

Simulation of Room Acoustics Using Transmission Line Matrix

Theory

The concept of the *Transmission Line Matrix* (TLM) method is to partition the simulated space into a grid of discrete points, called nodes. Each node is connected to the adjacent nodes. The TLM method was first used to model electromagnetic wave propagation, and in that context the connections are considered to be some form of conductors. In the case of acoustics they can be seen as small pipes in which pressure waves travel. The nodes can be seen as pipe junctions. The TLM method is not restricted to a specific dimension. In the two dimension case we are simulating, every node is connected to four other nodes.

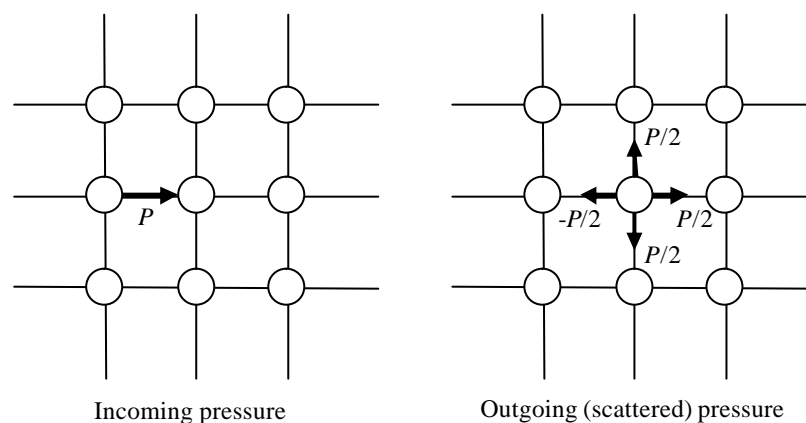


Fig. 1. The scattering of the pressure.

At any given time, it is possible to calculate how the pressure coming in to a node from a neighbouring node is scattered to all connected nodes. In the two dimensional case, the pressure P is scattered according to the figure above: $-\frac{1}{2}P$ is returned to the node it was coming from, the three other nodes receive $\frac{1}{2}P$ each.

The reason $-\frac{1}{2}P$ is reflected back is the same phenomenon as wave reflections at the end of an open pipe. As a pressure wave enters a junction, it is transferred from one pipe to three pipes, thus the cross section area is increased by a factor of three.



The quotient of the outgoing pressure P_r and the incoming pressure P_i is:

$$\frac{P_r}{P_i} = \sqrt{\frac{(S_1 - S_2)^2}{(S_1 + S_2)^2}},$$

where S_1 and S_2 are the source and destination cross section areas (Kinsler et al. *p. 286*).

Since we know that $\frac{S_2}{S_1} = 3$, this gives us:

$$\frac{P_r}{P_i} = \sqrt{\frac{(S_1 - 3S_1)^2}{(S_1 + 3S_1)^2}} = \frac{-2S_1}{4S_1} = -\frac{1}{2} \Rightarrow P_r = -\frac{1}{2}P_i.$$

The reason $\frac{1}{2}P$ is scattered to the other three nodes can be found in energy conservation.

The energy density of a harmonic pressure wave can be described as $\langle w \rangle = \frac{P^2}{2c^2 \rho}$,

where P is the amplitude of the pressure wave, c is the phase velocity and ρ is the density (Nordling et al. *p. 229*). Since the energy density and the denominator is constant, P^2 must also be constant. This means that if P_1 is the resulting pressure in each of the other three pipes, then

$$P^2 = \left(\frac{1}{2}P\right)^2 + 3P_1^2 \Rightarrow P_1^2 = \frac{P^2 - \left(\frac{1}{2}P\right)^2}{3} = \frac{P^2}{4} \Rightarrow P_1 = \frac{1}{2}P.$$

It is possible to add boundary conditions to nodes. A node can be reflective, and in that case the scattering described above is not used. Instead of spreading the pressure to all four connections, the pressure is returned to the node it came from. By adding an area of reflective nodes it is possible to simulate a reflective wall. Absorbing nodes, which does not return any pressure at all can simulate absorbing walls. By using boundary conditions such as these it is possible to create closed areas to observe echoes or view diffraction effects of a gap in a wall.



Implementation

The program is written in C++. We use *Simple Direct Media Layer* to handle the graphics and mouse and keyboard input. The implementation of the method is pretty straightforward.

We store the incoming (input) and the outgoing (output) pressure in the four directions for every node. We also need to know if there are boundary conditions in that node. The input and the output can be seen as two vectors where each element corresponds to a specific connection (up, right, left, down). The boundary conditions and the scattering are stored in 4x4 matrices, and the output can then be calculated by multiplying the input with one of the matrices.

We use three different matrices:

Reflection:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Absorption:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Scatter:

$$\frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

The Reflection and the Absorption Matrix are used where there's a reflecting or an absorbing wall. Multiplying the input vector with the Reflection Matrix (which is the identity matrix) will result in the input vector again, i.e. the pressure is reflected back. Multiplying with the Absorption Matrix will result in a zero vector, i.e. no pressure leaves this node – everything is absorbed. The Scatter Matrix is used where there are no walls, so this is the one used most frequent. It performs the scattering described earlier (see fig.1). With this implementation it is very easy to add other properties (e.g. half absorbing nodes). We would only have to create a new matrix to perform the desired transformation.

Although we used C++, we did not define and use node, vector or matrix classes in our implementation. Instead we save all inputs and outputs in two different one dimensional arrays. A third array stores a pointer to a matrix to be used for every node. The matrices are saved in three different arrays of length 16.

The reason we used this very simple data structure was because we considered taking advantage of 3DNow and/or SSE/SSE2 to perform the actual multiplication with the matrix. Since this matrix operation requires 16 multiplications and 12 additions per node and is performed for every node it constitutes the far majority of the computations. By utilizing these processor functions (which can perform 4 floating point multiplications in one instruction) we believe it would be possible to speed up the program significantly. Unfortunately there was no time left to implement this, but it would be interesting to add



Anders Stenberg
Mattias Stridsman
2003-03-05

support for this because the performance could improve significantly. The disadvantage with this simple data structure was that it made it a bit more cumbersome to write the code since all nodes' inputs are stored in sequence.

When the pressure is visualized, every node is represented as a pixel where a blue colour represents low pressure and a green colour represents high pressure.

Limitations

The TLM method can quickly consume large amounts of processing power, since it requires a large amount of nodes in order to be able to view interesting phenomena. We used a default grid of 300x300 nodes, and if the grid is much larger than that, the process will be very slow on an average machine. Still this method of simulation is faster than many other approaches.

Another limitation we discovered was that a “wall” of absorbing nodes creates faint reflections. We cannot find any faults in our implementation, and we have noted that others mention this, and there are supposedly solutions to this problem, but that is nothing we have been looking into.



Anders Stenberg
Mattias Stridsman
2003-03-05

References

- Wilde, Andreas & Eccardt, Peter-Christian & O'Connor, William (2001).
"Modelling acoustic transducer surface waves by Transmission Line Matrix method" in
19th CAD-FEM User's Meeting 2001, Berlin
- Kinsler, Lawrence E. & Frey, Austin R. & Coppens Alan B. & Sanders, James V. (2000)
"Fundamentals of Acoustics", John Wiley & Sons Inc, Forth Edition
- Nordling, Carl & Österman, Jonny (1996). "Physics Handbook for Science and
Engineering", Studentlitteratur, Fifth Edition