Toward a Real-Time Waveguide Mesh Implementation

Tamara Smyth,[1†] *Jennifer Hsu,*[2] *Ryan Done*[1]

[1]Department of Music, UC San Diego, La Jolla, California

[†]trsmyth@ucsd.edu

ABSTRACT

This work presents an analytic solution to a 2-D waveguide mesh, made square with input and output at the center to produce a symmetry that reduces redundancy and increases computational savings. The result is a frequency-domain representation of the mesh, a parametric transfer function describing the ratio of output to input, symbolically computed to retain parameters. In this form, the mesh, known for being computationally prohibitive, can be further made to perform in real-time, with (symbolic) parameters allowing for interaction by the user. This work is motivated by a desire to explore real-time interactive/parametric percussion synthesis.

1. INTRODUCTION

It is well known that one-dimensional (1-D) wave propagation can be *efficiently* modeled using a 1-D digital waveguide, a bi-directional delay line that models acoustic propagation delay associated with waves travelling to the right and left along one-dimensional systems such as strings and cylindrical (and conical) bores found in musical instruments. This efficiency is significant for virtual musical instrument design because it allows these systems to be modeled *parametrically* for use in *real time*, allowing the user to change parameter values while hearing the (perceptually) immediate response.

The efficiency of the digital waveguide is lost when brought to higher spatial dimensions. Though 2-D and 3-D waveguides may be used to model wave propagation on membranes, plates, and cavities [1], typical (time-domain) implementations are computationally prohibitive for real-time performance. A 2001 paper [2] reports running a 6×6 2-D mesh in real time on a Pentium III 500 MHz (and 10×10 on a Athlon 1.3 GHz), with more current tests (on an Intel core i5 2.2 GHz) supporting (but struggling at) up to 45×45. Many musically interesting sounds are produced with meshes far above 100×100 and [2] point out that further research is clearly required to optimise the mesh algorithm if it is to be more widely used. As a result, their use in virtual musical instruments has been largely limited to *offline* applications: analysis, non-interactive sound synthesis, or modeling of components not expected to change in real time (e.g. resonators such as soundboards or instrument bodies). The lack of real-time low-latency interaction has made musical exploration of these systems more difficult than their 1-D counterpart. Here, an analytic solution of a simplified (made to have computation-saving symmetry) 2-D waveguide mesh (so called because of the interlaced nature of its time-domain signal flow) is presented, providing an alternate, yet equal, symbolic frequency-domain representation (a transfer function that retains boundary loss parameters), that is intended for parametric percussion synthesis.

2. THE WAVEGUIDE MESH

The 2-D mesh can be viewed as a matrix of junctions with each junction $J_{i,j}$ having 4 (north, south, east, and west) input-output (I/O) ports, as indicated by the '$+$' and '$-$' superscripts, respectively (see Figure 1). In this work, the mesh is constrained to be square ($M \times M$).

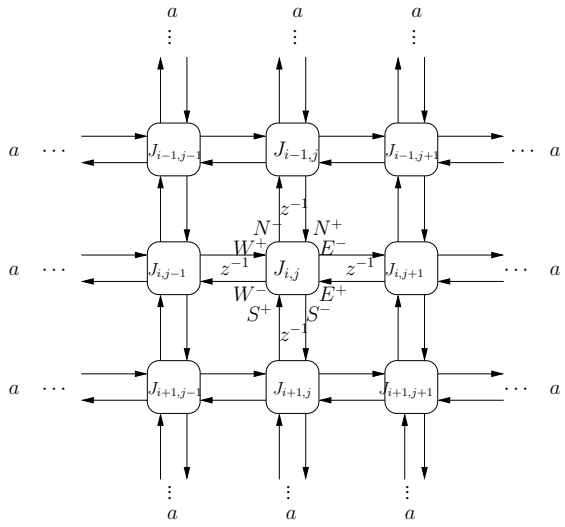

Figure 1. Mesh junctions showing input/outputs on the north N, south S, west W and east E ports and a delay of one sample (z^{-1}) between port inputs and neighbouring port outputs.

The port inputs on junction $J_{m,n}$ may be expressed as a function of neighbouring port outputs with a delay of one sample, with the exception of a mesh boundary where the round-trip from output to input on a port introduces a 2-sample delay and a multiply with boundary scalar reflection loss a:

$$
\begin{aligned}
N^+_{i,j}(n) &= \begin{cases} aN^-_{i,j}(n-2), & \text{if } i = 1 \text{ (north bndry);} \\ S^-_{i-1,j}(n-1) & \text{otherwise.} \end{cases} \\
S^+_{i,j}(n) &= \begin{cases} aS^-_{i,j}(n-2), & \text{if } i = N \text{ (south bndry);} \\ N^-_{i+1,j}(n-1) & \text{otherwise.} \end{cases} \\
E^+_{i,j}(n) &= \begin{cases} aE^-_{i,j}(n-2), & \text{if } j = M \text{ (east bndry);} \\ W^-_{i,j+1}(n-1), & \text{otherwise.} \end{cases} \\
W^+_{i,j}(n) &= \begin{cases} aW^-_{i,j}(n-2), & \text{for } j = 1 \text{ (west bndry);} \\ E^-_{i,j-1}(n-1), & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}
$$

The output on each port at time sample n is given by the junction's total velocity (the wave variable used here),

$$vJ_{i,j}(n) = \frac{N^+_{i,j}(n) + S^+_{i,j}(n) + E^+_{i,j}(n) + W^+_{i,j}(n)}{2}, \quad (2)$$

minus the port input which, when applying (2), yields:

$$\begin{aligned}
N^-_{i,j}(n) &= vJ_{i,j}(n) - N^+_{i,j}(n) \\
&= \frac{-N^+_{i,j}(n) + S^+_{i,j}(n) + E^+_{i,j}(n) + W^+_{i,j}(n)}{2} \\
S^-_{i,j}(n) &= vJ_{i,j}(n) - S^+_{i,j}(n) \\
&= \frac{N^+_{i,j}(n) - S^+_{i,j}(n) + E^+_{i,j}(n) + W^+_{i,j}(n)}{2} \\
E^-_{i,j}(n) &= vJ_{i,j}(n) - E^+_{i,j}(n) \\
&= \frac{N^+_{i,j}(n) + S^+_{i,j}(n) - E^+_{i,j}(n) + W^+_{i,j}(n)}{2} \\
W^-_{i,j}(n) &= vJ_{i,j}(n) - W^+_{i,j}(n) \\
&= \frac{N^+_{i,j}(n) + S^+_{i,j}(n) + E^+_{i,j}(n) - W^+_{i,j}(n)}{2}.
\end{aligned} \quad (3)$$

Substituting (1) into (3) yields final expressions (given here in matrix form) for port outputs (column vector[1] $\mathbf{J}_{i,j}$) as a function of neighbouring port outputs (column vector $\mathbf{b}_{i,j}$), with the now dispensable '$-$' superscript henceforth omitted:

$$\mathbf{J}_{\mathbf{n}i,j} = \mathbf{A}\mathbf{b}_{\mathbf{n}i,j} \quad (4)$$

where

$$\mathbf{J}_{\mathbf{n}i,j} = (N_{i,j}(n), S_{i,j}(n), E_{i,j}(n), W_{i,j}(n)), \quad (5)$$

$$\mathbf{A} = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}, \quad (6)$$

and where the neighbouring port dependencies is given by

$$\mathbf{b}_{\mathbf{n}i,j} = \begin{bmatrix} S_{i-1,j}(n-1) \\ N_{i+1,j}(n-1) \\ W_{i,j+1}(n-1) \\ E_{i,j-1}(n-1)) \end{bmatrix} \quad (7)$$

with exceptions as per boundary conditions in (1):

$$\begin{aligned}
\mathbf{b}_{\mathbf{n}1,j}(1) &= aN_{1,j}(n-2) \quad \text{(north boundary)} \\
\mathbf{b}_{\mathbf{n}M,j}(2) &= aS_{M,j}(n-2) \quad \text{(south boundary)} \\
\mathbf{b}_{\mathbf{n}i,M}(3) &= aE_{i,M}(n-2) \quad \text{(east boundary)} \\
\mathbf{b}_{\mathbf{n}i,1}(4) &= aW_{i,1}(n-2) \quad \text{(west boundary).}
\end{aligned} \quad (8)$$

2.1. Mesh Input and Output

The mesh may be "excited" at a location corresponding to junction $J_{i,j}$ using an input signal $x(n)$ symmetrically distributed about the junction's port inputs (or equivalently, the neighbouring junction's port outputs), such that the output signal $y(n)$ is the junction's total velocity:

$$\begin{aligned}
y(n) &= vJ_{i,j}(n) \\
&= \frac{1}{2}\sum_{u=1}^{4} \mathbf{b}_{\mathbf{n}i,j}(u) + x(n) \\
&= \frac{1}{2}\sum_{u=1}^{4} \left(\mathbf{b}_{\mathbf{n}}(u) + \frac{x(n)}{2}\right),
\end{aligned} \quad (9)$$

with boundaries according to (8). Equation (9) can be further reduced if the I/O location is at the center,

$$\begin{aligned}
y(n) &= vJ_{\frac{M+1}{2},\frac{M+1}{2}}(n) \quad \text{for odd } M, \\
&= \frac{1}{2}\left(4S_{\frac{M-1}{2},\frac{M+1}{2}}(n-1) + 4\frac{x(n)}{2}\right) + ... \\
&= 2S_{\frac{M-1}{2},\frac{M+1}{2}}(n-1) + x(n),
\end{aligned} \quad (10)$$

where, due to symmetry, the input on all ports is equal.

3. REDUCTION BY MESH SYMMETRY

Here, the input/output location is chosen to be the center of a square $M \times M$ mesh, for odd M, creating a symmetry and redundancy that reduces computation (not taken advantage of in a typical mesh implementation which performs the same computation regardless of the I/O location). Of course, this same symmetry that reduces computation also results in cancelled modal resonances—the most reduced spectrum for a given dimension.

When the mesh is excited at the center junction $J_{\frac{M+1}{2},\frac{M+1}{2}}$, this junction's port outputs are equal in all directions,

$$N_{\frac{M+1}{2},\frac{M+1}{2}} = S_{\frac{M+1}{2},\frac{M+1}{2}} = E_{\frac{M+1}{2},\frac{M+1}{2}} = W_{\frac{M+1}{2},\frac{M+1}{2}}, \quad (11)$$

(with n being suppressed for compactness), creating a symmetry that is propagated outward to the mesh edges, and a redundancy requiring computation of only half a quadrant of the entire mesh (as indicated by shading in (12)),

$$\begin{bmatrix}
J_{1,1} & J_{1,2} & \dots & J_{1,\frac{M+1}{2}} & \dots & J_{1,M} \\
J_{2,1} & J_{2,2} & \dots & J_{2,\frac{M+1}{2}} & \dots & J_{2,M} \\
J_{3,1} & J_{3,2} & \dots & J_{3,\frac{M+1}{2}} & \dots & J_{3,M} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
J_{\frac{M+1}{2},1} & J_{\frac{M+1}{2},2} & \dots & J_{\frac{M+1}{2},\frac{M+1}{2}} & \dots & J_{\frac{M+1}{2},M} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
J_{M,1} & J_{M,2} & \dots & J_{M,\frac{M+1}{2}} & \dots & J_{M,M}
\end{bmatrix} \quad (12)$$

a reduction from M^2 junctions to

$$Nj = \frac{\frac{M+1}{2}\left(\frac{M+1}{2}+1\right)}{2}, \quad \text{for odd } M. \quad (13)$$

[1] Notation whereby column vectors are notated with parentheses and commas is adopted when compactness is necessary

3.1. New Boundary Conditions

Reducing the mesh to half a quadrant introduces new boundaries (points of symmetry) along the mesh's first half of the

- **center column (col)** ($J_{1,\frac{M+1}{2}}, J_{2,\frac{M+1}{2}}, \ldots, J_{\frac{M+1}{2},\frac{M+1}{2}}$):

$$W_{i,j+1}(n-1) = E_{i,j-1}(n-1), \text{ for } j = \frac{M+1}{2}. \tag{14}$$

- **quadrant diagonal (diag)** ($J_{1,1}, J_{2,2}, \ldots, J_{\frac{M+1}{2},\frac{M+1}{2}}$):

$$\left.\begin{aligned} N_{i+1,j}(n-1) &= W_{i,j+1}(n-1) \\ E_{i,j-1}(n-1) &= S_{i-1,j}(n-1) \end{aligned}\right\} \quad \text{for } i = j. \tag{15}$$

Employing these symmetries (14-15) alone in a direct implementation of (4-8), (10)), and can significantly reduce mesh computation times (see Table 1, column 3 vs. column 2).

4. OBTAINING THE TRANSFER FUNCTION

Taking the z-transform of (4) and rearranging elements (for convenience in algorithm) yields

$$\mathbf{J}_{\mathbf{z}i,j} = \mathbf{H}_{\mathbf{z}i,j}\mathbf{b}'_{\mathbf{z}i,j}, \tag{16}$$

where for non boundaries

$$\mathbf{J}_{\mathbf{n}i,j} = \left(W_{i,j}(z)N_{i,j}(z), E_{i,j}(z), S_{i,j}(z)\right), \tag{17}$$

$$\mathbf{b}_{\mathbf{z}i,j} = (E_{i,j-1}(z), S_{i-1,j}(z), W_{i,j+1}(z), N_{i+1,j}(z)), \tag{18}$$

and

$$\mathbf{H}_{\mathbf{z}i,j} = \frac{z^{-1}}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}, \tag{19}$$

and where boundary conditions are later applied following (8) (see Section 4.1).

At this point, the mesh is represented as a system of equations for each junction (expressed as a function of neighbouring junctions) which, when excited and tapped in the center, yields an output as a function of the southern port output of the center junction's northern neighbour and the input (9), the z-transform of which is given by

$$Y(z) = 2S_{\frac{M-1}{2},\frac{M+1}{2}}(z)z^{-1} + X(z). \tag{20}$$

What is needed to produce a transfer function however, is an output solely as a function of the input,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{N_b} z^{N_b}}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{N_a}}, \tag{21}$$

or alternatively (and perhaps preferably) in factored form. Thus, beginning with (16) and an initialization step, dependencies of junctions on their neighbors is sequentially eliminated, starting at the top row, and then working down the (triangular) mesh.

4.1. Initialization

Elements in (16) can be initialized to values that, in the case of boundaries, lead to reduced sizes for $\mathbf{J}_\mathbf{z}$, $\mathbf{b}_\mathbf{z}$ and $\mathbf{H}_\mathbf{z}$. Applying boundaries (8, 14, 15), yields initial conditions

$$\mathbf{J}_{i,j} = \begin{cases} E_{1,1}(z), & \text{top left corner} \\ N_{\frac{M+1}{2},\frac{M+1}{2}}(z), & \text{bottom apex} \\ \left(W(z), \quad S(z)\right)_{1,\frac{M+1}{2}}, & \text{top right corner} \\ \left(N(z), \quad E(z)\right)_{i,j}, & \text{diagonal } (i=j) \\ \left(W(z), \quad E(z), \quad S(z)\right)_{1,j}, & \text{top row} \\ \left(W(z), \quad N(z), \quad S(z)\right)_{i,\frac{M+1}{2}}, & \text{right column} \end{cases} \tag{22}$$

and the corresponding $\mathbf{b}_\mathbf{z}$ vector with cases given respectively (but with case statements removed for brevity):

$$\mathbf{b}_{\mathbf{z}i,j} = \begin{cases} W_{1,2}(z), \\ \left(S_{\frac{M-1}{2},\frac{M+1}{2}}(z), \quad X(z)\right), \\ \left(E_{1,\frac{M-1}{2}}(z), \quad N_{2,\frac{M+1}{2}}(z)\right), \\ \left(S_{i-1,j}(z), \quad W_{i,j+1}(z)\right), \\ \left(E_{1,j-1}(z), \quad W_{1,j+1}(z), \quad N_{2,j}(z)\right), \\ \left(E_{i,\frac{M-1}{2}}(z), \quad S_{i-1,\frac{M+1}{2}}(z), \quad N_{i+1,\frac{M+1}{2}j}(z)\right), \end{cases} \tag{23}$$

and corresponding $\mathbf{H}_\mathbf{z}$ matrix:

$$\mathbf{H}_{\mathbf{z}i,j} = \begin{cases} az^{-3}, & \text{top right corner} \\ \begin{bmatrix} z^{-1} & z^{-1} \end{bmatrix} & \text{bottom apex} \\ \begin{bmatrix} H_r & H_b \\ 2H_b & H_a \end{bmatrix}, & \text{top right corner} \\ \begin{bmatrix} 0 & z^{-1} \\ z^{-1} & 0 \end{bmatrix}, & \text{diagonal} \\ \begin{bmatrix} H_a & H_b & H_b \\ H_b & H_a & H_b \\ H_b & H_b & H_a \end{bmatrix}, & \text{top row} \\ \begin{bmatrix} 0 & \frac{z^{-1}}{2} & \frac{z^{-1}}{2} \\ z^{-1} & -\frac{z^{-1}}{2} & \frac{z^{-1}}{2} \\ z^{-1} & \frac{z^{-1}}{2} & -\frac{z^{-1}}{2} \end{bmatrix}, & \text{right column} \end{cases} \tag{24}$$

where

$$H_r = \frac{az^{-3}}{2 + az^{-2}}, \quad H_a = \frac{-z^{-1}}{2 + az^{-2}}, \quad H_b = \frac{az^{-3} + z^{-1}}{2 + az^{-2}}, \tag{25}$$

4.2. Dependency Elimination Algorithm

Eliminations are done by updating and operating on the dependency matrix $\mathbf{H}_\mathbf{z}$. Vector $\mathbf{b}_\mathbf{z}$ is shown for reference only. First steps are shown for illustration, but later supressed for brevity.

4.2.1. Traverse first row from left to right:

eliminate $E_{1,1}$ in $\mathbf{b}_{\mathbf{z}1,2}$:

$$\mathbf{J}_{\mathbf{z}1,2} = \mathbf{H}_{\mathbf{z}1,2}\left(\cancel{E_{1,1},}\overset{W_{1,2}}{\nearrow} W_{1,3}, \quad N_{2,2}\right)$$

by multiplying the first column of $\mathbf{H}_{\mathbf{z}1,2}$ by the first row of $\mathbf{H}_{\mathbf{z}1,1}$ and updating $\mathbf{H}_{\mathbf{z}1,2}$:

$$\mathbf{H}_{\mathbf{z}1,2} \rightarrow \left[\begin{bmatrix} H_{1,1} \\ H_{2,1} \\ H_{3,1} \end{bmatrix} \times \begin{bmatrix} H_{1,1} \end{bmatrix}_{1,1} \quad \begin{matrix} H_{1,2} & H_{1,3} \\ H_{2,2} & H_{2,3} \\ H_{3,2} & H_{3,3} \end{matrix}\right]_{1,2}$$

eliminate $W_{1,2}$ in $\mathbf{b}_{\mathbf{z}1,2}$ (recursive)

$$\mathbf{J}_{1,2} = \mathbf{H}_{\mathbf{z}1,2}\left(\cancel{W_{1,2},} \quad W_{1,3}, \quad N_{2,2}\right),$$

$$\begin{aligned}\mathbf{H}_{\mathbf{z}1,2} \quad \rightarrow \quad & \left[\begin{bmatrix} 1 \\ H_{2,1} \\ H_{3,1} \end{bmatrix} \times \begin{bmatrix} H_{1,2} & H_{1,3} \end{bmatrix}\right]_{1,2} / (1 - \begin{bmatrix} H_{1,1} \end{bmatrix}_{1,2}) \\ & + \begin{bmatrix} 0 & 0 \\ H_{2,2} & H_{2,3} \\ H_{3,2} & H_{3,3} \end{bmatrix}_{1,2} \end{aligned} \tag{26}$$

eliminate $E_{1,2}$ from $\mathbf{b}_{\mathbf{z}1,3}$

$$\mathbf{J}_{1,3} = \mathbf{H}_{\mathbf{z}1,3}\left(\cancel{E_{1,2},}\overset{W_{1,3}}{\nearrow} W_{1,4}, \quad N_{2,3}\right),$$

$$\mathbf{H}_{\mathbf{z}1,3} \rightarrow \left[\begin{bmatrix} H_{1,1} \\ H_{2,1} \\ H_{3,1} \end{bmatrix} \times \begin{bmatrix} H_{2,1} & H_{2,2} \end{bmatrix}_{1,2} \quad \begin{matrix} H_{1,2} & H_{1,3} \\ H_{2,2} & H_{2,3} \\ H_{3,2} & H_{3,3} \end{matrix}\right]_{1,3}$$

eliminate $W_{1,3}$ in $\mathbf{b}_{\mathbf{z}1,3}$

$$\mathbf{J}_{1,3} = \mathbf{H}_{\mathbf{z}1,3}\left(\cancel{W_{1,3},}\overset{N_{2,2}}{\nearrow} W_{1,4}, \quad N_{2,3}\right), \tag{27}$$

$$\begin{aligned}\mathbf{H}_{\mathbf{z}1,3} \rightarrow & \left[\begin{bmatrix} 1 \\ H_{2,1} \\ H_{3,1} \end{bmatrix} \times \begin{bmatrix} H_{1,2} & H_{1,3} & H_{1,4} \end{bmatrix}\right]_{1,3} \\ & / \left(1 - \begin{bmatrix} H_{1,1} \end{bmatrix}_{1,3}\right) + \begin{bmatrix} 0 & 0 & 0 \\ H_{2,2} & H_{2,3} & H_{2,4} \\ H_{3,2} & H_{3,3} & H_{3,4} \end{bmatrix}_{1,3} \end{aligned}$$

eliminate $E_{1,j}$ in $J_{1,j+1}$ to row end;
eliminate $W_{1,j+1}$ in $J_{1,j+1}$ to row end;

4.2.2. Traverse first row from right to left:

eliminate $W_{1,j}$ from $J_{1,j-1}$, for $3 <= j <= \frac{M+1}{2}$.

4.2.3. Traverse from first to second row

eliminate $S_{1,j}$ in $J_{2,j}$, for $2 <= j <= \frac{M+1}{2}$.

4.2.4. Traverse second row from left *to* right

eliminate $E_{2,j}$ in $J_{2,j+1}$ to row end;
eliminate $W_{2,j+1}$ in $J_{2,j+1}$ to row end;

4.2.5. Traverse second row from right *to* left

eliminate $W_{2,j}$ from $J_{2,j-1}$ to row end;

4.2.6. Traverse second row from left *to* right

eliminate $N_{2,j}$ from $J_{2,j}$, $J_{2,j+1}$, for $2 <= j <= \frac{M+1}{2}$

4.2.7. Traverse to next (and subsequent) rows and repeat

Once the bottom junction is reached, the transfer function is given by the final expression in $\mathbf{H}_{\mathbf{z}\,\frac{M+1}{2},\frac{M+1}{2}}$ for an unit-impulse input.

MXM (order)	full mesh	symmetric mesh	DF II
3×3 (8)	2 ms	0.7 ms	0.6 ms
5×5 (16)	6 ms	1.1 ms	1.0 ms
7×7 (26)	13 ms	1.2 ms	1.5 ms
9×9 (38)	22 ms	4.1 ms	2.0 ms
11×11 (52)	34 ms	6.5 ms	2.9 ms

Table 1. Mesh computation times for the square $M \times M$ mesh, mesh with applied symmetry, and direct form II implementation of transfer function with given order (and where comparable timings are expected for a second-order section cascade) for one second of 44100-Hz-audio).

5. CONCLUSION

Because of the computational limitations of the 2-D waveguide mesh, it is not often used for real-time applications. This work presents an initial step toward creating a frequency-domain representation, with the eventual aim of developing a real-time implementation. Though the current algorithm yields a transfer function, which for low-orders can be implemented in canonical form (direct form II), higher orders experience coefficient quantization that perturbs the poles, often outside the unit circle, and leading to instabilities. Furthermore, direct form II would not be an ideal real-time implementation due to known instabilities that arise when coefficients are made time varying. It is thus desirable to obtaining the transfer function in factored form so that alternate implements, such as a back of second order sections, or a parametric impulse response, can be used. This work is being developed in Mathematica, a symbolic programming language, allowing for analysis of symbolic (rather than numeric) results, less prone to precision error. Table 1 shows current timing results.

REFERENCES

[1] S. V. Duyne and J. O. Smith, "The 2-d digital waveguide mesh," in *Proceedings of the 1993 IEEE Workshop of Applied Signal Processing to Audio and Acoustics*. New Paltz, New York: IEEE Press, October 1993.

[2] D. T. Murphy, C. J. C. Newton, and D. M. Howard, "Digital waveguide mesh modelling of room acoustics: Surround-sound, boundaries and plugin implementation," in *Proceedings of the Conference on Digital Audio Effects (DAFX-01)*, Limerick, Ireland, December 2001.