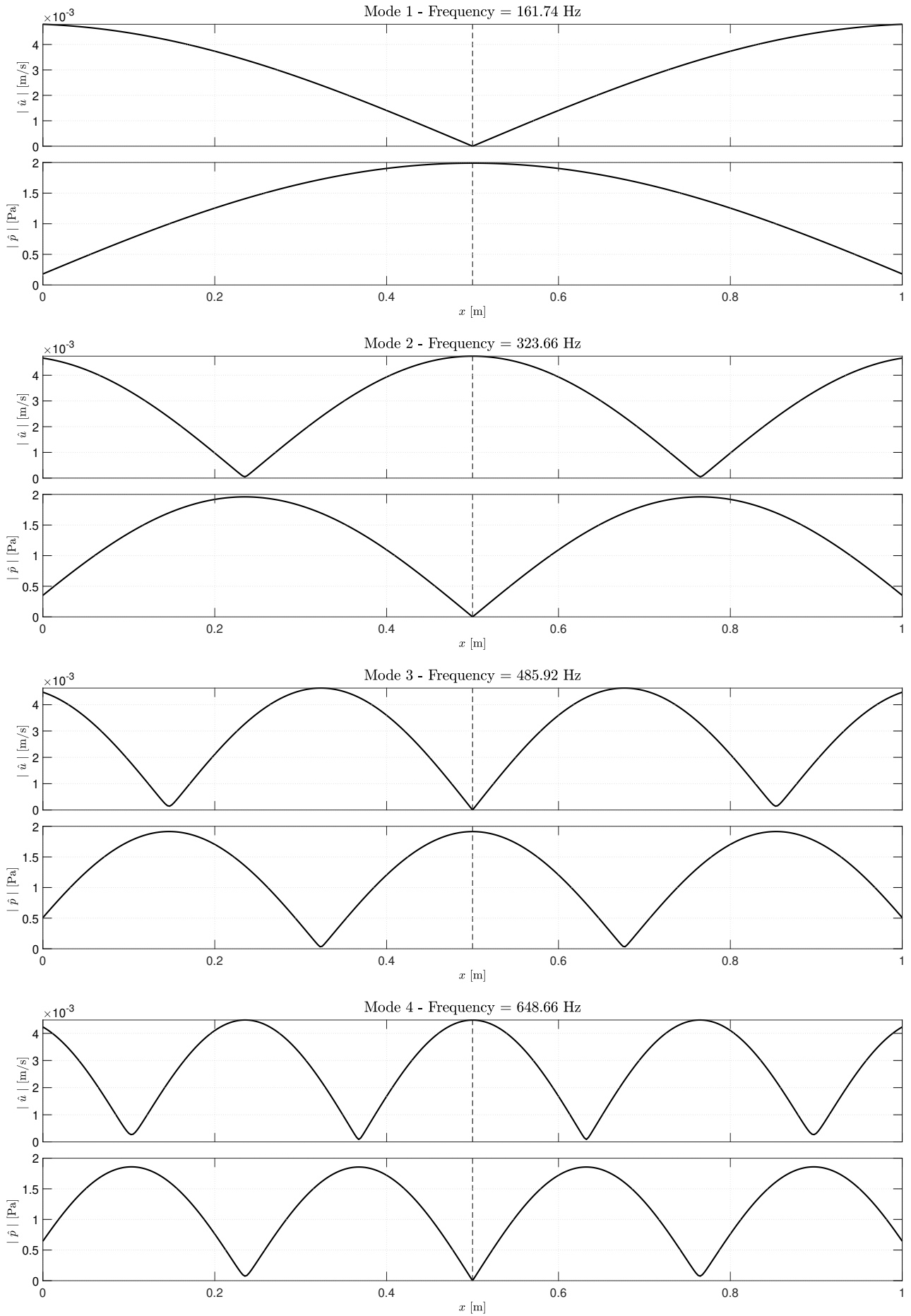


Figure 22: Mode shape plots: acoustic velocity and pressure for the first 4 modes of an open cylinder, $\varnothing 100$ mm and 1 m in length, with the Levine-Schwinger boundary condition applied at each end.

if `run_name` is specified by the configuration file, `./Outputs/run_name/Results/Mode_n.*`), where `.*` denotes `.fig` or `.pdf`, and `Mode_n` is replaced by `Mode_1`, `Mode_2` etc.

Where the geometry contains radial discontinuities, corresponding discontinuities in the velocity distribution will be evident due to the area change. The evolution of velocity over each section may be increasing or decreasing depending on the phase of the velocity at any given point, and this may be either concordant with or opposite to the direction of the velocity jump at the interface.

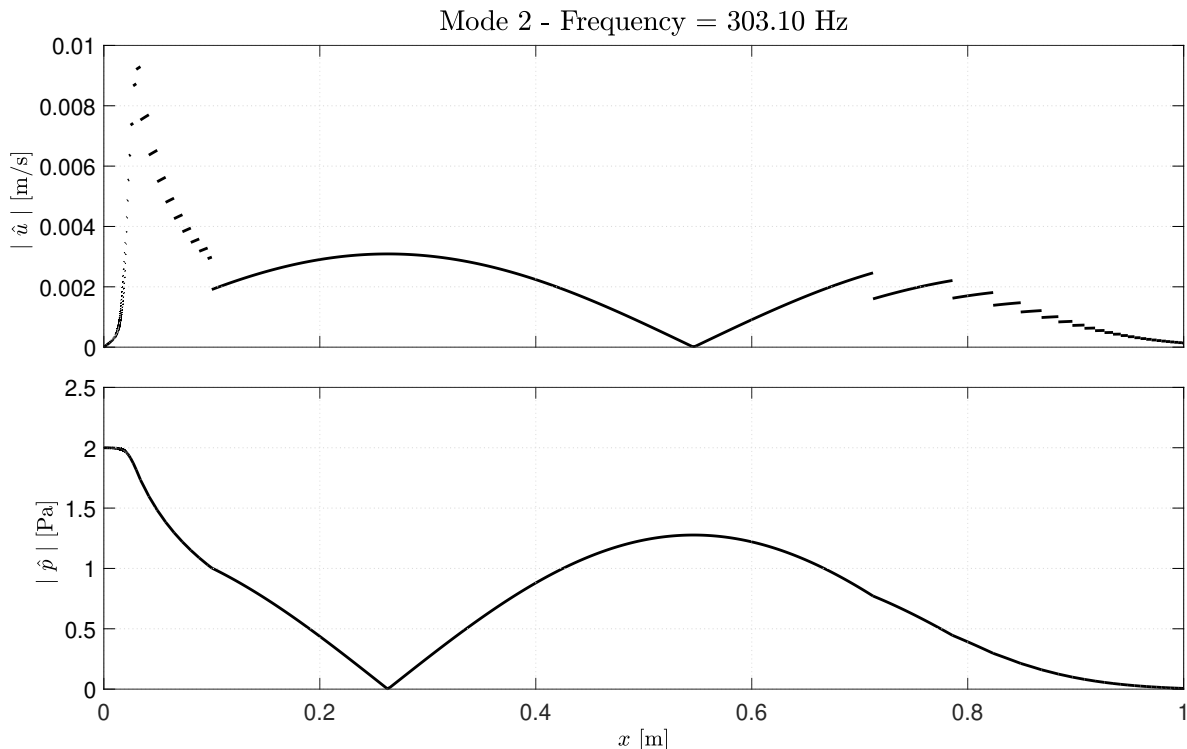


Figure 23: Mode shape for 2nd mode of trumpet-like geometry showing discontinuities in velocity plot in regions of large radius gradient due to staircasing effect and 1D approximation.

One relatively extreme example is given in Fig. 23, generated by the coarsely-discretised trumpet-like geometry in Fig. 14. Here the evolution of the acoustic velocity opposes that of the mean flow velocity, creating an unusual terracing in the mouthpiece, backbore and bell regions. This behaviour naturally does not exist in reality, arising here from the staircase approximation to smoothly varying bore shapes. The velocity discontinuity between the backbore and cylindrical section, despite resulting from a physical radius discontinuity, is also unphysical - an artefact of the 1D approximation. Nevertheless, the mode shapes provide valuable information about the location of pressure and velocity nodes, and are useful for validating the configuration of boundary conditions. The pressure is of course continuous throughout.

5 Function Calls to OSCILOS_brass()

OSCILOS_{brass} is wrapped in a single function which may be invoked from any other script. This provides a convenient abstraction for use in parameter sweep or optimisation routines.

5.1 Function Definition

```
eigenvalues = OSCILOS_brass(Name, Value, ... )
```

Arguments: all parameters detailed in Sec. 3.1 may be passed to the OSCILOS_brass function in Name, Value optional argument pairs. These values override those parsed from the config file. If no arguments are provided, the solver is configured solely by the config file.

Return Value: an array of complex eigenvalues in Laplace variable form.

5.2 Code Example

A short MATLAB example, using the solver to calculate the effect of temperature on the pitch of the 4th mode of a trumpet-like geometry, follows. It is assumed that the solver is set up exactly as the horn example in Sec. 1.2.2 (equivalent Config.txt and Geometry.txt files).

```
%% Script to demonstrate function calls to OSCILOS_brass
clear all
addpath('C:/Users/amacl/Downloads/OSCILOS_brass_release_1/');

T = (-20:5:30) + 273.2; % Temperatures in K from -20°C to 30°C

for i = 1:length(T) % For each Temperature value
    EigVals{i} = OSCILOS_brass('T1', T(i), ...
        ... % Name of output file (retain results for each iteration)
        'eig_filename', "EigVals_T_" + num2str(T(i)) + ".txt", ...
        ... % Suppress graphical outputs to speed up calculation
        'no_popups', true, ...
        'cl_out', false, ...
        'log_out', false);
end

% Extract 4th eigenvalues, convert to frequency in Hz, and plot
FreqsMode4 = imag( cellfun( @(x) x(4), EigVals) )./2./pi;
plot(T - 273.2, FreqsMode4, 'g', 'LineWidth', 2);
xlabel('T (°C)'); ylabel('f_4 (Hz)');
```

The output produced by this routine is shown in Fig. 24

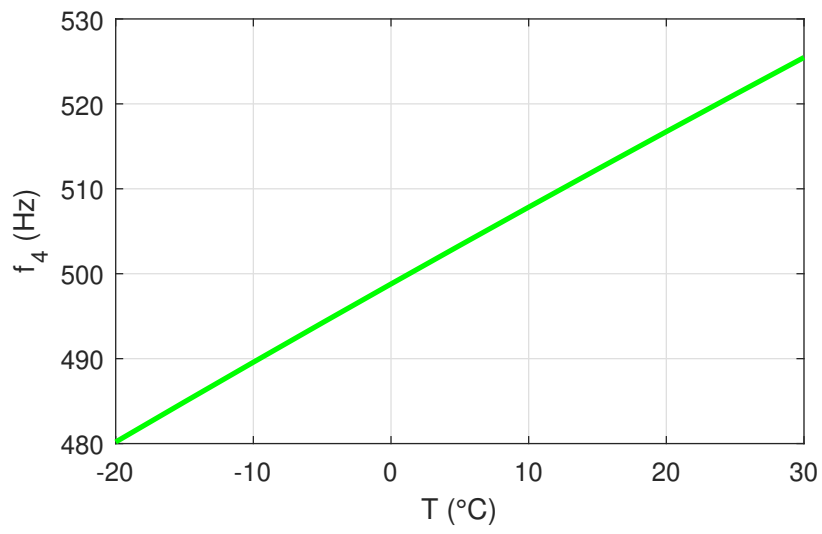


Figure 24: Temperature dependence of frequency of 4th mode f_4 for example in Sec. 1.2.2.

References

- [1] J. Li, D. Yang, C. Luzzato, and A.S. Morgans. OSCIOS Report. Technical report, Imperial College London, London, 2017.
- [2] H. Levine and J. Schwinger. On the Radiation of Sound from an Unflanged Circular Pipe.pdf. Physical Review, 73(4):137–153, 1948. doi:10.1103/PhysRev.73.383.
- [3] J. Backus and T. C. Hundley. Harmonic Generation in the Trumpet. The Journal of the Acoustical Society of America, 49(2B):509–519, 2 1971. doi:10.1121/1.1912380.
- [4] A. C. P. Braden. Bore optimisation and impedance modelling of brass musical instruments. PhD thesis, University of Edinburgh, 2007. URL: http://www.acoustics.ed.ac.uk/wp-content/uploads/Theses/Braden_Alistair___PhDThesis_UniversityOfEdinburgh_2006.pdf.
- [5] S. Candel and T. Poinso. Tutorial On Acoustics, 1988.
- [6] S. Bilbao and J. Chick. Finite difference time domain simulation for the brass instrument bore. The Journal of the Acoustical Society of America, 134(5):3860–3871, 2013. doi:10.1121/1.4822479.
- [7] A. Hirschberg, J. Gilbert, R. Msallam, and A. P. J. Wijnands. Shock waves in trombones. The Journal of the Acoustical Society of America, 99(3):1754–1758, 3 1996. doi:10.1121/1.414698.
- [8] A. P. Dowling and S. R. Stow. Acoustic Analysis of Gas Turbine Combustors. Journal of Propulsion and Power, 19(5):751–764, 2003. URL: <http://arc.aiaa.org>, doi:10.2514/2.6192.
- [9] A. N. Norris and I. C. Sheng. Acoustic radiation from a circular pipe with an infinite flange. Journal of Sound and Vibration, 135(1):85–93, 11 1989. doi:10.1016/0022-460X(89)90756-6.
- [10] O. K. Mawardi. On the Generalization of the Concept of Impedance in Acoustics. Journal of the Acoustical Society of America, 23(5):571–576, 1951. doi:10.1121/1.1906806.
- [11] F. P. Mechel. Formulas of Acoustics. Springer, 2008.
- [12] K. M. Ivanov-Schitz, S. N. Rscherkin, and K. A. Welischanina. Über die Wirkung des Schallabsorbers, der eine schwingende Oberfläche bedeckt, auf die Schallabstrahlung. Acustica, 13(6):403–406, 1963.
- [13] Modified Akima piecewise cubic Hermite interpolation - MATLAB makima - MathWorks United Kingdom. URL: <https://uk.mathworks.com/help/matlab/ref/makima.html>.
- [14] J. P. Chick, C. Lumb, and D. M. Campbell. Passive acoustic characteristics and intonation problems of modern orchestral horns. In Proceedings of the ISMA2004, Nara, Japan, 2004.