Organ Pipe Sound Synthesis with Finite Difference Schemes

Sondre Sigstad Wikberg, s1863042@sms.ed.ac.uk
Supervisor: Dr. Charlotte Desvages, charlotte.desvages@ed.ac.uk
School of Mathematics, University of Edinburgh

Vacation Scholarship: School of Mathematics & College of Science and Engineering
(Dated: 18th August 2022)

In this text, we present a finite difference scheme for Webster's Horn Equation modelling the propagation of sound waves in flue organ pipes. Throughout, emphasis has been put on explanations accessible for mathematics students at an undergraduate level. We present a model and associated code capable of emulating sound propagation inside a tube of any shape, provided the cross-section is circular. Energy analysis is used to find conditions under which the solution approximated by the model remains stable.

BACKGROUND

There are several ways to computationally synthesise and approximate the sounds produced by physical musical instruments. Physical models are reliant on descriptions of vibrations: in the instrument components and the air surrounding them ([1]). Given a deviation from resting state, assumptions about configuration, stiffness, and the properties of air lead naturally to differential equations (of one kind or another) describing these vibrations. The degree, then, to which a model emulates the behaviour of an instrument is dependent on the accuracy of assumptions and "exactness" of computed solutions. With accuracy comes complexity, however, and a balance must be struck between realistic models and computational cost. Finite difference schemes are advantageous due to their simple implementation and the ease with which stability conditions may be investigated.

I. PRELIMINARIES

The 1D wave equation

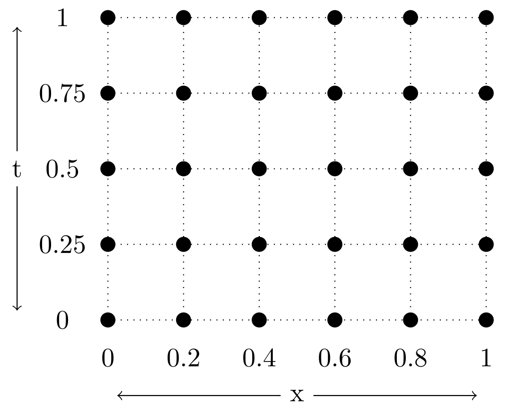
The starting point for sound synthesis schemes in this text is the 1D wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

Here, $u(x,t)$ denotes some parameter which varies with time, t, and space, x. The constant c is the wave speed. The equation is called one-dimensional because variations are only considered along a single spatial domain (i.e. $x \in \mathbb{R}$). For a general derivation of this all-important equation to physics see [1]. Later, we shall see that it is useful for modelling the vibrations of air inside a pipe.

Finite Difference Schemes

Finite difference schemes approximate solutions to differential equations iteratively over one of the independent variables (usually the time variable). First, the domain of each independent variable is discretised into a countable number of points. When combined, these points form a "grid" with dimensions equal to the number of variables. In the case of the 1D wave equation with a spatial domain of $[0,1]$ and time domain of $[0,1]$, the spatial domain may be discretised as $x \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ and the time domain as $t \in \{0, 0.25, 0.5, 0.75, 1\}$. We index these points as $l \in \{0,1,2,3,4,5\}$ and $n \in \{0,1,2,3,4\}$ for the spatial- and time domain, respectively. The associated "grid" of points at which the discrete dependent variable $\hat{u}_l^n$ (which approximates u) must be calculated is accordingly:



Second, initial- and boundary conditions are set according to assumptions about the system at hand. Third, finite difference approximations are plugged in for each of the derivatives which appear in the equation. The denominators appearing in these approximations correspond, respectively, to the distance between grid points in the discretisation. Finally, the terms of the finite differences are manipulated algebraically to yield an expression for the next (time)-step in terms of values already calculated. As a preliminary example, we plug in spatial and temporal second-order central difference approximations in the 1D wave equation at time point n and spatial point l. Suppressing the hat on $\hat{u}_l^n$, this yields

$$\frac{1}{k^2}(u_l^{n+1} - 2u_l^n + u_l^{n-1}) = c^2\frac{1}{h^2}(u_{l+1}^n - 2u_l^n + u_{l-1}^n) \quad (2)$$

This expression is manipulated algebraically to arrive at

$$u_l^{n+1} = c^2\frac{k^2}{h^2}(u_{l+1}^n + u_{l-1}^n) + 2(1 - c^2\frac{k^2}{h^2})u_l^n - u_l^{n-1} \quad (3)$$

In this implementation, it is assumed that the values u_{l+1}^n, u_l^n, u_{l-1}^n, and u_l^{n-1} are known, either as a result of initial- and boundary conditions, or as a result of previous calculations. It follows that u_l^{n+1} must be computed for all values of l before the next step in the iteration. Note that, for the discretisation seen above, $k = 0.25$ and $h = 0.2$, the difference between time- and spatial points, respectively.

Operators

Other difference approximations than the second-order central may be used to discretise the 1D wave equation, and indeed any other partial differential equation. In order to simplify notation and calculations, it is very useful to identify a given difference approximation with a linear operator. These operators, in turn, are decomposed into shifting operators. For example the spatial second-order difference approximation applied to the discrete variable u_l^n may be written with help from operator δ_{xx} as

$$\begin{aligned}\delta_{xx}u_l^n &= \frac{1}{h^2}(e_{x+} - 2I + e_{x-})u_l^n \\ &= \frac{1}{h^2}(u_{l+1}^n - 2u_l^n + u_{l-1}^n) \quad \approx \frac{\partial^2 u}{\partial x^2}\end{aligned}$$

The shifting operators e_{x+} and e_{x-} send u_l^n to u_{l+1}^n and u_{l-1}^n, respectively, while I is simply an identity operator. Similarly, the temporal second-order difference approximation may be written with help from operator δ_{tt} as

$$\begin{aligned}\delta_{tt}u_l^n &= \frac{1}{k^2}(e_{t+} - 2I + e_{t-})u_l^n \\ &= \frac{1}{k^2}(u_l^{n+1} - 2u_l^n + u_l^{n-1}) \approx \frac{\partial^2 u}{\partial t^2}\end{aligned}$$

It can be easily shown using Taylor series that $\delta_{xx}u_l^n$ and $\delta_{xx}u_t^n$ are second order accurate approximations to $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial t^2}$, respectively. Hence we say that the operators $\delta_{xx}u_l^n$ and $\delta_{xx}u_t^t$ are second-order accurate. From the shifting operators defined above, it follows immediately that the familiar first-order forward, backward and central difference operators may be defined, respectively, as

$$\left.\begin{array}{ll}\delta_{x+} = \frac{1}{h}(e_{x+} - I) & \Rightarrow \delta_{x+}u_l^n = \frac{1}{k}(u_{l+1}^n - I) \\ \delta_{x-} = \frac{1}{h}(I - e_{x-}) & \Rightarrow \delta_{x-}u_l^n = \frac{1}{k}(I - u_{l-1}^n) \\ \delta_{x\cdot} = \frac{1}{2h}(e_{x+} - e_{x-}) & \Rightarrow \delta_{x\cdot}u_l^n = \frac{1}{2k}(u_{l+1}^n - u_{l-1}^n)\end{array}\right\} \approx \frac{\partial u}{\partial x}$$

in the spatial case. Equivalent operators δ_{t+}, δ_{t-}, and $\delta_{t\cdot}$ exist for the temporal domain, albeit with h replacing k. The first-order backward and forward difference operators are first order accurate, while the centred ones are second order accurate. A less familiar, but very useful group of operators are the averaging operators. In the spatial case, the first order averaging operators may be defined as

$$\left.\begin{array}{ll}\mu_{x+} = \frac{1}{2}(e_{x+} + I) & \Rightarrow \mu_{x+}u_l^n = \frac{1}{2}(u_{l+1}^n + I) \\ \mu_{x-} = \frac{1}{2}(I + e_{x-}) & \Rightarrow \mu_{x-}u_l^n = \frac{1}{2}(I + u_{l-1}^n) \\ \mu_{x\cdot} = \frac{1}{2}(e_{x+} + e_{x-}) & \Rightarrow \mu_{x\cdot}u_l^n = \frac{1}{2}(u_{l+1}^n + u_{l-1}^n)\end{array}\right\} \approx u(x,t)$$

Again, equivalent operators μ_{t+}, μ_{t-}, and $\mu_{t\cdot}$ exist for the temporal domain. Using Taylor series, it can be shown that μ_{t+}, μ_{t-}, μ_{x+}, μ_{x-} are first order accurate, while $\mu_{x\cdot}$, and $\mu_{t\cdot}$ are second order accurate. With these definitions out of the way, it is important to note briefly the relationships that exist between these operators and how they interact. Due to the linearity of their definitions, the given operators may be applied in any order to yield the same approximation. However, an operator outside a parenthesis signifies an operation on the entire expression inside. With this is mind, note for example that

$$\begin{aligned}&(\delta_{t\cdot}u_l^n)(\delta_{tt}u_l^n) \\ =&\frac{1}{2k^3}((u_l^{n+1})^2 - 2(u_l^n)(u_l^{n+1}) + 2(u_l^n)(u_l^{n-1}) - (u_l^{n-1})^2) \\ =&\frac{1}{2k^3}((u_l^{n+1})^2 - 2(u_l^n)(u_l^{n+1}) + 2(u_l^n)^2 \qquad (4) \\ &- 2(u_l^n)^2 + 2(u_l^n)(u_l^{n-1}) - (u_l^{n-1})^2) \\ =&\delta_{t+}(\frac{1}{2}(\delta_{t-}u_l^n)^2)\end{aligned}$$

It may be shown in a similar manner that

$$u_l^n e_{t-}u_l^n = (\mu_{t-}u_l^n)^2 - \frac{k^2}{4}(\delta_{t-}u_l^n)^2 \quad (5)$$

II. THE HORN EQUATION

The following derivation for the Horn Equation is adapted from that presented in [2]. We envision a tube stretching along spatial dimension x with varying cross-sectional area $S(x)$. At some position and time (x,t), the pressure is given by $p = P_0 + \Delta p$, and the density by $\rho = \rho_0 + \Delta\rho$, where P_0, ρ_0 are average pressure and air density of the surroundings, respectively. Air-particles pass through the tube at a velocity v in the positive x-direction. The cross-sectional area S does not vary with time. We consider what occurs in the slice of the tube between x and $x + \epsilon$ over a small period in time $[t, t+\tau]$. Critically, we assume that the density ρ is spatially uniform within the slice and that the particle velocity v is time-invariant over $[t, t+\tau]$ at the slice's endpoints. Since density $\rho = m/V$, the mass of air entering the slice is

$$\rho S v \tau \quad (6)$$

The mass of air exiting is approximately

$$\begin{aligned}\rho\Big(S+\frac{dS}{dx}\epsilon\Big)\Big(v+\frac{\partial v}{\partial x}\epsilon\Big)\tau &\approx \rho\Big(Sv+S\frac{\partial v}{\partial x}\epsilon+\frac{dS}{dx}v\epsilon\Big)\tau\\ &= \rho\Big(Sv+\frac{\partial(Sv)}{\partial x}\epsilon\Big)\tau\end{aligned} \tag{7}$$

where the approximation signifies that the term with factor ϵ^2 has been neglected. Subtracting (6) from (7) yields

$$\rho S\frac{\partial(Sv)}{\partial x}\epsilon\tau \tag{8}$$

indicating a discrepancy in the mass entering and exiting the slice. The reduction of mass in the slice resulting from changes in the density is approximately

$$-\frac{\partial\rho}{\partial t}\epsilon\tau \tag{9}$$

In order to satisfy conservation of mass it must hold that

$$\begin{aligned}\rho\frac{\partial(Sv)}{\partial x}\epsilon\tau &= -S\frac{\partial\rho}{\partial t}\epsilon\tau\\ \Rightarrow \rho\frac{\partial(Sv)}{\partial x} &= -S\frac{\partial\rho}{\partial t}\end{aligned} \tag{10}$$

Consider now a small *cylinder* of air inside the tube, stretching in the x-direction, with length $\hat{\epsilon}$ and cross-sectional area $S(x)$. At time t, air-pressure acts on the cylinder at points x and $x+\hat{\epsilon}$. The pressure difference between points x and $x+\hat{\epsilon}$ produces a resultant force of approximately

$$-S\frac{\partial p}{\partial x}\hat{\epsilon} \tag{11}$$

in the rightward direction. By $F=ma$, it follows that

$$-S\frac{\partial p}{\partial x}\hat{\epsilon} = \rho S\frac{dv}{dt}\hat{\epsilon} = \rho S\Big(\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}\frac{dx}{dt}\Big)\hat{\epsilon} \approx \rho S\frac{\partial v}{\partial t}\hat{\epsilon} \tag{12}$$

According to [2], this approximation is justified because of the "small [particle] velocities met in phonetics problems".
Notice now, that the rate at which mass passes through a point in the aforementioned cylinder is given by

$$\frac{dm}{dt} = \frac{d}{dt}(\rho S\hat{\epsilon}) = \rho S v \tag{13}$$

Given this formula and the so-called continuity equation, it may be shown with relative ease ([3]) that

$$c^2\frac{\partial\rho}{\partial t} \approx \frac{\partial p}{\partial t} \tag{14}$$

where c is the speed of sound in the air. Assume now that the particle velocity v is the spatial derivative of a potential function Ψ, such that

$$v = \frac{\partial\Psi}{\partial x} \tag{15}$$

Plugging this into (12) yields

$$\frac{\partial p}{\partial x} = -\rho\frac{\partial^2\Psi}{\partial x\partial t} \tag{16}$$

We would also like the potential function Ψ to conform to the equation

$$\Delta p = -\rho_0\frac{\partial\Psi}{\partial t} \tag{17}$$

which indeed is consistent with (16) provided that $\Delta\rho\frac{\partial^2\Psi}{\partial t\partial x}$ and $\frac{\partial\Delta\rho}{\partial x}\frac{\partial\Psi}{\partial t}$ are small. As a final preparation to the horn equation, see now that from (14) and (17)

$$-\frac{1}{\rho}\frac{\partial\rho}{\partial t} = -\frac{1}{c^2}\frac{1}{\rho}\frac{\partial p}{\partial t} = \frac{1}{c^2}\frac{\partial^2\Psi}{\partial t^2} \tag{18}$$

From (10) it follows that

$$\begin{aligned}\rho\frac{\partial(Sv)}{\partial x} &= -S\frac{\partial\rho}{\partial t}\\ \Rightarrow \frac{1}{S}\frac{\partial}{\partial x}\Big(S\frac{\partial\Psi}{\partial x}\Big) &= -\frac{1}{\rho}\frac{\partial\rho}{\partial t}\\ \Rightarrow \frac{1}{S}\frac{\partial}{\partial x}\Big(S\frac{\partial\Psi}{\partial x}\Big) &= \frac{1}{c^2}\frac{\partial^2\Psi}{\partial t^2}\end{aligned} \tag{19}$$

where the final equation is commonly known as Webster's horn equation, after one of the first scientists to describe it . By expanding out the terms, one sees that this is a form of the 1D-wave equation where an extra term has been added to the right hand side:

$$\frac{\partial^2\Psi}{\partial t^2} = c^2\frac{\partial^2\Psi}{\partial x^2} + \frac{c^2}{S}\frac{\partial S}{\partial x}\frac{\partial\Psi}{\partial x} \tag{20}$$

For Webster's horn equation, and indeed for the analysis of any version of the 1D wave equation, it is often very useful to scale the equation in the spatial domain. We express the equation in terms of $x' = x/L$, such that $x' \in [0,1]$. For the horn equation, it will also be useful to scale the cross-sectional area with respect to a reference, here chosen as $S(0)$, by setting $S'(x) = S(x)/S(0)$. By employing the chain rule, it follows then that

$$\frac{c^2}{L^2}\Big(\frac{\partial^2\Psi}{\partial(x')^2} + \frac{1}{S'}\frac{\partial S'}{\partial x'}\frac{\partial\Psi}{\partial x'}\Big) = \frac{\partial^2\Psi}{\partial t^2} \tag{21}$$

Suppressing the primes and letting $\gamma = c/L$ we arrive finally at

$$\gamma^2\frac{\partial^2\Psi}{\partial x^2} = \frac{\partial^2\Psi}{\partial t^2} - \frac{\gamma^2}{S}\frac{\partial S}{\partial x}\frac{\partial\Psi}{\partial x} \tag{22}$$

By the chain rule, this is equivalent to

$$\gamma^2\frac{\partial}{\partial x}(S\frac{\partial\Psi}{\partial x}) = S\frac{\partial^2\Psi}{\partial t^2} \tag{23}$$

III. BOUNDARY CONDITIONS

There are several ways to set boundary conditions for the horn equation as described above. However, it makes sense to let the particle velocity equal some function u_{in}, of time, at the closed end of the pipe, $x = 0$. This may be written as $\Psi_x(0,t) = u_{in}(t)$. Most flue organ pipes have a conical foot through which air is blown. When this air meets an aperture, or "lip", on one side of the pipe, an instability arises, causing air pressure and particle velocity to fluctuate at the point of the lip. In our model, we envision spatial point $x = 0$ as the point at which the lip is located. The nature of instability at this point can be very intricate [4]. However, the precise form of u_{in} is of little importance in energy analysis and stability of difference schemes as long as u_{in} is bounded. We will assume a sinusoidal form for u_{in} in the computational implementation seen later. More choices are possible at the open end. For example, a zero-pressure condition would be $\Psi_t(1,t) = 0$. Other possible boundary conditions are those for which the pressure at the open end depends on the velocity at this point. It has been found experimentally (see [1], [5]) that for an unflanged pipe, a good boundary condition for the open end is

$$\Psi_x(1,t) = -\alpha_1\Psi_t(1,t) - \alpha_2\Psi(1,t) \qquad \text{where}$$
$$\alpha_1 = \frac{1}{4(0.61233)^2\gamma} \quad \alpha_2 = \frac{L * \sqrt{\pi}}{0.61233\gamma\sqrt{S(0)S(1)}} \tag{24}$$

Organ pipes are uflanged, and so we will use this relatively simple boundary condition in our model.

IV. A DIFFERENCE SCHEME FOR WEBSTER'S HORN EQUATION

Difference schemes for differential equations can be pathologically difficult to design. Under a choice of the wrong operators, the solutions approximated by such schemes are prone to grow exponentially. It is therefore important to analyse the numerical energy of a given system as implied by a scheme before applying it. As we shall see, a provably stable scheme for Webster's horn equation with boundary conditions as above is that as given by Bilbao ([1]):

$$(\mu_{xx}S_l)\delta_{tt}\Psi_l^n = \gamma^2\delta_{x+}((\mu_{x-}S_l)(\delta_{x-}\Psi_l^n)) \tag{25}$$

with boundary conditions

$$\delta_{x\cdot}\Psi_0^t = u_{in} \qquad \delta_{x\cdot}\Psi_N^t = -\alpha_1\delta_{t\cdot}\Psi_N^t - \alpha_2\mu_{t\cdot}\Psi_N^t \tag{26}$$

Note that $\mu_{xx}S_l = \mu_{x-}\mu_{x+}S_l = \frac{1}{4}(S_{l+1} + 2S_l + S_{l-1})$ is a second order accurate *approximation* to $S_l = S(lh)$. N denotes the spatial index of the points at which $x = 1$ (i.e. $Nk = 1$). Given choices for h and k, a notationally useful constant is $\lambda = \gamma k/h$. When expanded out, the scheme leads to the following expression for the approximated solution at the next time step:

$$\begin{aligned}\Psi_l^{n+1} =& \frac{\lambda^2(S_{l+1}+S_l)}{2\mu_{xx}S_l}\Psi_{l+1}^n + \frac{\lambda^2(S_l+S_{l-1})}{2\mu_{xx}S_l}\Psi_{l-1}^n \\ &+ 2(1-\lambda^2)\Psi_l^n - \Psi_l^{n-1}\end{aligned} \tag{27}$$

In addition, the boundary conditions (26) may be expanded to yield

$$\Psi_N^{n+1} = \frac{k}{(\alpha_1+\alpha_2 k)h}(\Psi_{N-1}^n - \Psi_{N+1}^n) + \frac{\alpha_1 - \alpha_2 k}{(\alpha_1+\alpha_2 k)}\Psi_N^{n-1} \tag{28}$$

$$\Psi_{-1}^t = \Psi_1^t - 2h(u_{in}(t)) \tag{29}$$

We notice that the values Ψ_{N+1}^t,Ψ_{-1}^t appear in these expressions. These are called virtual grid points, and given proper boundary conditions as above, their values are implied by values in the interior. For example, combining (27) and (28) yields

$$\begin{aligned}\Psi_N^{n+1} =& \frac{1}{\tau}\Big(4k\mu_{xx}S_l(2\Psi_N^n - \Psi_N^{n-1}) \\ &+ 4\lambda^2 k\mu_{xx}S_l(\Psi_{N-1}^n - \Psi_N^n) \\ &+ \lambda^2 h(S_{N+1}+S_N)(\alpha_1 - \alpha_2 k)\Psi_N^{n-1}\Big)\end{aligned} \tag{30}$$

where

$$\tau = \lambda^2 h(S_{N+1}+S_N)(\alpha_1+\alpha_2 k) + 2k\mu_{xx}S_l$$

Likewise, combining (27) and (29) produces

$$\begin{aligned}\Psi_0^{n+1} =& 2(1-\lambda^2)\Psi_0^n + 2\lambda^2\Psi_1^n \\ &- \frac{2\lambda^2 h(S_0+S_{-1})}{\mu_{xx}S_l}u_{in}(nk) - \Psi_0^{n-1}\end{aligned} \tag{31}$$

In (30) and (31), we see the appearance of S_{N+1} and S_{-1}, points that lie outside our spatial domain for the pipe. These values, however, must be set according to what we would like $\frac{dS}{dx}$ to be at the endpoints. Using a central difference approximation, we arrive at

$$S_{N+1} = 2hS'(1) + S_{N-1} \quad \text{and} \quad S_{-1} = S_1 - 2hS'(0) \tag{32}$$

Energy and Stability

Consider an inner product and the respective norm of two functions $f(x,t)$ and $g(x,t)$ as

$$<f,g>_D = \int_D fg dx \qquad ||f||_D = \sqrt{<f,f>_D} \tag{33}$$

where D is some domain. By taking the inner product with $\frac{\partial\Psi}{\partial t} = \Psi_t$ on both sides of Webster's equation, it is possible to arrive at an expression for the energy contained in the system ([1]). Specifically, integration by parts is used to get

$$<\Psi_t, S\Psi_{tt}>_{[0,1]} = \gamma^2 <\Psi_t, (S\Psi_x)_x>_{[0,1]}$$

$$\Rightarrow <\Psi_t, \Psi_{tt}>_{[0,1]} + \gamma^2 <\Psi_{xt}, S\Psi_x>_{[0,1]} = \gamma^2(S(1)\Psi_t(1,t)\Psi_x(1,t) - S(0)\Psi_t(0,t)\Psi_x(0,t)) \tag{34}$$

Considering the definition of the norm, the boundary conditions from (24), and the definitions of Δp and u, we see that

$$\frac{d}{dt}\Big(\frac{1}{2}||\sqrt{S}\Psi_t||^2_{[0,1]} + \frac{\gamma^2}{2}||\sqrt{S}\Psi_x||^2_{[0,1]}\Big) = -\gamma^2\Big(\alpha_1 S(1)(\Psi_t(1,t))^2 + \alpha_2\Psi_t(1,t)S(1)\Psi(1,t) + \Psi_t(0,t)u_{in}(t)\Big) \tag{35}$$

Seeing that $\Psi_t(1,t)\Psi(1,t) = (\Psi(1,t))^2/2$, we then get

$$\frac{d}{dt}\Big(\frac{1}{2}||\sqrt{S}\Psi_t||^2_{[0,1]} + \frac{\gamma^2}{2}||\sqrt{S}\Psi_x||^2_{[0,1]} + \frac{\gamma^2\alpha_2 S(1)}{2}(\Psi(1,t))^2\Big) \tag{36}$$

$$= -\gamma^2\alpha_1 S(1)(\Psi_t(1,t))^2 - \gamma^2\frac{\Delta p_{in}(t)}{\rho}u_{in}(t)$$

or, equivalently

$$\frac{d\mathcal{H}}{dt} = -\gamma^2\alpha_1 S(1)(\Psi_t(1,t))^2 - \gamma^2\frac{p_{in}(t)}{\rho}u_{in}(t) \tag{37}$$

as an expression for the change in energy over time, where $\mathcal{H}$ denotes the total energy of the system. The system's energy as implied by a numerical scheme can be analysed in a similar way. However, we must now define two inner products and respective norms. Consider two discrete functions f_l^n and g_l^n. The first inner product and norm will be defined as

$$<f,g>_E^{\epsilon_1,\epsilon_2} = (\sum_{l=1}^{N-1} h f_l^n g_l^n) + \frac{\epsilon_1}{2}hf_0g_0 + \frac{\epsilon_2}{2}hf_Ng_N$$
$$||f||_E^{\epsilon_1,\epsilon_2} = \sqrt{<f,g>_E^{\epsilon_1,\epsilon_2}} \tag{38}$$

for an ordered, discrete set of spatial points E with indices ranging from 0 to N. The second inner product and norm we define as

$$<f,g>_{\underline{E}} = (\sum_{l=0}^{N-1} h f_l^n g_l^n)$$
$$||f||_{\underline{E}} = \sqrt{<f,g>_{\underline{E}}} \tag{39}$$

Taking now the inner product of (25) with $\delta_{t\cdot}\Psi$ and letting E be the discrete set of spatial points in [0,1] as implied by h, we see first that for the right hand side

$$\gamma^2 <\delta_{t\cdot}\Psi_l^n, \delta_{x+}((\mu_{x-}S_l)(\delta_{x-}\Psi_l^n))>_E^{\epsilon_1,\epsilon_2}$$

$$= \frac{\gamma^2}{h}(<\delta_{t\cdot}\Psi_l^n, (\mu_{x+}S_l)(\delta_{x+}\Psi_l^n)>_E^{\epsilon_1,\epsilon_2} - <\delta_{t\cdot}\Psi_l^n, (\mu_{x-}S_l)(\delta_{x-}\Psi_l^n)>_E^{\epsilon_1,\epsilon_2})$$

$$= \gamma^2\sum_{l=0}^{N-1}(\mu_{x+}S_l)(\delta_{x+}\Psi_l^n)(\delta_{t\cdot}\Psi_l^n) - \gamma^2(\delta_{t\cdot}\Psi_l^n)(\mu_{x+}S_0)(\delta_{x+}\Psi_0^n)$$
$$+ \gamma^2\frac{\epsilon_2}{2}(\delta_{t\cdot}\Psi_N^n)(\mu_{x+}S_N)(\delta_{x+}\Psi_N^n) + \gamma^2\frac{\epsilon_1}{2}(\delta_{t\cdot}\Psi_0^n)(\mu_{x+}S_0)(\delta_{x+}\Psi_0^n)$$

$$-\gamma^2\sum_{l=1}^{N}(\mu_{x-}S_l)(\delta_{x-}\Psi_l^n)(\delta_{t\cdot}\Psi_l^n) + \gamma^2(\delta_{t\cdot}\Psi_l^n)(\mu_{x-}S_N)(\delta_{x-}\Psi_N^n)$$
$$-\gamma^2\frac{\epsilon_2}{2}(\delta_{t\cdot}\Psi_N^n)(\mu_{x-}S_N)(\delta_{x-}\Psi_N^n) - \gamma^2\frac{\epsilon_1}{2}(\delta_{t\cdot}\Psi_0^n)(\mu_{x-}S_0)(\delta_{x-}\Psi_0^n)$$

$$= -\gamma^2\sum_{l=0}^{N-1}(\mu_{x+}S_l)(\delta_{x+}\Psi_l^n)(\delta_{t\cdot}\Psi_{l+1}^n - \delta_{t\cdot}\Psi_l^n) \tag{40}$$
$$+\gamma^2(\delta_{t\cdot}\Psi_l^n)\Big(\frac{\epsilon_2}{2}(\mu_{x+}S_N)(\delta_{x+}\Psi_N^n) + \Big(1-\frac{\epsilon_2}{2}\Big)(\mu_{x-}S_N)(\delta_{x-}\Psi_N^n)\Big)$$
$$-\gamma^2(\delta_{t\cdot}\Psi_l^n)\Big(\frac{\epsilon_1}{2}(\mu_{x-}S_0)(\delta_{x-}\Psi_0^n) + \Big(1-\frac{\epsilon_1}{2}\Big)(\mu_{x+}S_0)(\delta_{x+}\Psi_0^n)\Big)$$

See now that

$$-\gamma^2\sum_{l=0}^{N-1}(\mu_{x+}S_l)(\delta_{x+}\Psi_l^n)(\delta_{t\cdot}\Psi_{l+1}^n - \delta_{t\cdot}\Psi_l^n)$$
$$= \frac{-\gamma^2}{2k}\sum_{l=0}^{N-1}(\mu_{x+}S_l)(\delta_{x+}\Psi_l^n)(\delta_{x+}\Psi_l^{n+1} - \delta_{x+}\Psi_l^{n-1})$$
$$= -\delta_{t+}\Big(\frac{\gamma^2}{2}\sum_{l=0}^{N-1}h(\mu_{x+}S_l)(\delta_{x+}\Psi_l^n)(e_{t-}\delta_{x+}\Psi_l^n)\Big)$$
$$= -\delta_{t+}\Big(\frac{\gamma^2}{2}<((\mu_{x+}S_l)(\delta_{x+}\Psi_l^n), e_{t-}\delta_{x+}\Psi_l^n>_{\underline{E}}\Big) \tag{41}$$

For the left hand side, it follows from the identity in (4), that

$$<(\delta_{t\cdot}\Psi_l^n, (\mu_{xx}S_l)(\delta_{tt}\Psi_l^n)>_E^{\epsilon_1,\epsilon_2}$$
$$= \sum_{l=0}^{N-1}h(\mu_{xx}S_l)(\delta_{t\cdot}\Psi_l^n)(\delta_{tt}\Psi_l^n)$$
$$+\frac{\epsilon_1}{2}h(\mu_{xx}S_0)(\delta_{t\cdot}\Psi_0^n)(\delta_{tt}\Psi_0^n) + \frac{\epsilon_2}{2}h(\mu_{xx}S_N)(\delta_{t\cdot}\Psi_N^n)(\delta_{tt}\Psi_N^n)$$
$$= \frac{1}{2}\sum_{l=0}^{N-1}h(\mu_{xx}S_l)(\delta_{t+}(\delta_{t-}\Psi_l^n)^2)$$
$$+\frac{\epsilon_1}{4}h(\mu_{xx}S_0)(\delta_{t+}(\delta_{t-}\Psi_0^n)^2) + \frac{\epsilon_2}{4}h(\mu_{xx}S_N)(\delta_{t+}(\delta_{t-}\Psi_N^n)^2)$$
$$= \delta_{t+}\Big(\frac{1}{2}\Big(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_E^{\epsilon_1,\epsilon_2}\Big)^2\Big) \tag{42}$$

Now putting (40), (41), and (42) together (and paying attention to what was on the left- and right hand sides of (25)) we see that

$$\begin{aligned}&\delta_{t+}\Big(\frac{1}{2}(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_{\underline{E}}^{\epsilon_1,\epsilon_2})^2\\&\qquad+\frac{\gamma^2}{2}<((\mu_{x+}S_l)(\delta_{x+}\Psi_l^n),e_{t-}\delta_{x+}\Psi_l^n>_{\underline{E}}\Big)\\&=\gamma^2(\delta_{t\cdot}\Psi_N^n)\Big(\frac{\epsilon_2}{2}(\mu_{x+}S_N)(\delta_{x+}\Psi_N^n)\\&\qquad+\big(1-\frac{\epsilon_2}{2}\big)(\mu_{x-}S_N)(\delta_{x-}\Psi_N^n)\Big)\\&\quad-\gamma^2(\delta_{t\cdot}\Psi_0^n)\Big(\frac{\epsilon_1}{2}(\mu_{x-}S_0)(\delta_{x-}\Psi_0^n)\\&\qquad+\big(1-\frac{\epsilon_1}{2}\big)(\mu_{x+}S_0)(\delta_{x+}\Psi_0^n)\Big)\end{aligned}$$

We now let $\epsilon_1=\mu_{x+}S_0/\mu_{xx}S_0$, and $\epsilon_2=\mu_{x-}S_N/\mu_{xx}S_N$. Considering the boundary conditions (26), we see that

$$\begin{aligned}&\delta_{t+}\Big(\frac{1}{2}(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_{\underline{E}}^{\epsilon_1,\epsilon_2})^2\\&\qquad+\frac{\gamma^2}{2}<((\mu_{x+}S_l)(\delta_{x+}\Psi_l^n),e_{t-}\delta_{x+}\Psi_l^n>_{\underline{E}}\Big)\\&=-\gamma^2\frac{(\mu_{x-}S_N)(\mu_{x+}S_N)}{\mu_{xx}S_N}\big(\alpha_1(\delta_{t\cdot}\Psi_N^n)^2+\alpha_2(\delta_{t\cdot}\Psi_N^n)(\mu_{t\cdot}\Psi_N^n)\big)\\&\quad-\gamma^2\frac{(\mu_{x-}S_0)(\mu_{x+}S_0)}{\mu_{xx}S_0}(p_{in}(t))(u_{in}(t))\end{aligned}\tag{43}$$

It may be shown in a similar manner to (4) and (5), that

$$\alpha_2(\delta_{t\cdot}\Psi_N^n)(\mu_{t\cdot}\Psi_N^n)=\delta_{t+}(\frac{\alpha_2}{2}\mu_{x-}(\Psi_N^n)^2)\tag{44}$$

Reorganising (43), an expression for the total anergy and its first order difference approximation with respect to time appears:

$$\begin{aligned}\delta_{t+}\mathfrak{H}=&-\gamma^2\frac{(\mu_{x-}S_N)(\mu_{x+}S_N)}{\mu_{xx}S_N}\big(\alpha_1(\delta_{t\cdot}\Psi_N^n)^2\big)\\&-\gamma^2\frac{(\mu_{x-}S_0)(\mu_{x+}S_0)}{\mu_{xx}S_0}(p_{in}(t))(u_{in}(t))\end{aligned}\tag{45}$$

where

$$\begin{aligned}\mathfrak{H}=&\frac{1}{2}(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_{\underline{E}}^{\epsilon_1,\epsilon_2})^2\\&+\frac{\gamma^2}{2}<((\mu_{x+}S_l)(\delta_{x+}\Psi_l^n),e_{t-}\delta_{x+}\Psi_l^n>_{\underline{E}}+\frac{\alpha_2}{2}\mu_{x-}(\Psi_N^n)^2\end{aligned}\tag{46}$$

From (45), we see that for $\delta_{x\cdot}\Psi_0^n=u_{in}(t)=0$ the energy is non-increasing. This is to be expected given the lossy boundary conditions (26). Furthermore, given some input function $\Psi_{in}(t)$ the energy change is bounded above. It is a necessary condition for stability that the numerical energy is also non-negative ([1]). In this regard, we must derive one last inequality. By applying identity (5) to $\delta_{x+}\Psi_l^n$ we see that

$$\begin{aligned}&\frac{\gamma^2}{2}<((\mu_{x+}S_l)(\delta_{x+}\Psi_l^n),e_{t-}\delta_{x+}\Psi_l^n>_{\underline{E}}\\&=\frac{\gamma^2}{2}\sum_{l=0}^{N-1}h(\mu_{x+}S_l)\big((\delta_{x+}\mu_{t-}\Psi_l^n)^2-\frac{k^2}{4}(\delta_{x+}\delta_{t-}\Psi_l^n)^2\big)\\&\geq-\frac{k^2\gamma^2}{8}||\sqrt{\mu_{x+}S_l}\delta_{x+}\delta_{t-}\Psi_l^n||_{\underline{E}}^2\end{aligned}\tag{47}$$

Note now that

$$\begin{aligned}&||\sqrt{\mu_{x+}S_l}\delta_{x+}\delta_{t-}\Psi_l^n||_{\underline{E}}^2\\&=\sum_{l=0}^{N-1}h(\mu_{x+}S_l)\frac{1}{h^2}(\delta_{t-}\Psi_{l+1}^n-\delta_{t-}\Psi_l^n)^2\\&\leq\frac{2}{h}\sum_{l=0}^{N-1}(\mu_{x+}S_l)((\delta_{t-}\Psi_{l+1}^n)^2+(\delta_{t-}\Psi_l^n)^2)\end{aligned}\tag{48}$$

On the other hand, see that

$$\begin{aligned}&\Big(\frac{2}{h}||\sqrt{\mu_{x-}\mu_{x+}S_l}\delta_{t-}\Psi_l^n||_{\underline{E}}^{\epsilon_1,\epsilon_2}\Big)^2\\&=\frac{4}{h^2}\sum_{l=1}^{N-1}\frac{h}{2}(\mu_{x+}S_l+\mu_{x+}S_{l-1})(\delta_{t-}\Psi_l^n)^2\\&\quad+\frac{4}{h^2}\frac{\epsilon_1}{2}h(\mu_{xx}S_0)(\delta_{t-}\Psi_0^n)^2+\frac{4}{h^2}\frac{\epsilon_2}{2}h(\mu_{xx}S_N)(\delta_{t-}\Psi_N^n)^2\\&=\frac{2}{h}\sum_{l=1}^{N-1}(\mu_{x+}S_l)(\delta_{t-}\Psi_l^n)^2+\frac{2}{h}\sum_{l=0}^{N-2}(\mu_{x+}S_l)(\delta_{t-}\Psi_{l+1}^n)^2\\&\quad+\frac{2}{h}\epsilon_1(\mu_{xx}S_0)(\delta_{t-}\Psi_0^n)^2+\frac{2}{h}\epsilon_2(\mu_{xx}S_N)(\delta_{t-}\Psi_N^n)^2\end{aligned}\tag{49}$$

From (48), it follows then that

$$\begin{aligned}&\Big(\frac{2}{h}||\sqrt{\mu_{xx}S_l}\delta_{t-}\Psi_l^n||_{\underline{E}}^{\epsilon_1,\epsilon_2}\Big)^2-||\sqrt{\mu_{x+}S_l}\delta_{x+}\delta_{t-}\Psi_l^n||_{\underline{E}}^2\\&=\frac{2}{h}\epsilon_1(\mu_{xx}S_0)(\delta_{t-}\Psi_0^n)^2+\frac{2}{h}\epsilon_2(\mu_{xx}S_N)(\delta_{t-}\Psi_N^n)^2\\&\quad-\frac{2}{h}(\mu_{x+}S_0)(\delta_{t-}\Psi_0^n)^2-\frac{2}{h}(\mu_{x-}S_N)(\delta_{t-}\Psi_N^n)^2\\&=\frac{2}{h}(\mu_{x+}S_0)(\delta_{t-}\Psi_0^n)^2+\frac{2}{h}(\mu_{x-}S_N)(\delta_{t-}\Psi_N^n)^2\\&\quad-\frac{2}{h}(\mu_{x+}S_0)(\delta_{t-}\Psi_0^n)^2-\frac{2}{h}(\mu_{x+}S_N)(\delta_{t-}\Psi_N^n)^2\\&=0\end{aligned}\tag{50}$$

With this last equality at hand, it follows immediately from (47) that

$$\begin{aligned}&\frac{\gamma^2}{2}<((\mu_{x+}S_l)(\delta_{x+}\Psi_l^n),e_{t-}\delta_{x+}\Psi_l^n>_{\underline{E}}\\&\qquad\geq-\frac{\lambda^2}{2}\Big(||\sqrt{\mu_{xx}S_l}\delta_{t-}\Psi_l^n||_{\underline{E}}^{\epsilon_1,\epsilon_2}\Big)^2\end{aligned}$$

Finally, it follows from (46) that

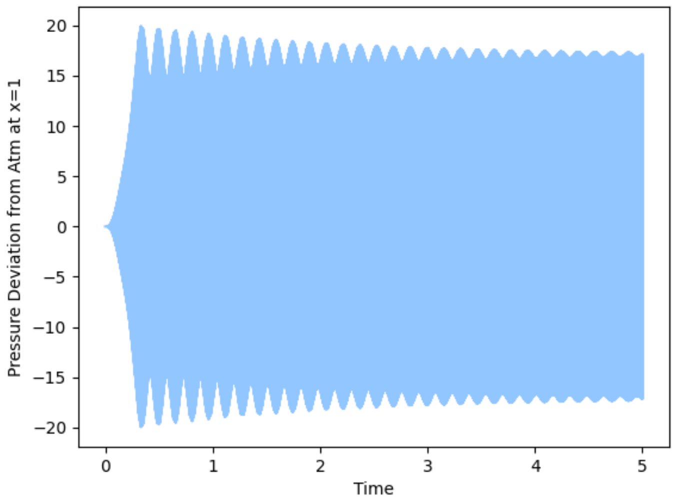
$$\begin{aligned}\mathfrak{H} \geq & \frac{1}{2}(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_E^{\epsilon_1,\epsilon_2})^2 - \frac{\lambda^2}{2}(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_E^{\epsilon_1,\epsilon_2})^2 \\ = & \frac{1-\lambda^2}{2}(||\sqrt{\mu_{xx}S_l}(\delta_{t-}\Psi_l^n)||_E^{\epsilon_1,\epsilon_2})^2\end{aligned}$$

Hence the numerical energy is non-negative for $\lambda \leq 1$. Below we show Python code that implements such a scheme for a pipe of length 3.7332 meters and cross-sectional area in square meters given by $S(x) = \pi(0.0549 + 0.05x)^2$ for scaled $x \in [0, 1]$. The samplerate is set to the common value of 44100 (and so $k = 1/44100$) in order to encapsulate most of the frequencies in the humanly audible spectrum ([1]). We set $h = \gamma * k$ (and so $\lambda = 1$) in order to maximise the accuracy of the spatial operators. There are however, deeper reasons to make λ as large as possible within the bound, as described by Bilbao ([1]), which we will not elaborate on here. A speed of sound of $345m/s$ and air-density of $1.204kg/m^3$ are assumed. Note that we let the particle velocity u_{in} be a sine function with frequency 200; high enough for the human spectrum. Furthermore, we let the amplitude of u_{in} increase parabolically up to time $t = 0.3$ in order to mimic the flow of air into the pipe gradually increasing and then reaching some steady rate. The pressure difference from atmospheric pressure, Δp, is measured at the open end of the pipe, $x = 1$. A playable .wav file is produced alongside a plot of the pressure at $x = 1$ against time. In the code below, the shape of the pipe, its length, and input function u_{in} may all be changed at the users behest to emulate the sound of different physical pipe-set-ups. It would also be possible to measure the effect of a particular input function u_{in} (not necessarily sinusoidal) for a given pipe by inputting a range of frequencies and measuring the amplitude of the output. That is outside the scope of this project, however. Note that, in accordance with (45) and the input function u_{in}, the energy will grow at a rate which is bounded above, and that the energy can never be negative.

V. CODE

```python
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.animation import FuncAnimation
import math
from scipy.io.wavfile import read as wavread
from scipy.io.wavfile import write as wavwrite
plt.style.use('seaborn-pastel')

# Set pipe parameters
length = 3.7332
sound_speed = 345
air_density = 1.204

# define gamma parameter as per equation
gamma = sound_speed / length

# define samplerate and length in seconds of output
samplerate = 44100
seconds = 5
timesteps = int(samplerate * seconds)

# array of time points at which I would like to approximate solution
time_points = np.linspace(0, seconds, timesteps, endpoint=False)

# Time step k
k = (1 / samplerate)

# spatial step satisfying CFL-condition (must not be less than gamma*k)
# Raise exception if violated
h = gamma * k

if (h < gamma * k):
    raise Exception("CFL-condition violated")

# Make a lambda variable
lambda_ = gamma * k / h

# Create array of spatial points at which I would like to approximate solution.
# The number of points must be an integer
spatial_count = int((1 / h) + 1)
spatial_points = np.linspace(0, 1, spatial_count)

# Define index number for last spatial point
N = spatial_count - 1

# Define Cross-sectional area S(x) in terms of radius
left_rad = 0.0549

# find reference cross-section S_0unscaled
S_0unscaled = ((left_rad)**2) * np.pi

# find area of cross section at different spatial points
# and scale by area at closed end
S = ((left_rad + spatial_points * 0.05)**2) * np.pi / S_0unscaled

# set virtual boundary points
S_min1 = S[1] - (0.05 * 4 * h * left_rad * np.pi / S_0unscaled)
S_Nplus1 = (0.05 * 4 * h * (left_rad + 0.05) *
            np.pi / S_0unscaled) + spatial_points[N - 1]


# Calculate mu_(xx)S
mu_xxS = np.zeros(spatial_count)
mu_xxS[0] = (S[1] + 2 * S[0] + S_min1) / 4
mu_xxS[N] = (S_Nplus1 + 2 * S[N] + S[N - 1]) / 4
for i in range(1, N):
    mu_xxS[i] = (S[i + 1] + 2 * S[i] + S[i - 1]) / 4

# Define parameters alpha_1, alpha_2
alpha_1 = (((2 * 0.6133)**2) * gamma)**(-1)
alpha_2 = length * np.sqrt(np.pi) / (0.6133 * np.sqrt(S_0unscaled * S[1]))

# define an input frequency
input_freq = 523.25


def u_in(t):
    # Input function with instability resulting from beating of reed
    # or lip on flue pipe. Assume particle velocity at x=0 is initially 0
    # and increasing up until time point t=0.3s
    if t <= 0.3:
        return (1 / (0.09)) * (t**2) * (np.sin(2 * np.pi * (input_freq) * t))
    else:
        return (np.sin(2 * np.pi * (input_freq) * t))


# Define matrix according to (27), (30), (31)
matrix = 2 * (1 - (lambda_**2)) * np.identity(spatial_count)
matrix[0, 1] = 2 * (lambda_**2)
for i in range(1, N):
    matrix[i, i - 1] = (lambda_**2) * (S[i] + S[i - 1]) / (2 * mu_xxS[i])
    matrix[i, i + 1] = (lambda_**2) * (S[i + 1] + S[i]) / (2 * mu_xxS[i])

# Define last row in matrix
tau = (lambda_**2) * h * (S_Nplus1 + S[N]) * \
    (alpha_1 + (alpha_2 * k)) + 2 * k * mu_xxS[N]
matrix[-1, -1] = 4 * k * (1 - (lambda_**2)) * (mu_xxS[N]) * (1 / tau)
matrix[-1, -2] = 4 * k * (lambda_**2) * mu_xxS[N] * (1 / tau)

# Define factor on u_in
u_in_factor = (lambda_**2) * h * (S_0unscaled + S_min1) / (mu_xxS[0])

# Coefficient on extra term for Psi_N_nmin1
Psi_N_nmin1_coeff = (lambda_**2) * h * (S_Nplus1 + S[N]) * alpha_1 * (1 / tau)

# Set initial conditions for the parameter Psi (time points 0 and 1).
Psi_x_nmin1 = np.zeros(spatial_count)
Psi_x_n = np.copy(Psi_x_nmin1)

# Initiate output at x=1. Output is measured in pressure deviation
# from some reference, p = -air-density * delta_
    (t dot)Psi
output = np.zeros(timesteps)

# Function that iteratively fills output array


def update(i):
    global Psi_x_nmin1, Psi_x_n, output, counter

    # calculate next time step as given by
    formula
    Psi_x_next = matrix @ (Psi_x_n.T) -
    Psi_x_nmin1
    Psi_x_next[0] -= u_in_factor * u_in(
    time_points[i])
    Psi_x_next[-1] += Psi_N_nmin1_coeff *
    Psi_x_nmin1[N]

    # calculate the pressure deviation from atm
    pressure_deviation = -air_density * (
    Psi_x_next - Psi_x_nmin1) / (2 * k)

    # append to output
    output[i] = pressure_deviation[N]

    # move indeces
    Psi_x_nmin1 = np.copy(Psi_x_n)
    Psi_x_n = np.copy(Psi_x_next)

    return pressure_deviation


# Run update function from time-point 1 to
    seconds*samplerate
for i in range(1, timesteps):
    update(i)

# Plot pressure profile at x=1 against time
plt.plot(time_points, output)
plt.xlabel("Time")
plt.ylabel("Pressure Deviation from Atm at x=1")
plt.show()

# Audio output at x=1:
# The final amplitude should span the signed 16-
    bit integers range [-2**15,2**15)
# iinfo returns the range for the int16 type,
    then max resolves to -> 2**15
print(np.max(output))
signal1 = output
max_value = np.max(abs(signal1))
amplitude = np.iinfo(np.int16).max
data = amplitude * signal1 / max_value
wavwrite("pipesim.wav", samplerate, data.astype(
    np.int16))

# display sample rate and samples
rate, data = wavread("pipesim.wav")
print('spatial_count:', spatial_count)
print('rate:', rate, 'Hz')
print('data is a:', type(data))
print('points are a:', type(data[0]))
print('data shape is:', data.shape)

# Play sound from wav file using relevant
    library
from playsound import playsound
file = '/Users/sondrew/Documents/jobb/Internship
     2022/Summer Project/CODE/pipesim.wav'
playsound(file, block=True)
```

Plot produced:



BIBLIOGRAPHY

[1] Bilbao, S., 2009. Numerical Sound Synthesis. 1st ed. Chichester: John Wiley & Sons, Ltd.

[2] Mol, H., 1970. IV. On Webster's Horn Equation. In (Ed.), Fundamentals of Phonetics, II: Acoustical Models, Generating the Formants of the Vowel Phonemes (pp. 37-44). Berlin, Boston: De Gruyter Mouton. https://doi.org/10.1515/9783110811070-005

[3] OpenStax. "17.3: Speed of Sound." Physics LibreTexts. Libretexts, March 7, 2022.

[4] J. Angster, P. Rucz, and A. M. Miklós, Acoustics Today 13, 10 (2017).

[5] M. Atig, J.-P. Dalmont, and J. Gilbert. Termination impedance of open-ended cylindrical tubes at high sound pressure level. Comptes Rendus Mécanique, 332:299–304, 2004.