A Generalization of the Hadamard Transform

Robin Schmidt (www.rs-met.com)

August 3, 2011

We consider a generalization of the Hadamard transform that contains 4 freely adjustable parameters and still can use the same efficient computational structure that is used for the fast Hadamard transform. By imposing some relations between the 4 parameters, we will obtain a two-parametric family of unitary transforms that are suitable for use in a feedback delay network (FDN) for artificial reverberation. Eventually, we'll also relate the two remaining parameters to ultimately arrive at a one-parametric family of unitary transforms.

The Hadamard Transform

Following [1], we'll define the Hadamard transform as a transformation of an input vector $\mathbf{x}$ to an output vector $\mathbf{y}$ via a multiplication with a matrix $\mathbf{H}_L$, where the index L is the base-2 logarithm of the size of the matrix:

$$\mathbf{y} = \mathbf{H}_L \, \mathbf{x} \tag{1}$$

The smallest transformation that makes sense is the case for $L = 1$, such that the size of the matrix is $N = 2^1 = 2$. This most basic transform matrix is given by:

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2}$$

Subsequent higher order matrices are obtained by applying a recursive construction using $\mathbf{H}_1$ as seed. This is known as the Sylvester construction and proceeds as follows:

$$\mathbf{H}_{L+1} = \begin{pmatrix} \mathbf{H}_L & \mathbf{H}_L \\ \mathbf{H}_L & -\mathbf{H}_L \end{pmatrix} \tag{3}$$

...well, formally one could also start this recursion with the scalar-matrix $\mathbf{H}_0 = 1$.

The Generalization

We generalize the construction above by starting with an arbitrary seed matrix for the $L = 1$ case, such that:

$$\mathbf{M}_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{4}$$

where we have 4 free parameters a, b, c, d. The recursive construction of higher order matrices proceeds as follows:

$$\mathbf{M}_{L+1} = \begin{pmatrix} a\mathbf{M}_L & b\mathbf{M}_L \\ c\mathbf{M}_L & d\mathbf{M}_L \end{pmatrix} \tag{5}$$

from which we easily see, that the whole thing reduces to the standard Sylvester construction for $a = b = c = 1, d = -1$.

The Fast Algorithm

The nice thing about a so constructed matrix is, that the matrix-vector multiplication $\mathbf{y} = \mathbf{M}_L \, \mathbf{x}$ can be carried out in $\mathcal{O}(N \cdot L) = \mathcal{O}(N \cdot \log_2(N))$ operations via the algorithm (in pseudo MatLab/Octave):

```
for i=1:L
 for j=1:N/2
  y(j)     = a*x(2*j-1) + b*x(2*j);
  y(j+N/2) = c*x(2*j-1) + d*x(2*j);
 end
 x = y; % reused for intermediate result
end
```

With $a = b = c = 1, d = -1$, this algorithm reduces to the fast Hadamard transform (without scaling and sequency-based ordering) and the multiplications inside the inner loops could be thrown away due to the fact that the factors would reduce to ± 1.

The Inverse Transform

For the $L = 1$ case, which transforms a 2-dimensional vector, the matrix that produces the inverse transform can be calculated directly via:

$$\mathbf{M}_1^{-1} = \frac{1}{D_1} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \qquad \text{where} \qquad D_1 = \det(\mathbf{M_1}) = ad - bc \tag{6}$$

from which we conclude that $\mathbf{M}_1$ must be non-singular (i.e. the determinant D_1 must be non-zero). This is the general condition for the existence of an inverse matrix. Let us denote the elements of the inverse matrix as a_i, b_i, c_i, d_i, so we can write:

$$\mathbf{M}_1^{-1} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \qquad \text{where} \qquad a_i = \frac{d}{D_1}, b_i = -\frac{b}{D_1}, c_i = -\frac{c}{D_1}, d_i = \frac{a}{D_1} \tag{7}$$

As it turns out, we may also construct the inverses of the higher-order inverse matrices $\mathbf{M}_L^{-1}, L > 1$ via the very same recursive construction that we have used to construct the matrices for the forward transform. That is, we may construct $\mathbf{M}_{L+1}^{-1}$ from $\mathbf{M}_L^{-1}$ via:

$$\mathbf{M}_{L+1}^{-1} = \begin{pmatrix} a_i\mathbf{M}_L^{-1} & b_i\mathbf{M}_L^{-1} \\ c_i\mathbf{M}_L^{-1} & d_i\mathbf{M}_L^{-1} \end{pmatrix} \tag{8}$$

which implies that we may use the same fast algorithm to compute the inverse transform, but with a_i, b_i, c_i, d_i instead of a, b, c, d. As of yet, i have not derived formal proof for this but experiments indicate that it indeed works like that.

Imposing Restrictions on the Parameters

So far, we may choose a, b, c, d arbitrarily. If we impose some restrictions on this choice, we get matrices that have some properties that are desirable for a feedback matrix in an FDN. Specifically, if the parameters have the either the relationship $(1) : c = b, d = -a$ or $(2) : c = -b, d = a$, the matrix $\mathbf{M}_L$ will satisfy $\mathbf{M}_L^T \mathbf{M}_L = \text{diag}(k)$ for some constant k that depends on a and b via the formula:

$$\boxed{k = (a^2 + b^2)^L} \tag{9}$$

To obtain an unitary transform, we would have to scale the whole matrix by $1/\sqrt{k}$. In terms of the fast algorithm, this just means that we scale our resulting vector with that value. A further specialization of the first case to $a = b = 1$ again reduces to the standard Hadamard transform. Let's have a closer look at the 2nd case. We use the seed matrix:

$$\mathbf{M}_1 = \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \tag{10}$$

The next matrix $\mathbf{M}_2$ looks like:

$$\mathbf{M}_2 = \begin{pmatrix} a\mathbf{M}_1 & b\mathbf{M}_1 \\ -b\mathbf{M}_1 & a\mathbf{M}_1 \end{pmatrix} = \begin{pmatrix} a^2 & ab & ab & b^2 \\ -ab & a^2 & -b^2 & ab \\ -ab & -b^2 & a^2 & ab \\ b^2 & -ab & -ab & a^2 \end{pmatrix} \tag{11}$$

from which we see that the main diagonal is solely populated with a^2, the other diagonal is populated with $\pm b^2$ and the off-diagonal elements are populated by the cross-terms. By inspection, we may convince ourselves that for $\mathbf{M}_3$, we would see a^3 on the main diagonal, $\pm b^3$ on the other diagonal and it goes on that way for higher order matrices.

Application in an FDN

Even with one of the two restrictions on the choices for a, b, c, d, as described above, we still have obtained a two-parametric family of unitary transforms (assuming that we do the division by $1/\sqrt{k}$). We may voluntarily further reduce the number of free parameters down to a single parameter by somehow relating a and b. For example, we could use $a = \cos(\phi), b = \sin(\phi)$ in which case the normalization constant k also reduces to unity. The second case for the restriction: $c = -b, d = a$, together with the parameterization via ϕ could be an interesting choice to be used in a feedback delay network for artificial reverberation. Via the parameter ϕ, we would have a macro-parameter that allows us to control the amount of scattering between the delaylines. For $\phi = n\pi, n \in \mathbb{Z}$, there would be no scattering at all - the whole FDN would reduce to a bank of parallel comb filters in this case. We can get a feeling for this by looking at equation (11) when we realize that the diagonal terms produce self-feedback and the off-diagonal elements produce cross-feedback. If $|a|$ is unity and b is zero, which is the case for that choice of ϕ, we see that there is only self-feedback. The sign of a will determine whether these combs produce a full series of harmonics (when $a = +1$ which is the case for even n) or only odd harmonics (for odd n, such that $a = -1$). By using $\phi = \frac{\pi}{2} + n\pi, n \in \mathbb{Z}$, there would only be some crossfeedback between pairs of delaylines (via the b^2 elements). If, on the other hand, $\phi = \frac{\pi}{4} + n\frac{\pi}{2}, n \in \mathbb{Z}$, which are the values of ϕ where sine and cosine have equal absolute values (of $1/\sqrt{2}$), scattering is maximal as we would have a feedback matrix where all entries have the same absolute value. We could provide a "scatter", "diffusion", "density" parameter to the user by mapping some user input range (say p = 0%...100%) either to the range $0...\frac{\pi}{4}$ or to the range $\frac{\pi}{2}...\frac{\pi}{4}$. In the former case we would only see self-feedback for $p = 0$ and in the latter case we would only see pairwise crossfeedback for $p = 0$. A FDN based on such a generalized Hadamard transform may also lend itself well to modulation of the pole locations, because we can easily modulate ϕ.

References

[1] Charles Constantine Gumas. A century old, the fast Hadamard transform proves useful in digital communications