Wendy Bergh

Thesis Advisor: Dr. Eric Lund

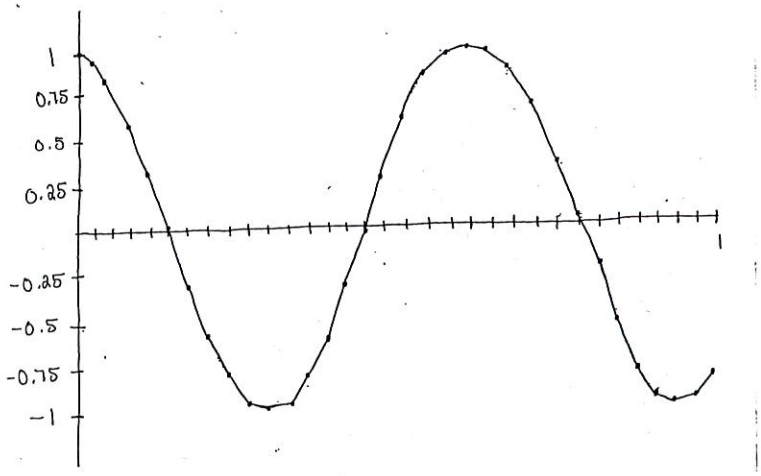
An Investigation of Wavelets

In everyday terms, a *wavelet* would probably be understood to be a small wave. To a mathematician, however, there is a more specific and technical usage for this term. In the mathematical world of signal processing, wavelets are the basis elements of various "wavelet spaces," which in turn are orthogonal complements of other vector spaces. Although the theory behind wavelets is somewhat complicated, the vector spaces that give rise to them (to Haar wavelets, anyway, which will be the main subject of this paper) have a very straightforward definition. In this paper we will examine Haar wavelets, showing the utility of this mathematic topic, presenting some calculations involving them, and explaining the mathematical terminology already mentioned, such as *vector space* and *orthogonal complement*. Then we will discuss how the mathematics behind Haar wavelets can be extended to develop other families of wavelets, families that are more commonly used in actual practice.

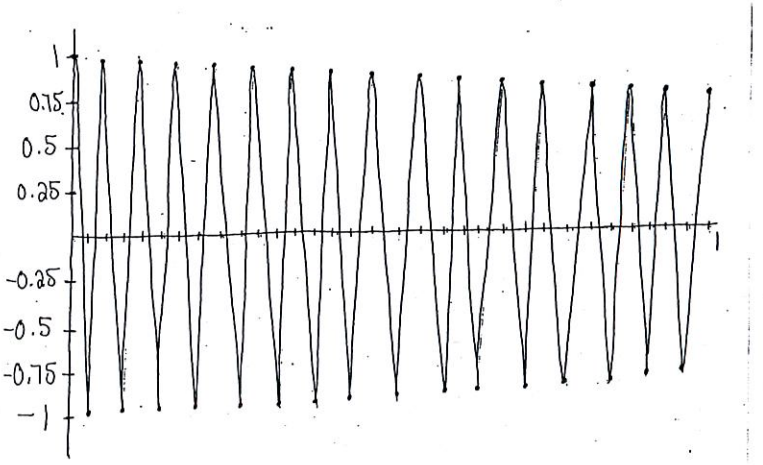
Say that we have a set of data points (x_i, y_i) such that each of these points is a sample of a function y = f(x). One way to display these data points is to plot each individual point and then connect all neighboring points with line segments. When computer-algebra packages graph functions, this is what they are doing. Using uniformly-spaced sample points often isn't the best approach to plotting a function unless we use a large number of points. Here is an example of how using sample points can be

problematic. Consider the functions y = cos(10x) and y = cos(100x). On the interval [0,1], using 32 equally spaced x-values, the graphs of these functions look like this:

y = cos(10x)



y = cos(100x)



The actual graph of y = cos(10x) is very close to the one that we just plotted. However, the actual graph of y = cos(100x) does not look like the other one we just plotted. The graph of y = cos(100x) that we plotted leads us to believe that it is a function which oscillates in a pulsing pattern, when our intuition leads us to believe that it should be just a horizontally telescoped version of the y = cos(10x) that we plotted. Our intuition is correct, but the uniform sampling of only 32 points results in a graph unlike,

although close to, what is correct. The phenomenon in the graph that deviates from our expectations is known as *aliasing*: sometimes this results in a jagged, stairstep effect on curved or diagonal lines that are reproduced in low resolution, as on a computer display. If we were to use more sampled points to replot the y = cos(100x) function, we would see a faster repeating form of the y = cos(10x) graph we plotted, just like we would expect to see.

Simply sampling and connecting points of a function is not the only way to get an approximate graph of that function. Another way uses the idea of sampling points, but focuses on clustering points where a function has a large amount of variation. This is an *adaptive plotting routine* that examines angles between connecting line segments in "provisional internally-generated" plots which are based on uniform sampling. After identifying regions of large variation, the adaptive plotting routine will produce a visible plot following the subdivision of certain intervals. Wavelets give rise to an adaptive plotting scheme that does not require any person or computer to consider angles first when looking at default plots. Concerning these two methods of graphing a function, it seems that if we continually replot a function with more and more samples we will get numerous graphs that eventually converge to the true graph. This is generally true using the first type of sampling mentioned, uniform sampling, and the second type of sampling mentioned, uniform sampling with clustering where the function has significant variation. However, it doesn't matter what scale we utilize, there will come a time where we won't know if additional sampled points are necessary or even helpful. The question is this: How many points do we need to plot to get the correct graph? The answer to this question depends highly on the amount of variation the function has over the chosen interval and the size of the picture we are going to look at.

The two examples of different methods to graph a function that I mentioned above work decently in most cases, but not always. Say we wanted to graph a function like $y = \cos\left(\frac{1}{x}\right)$. This function has an

infinite number of extrema on the interval (0,1]. That will cause problems for any plotting routine! If we were to plot the graph using 32 uniformly sampled points, the result would be misleading and not a true depiction of the function. To make the problem worse, if we were to use more points (uniformly or adaptively), the graph would appear much denser because of the thickness of the line segments connecting the points. Keep increasing the sampled points, and the graph will just fill up with the connecting line segments. Because of the limitations of plotters and printers, it is impossible to accurately graph the function $y = \cos\left(\frac{1}{x}\right)$. And this is not the only function that gives plotters or printers trouble-there are lots more.

Instead of joining points with line segments, we could also use the y-values as step levels for a staircase effect. In doing so, we would just use vertical lines to connect the steps of y-values. One might think that these vertical lines would be unwanted, or misleading, but that needn't be the case. The staircase method sometimes leads to better approximations of a true graph when more points are used, although a lot more are needed to minimize the jaggedness and obtain a continuous effect.

There are also questions of data storage and transmission. These become very important when looking at higher-dimensional analogues of data points in the plane, like in digital images. Let's think about images that are derived from two-dimensional arrays of *pixels*: numbers that represent gray-levels ranging from black (minimum number) to white (maximum number). These can be thought of as data points (x, y, z) where z measures the gray-level at position (x, y). Now what we really have are two-dimensional step functions where the steps are shaded according to height. If we have an image that is composed of 256×256 pixels, it is derived from a matrix of $256^2 = 65536$ pieces of data, each representing a gray-level. If we were to use a lower resolution for the image it would appear noticeably more "blocky" than a higher resolution image, so the higher the resolution, the better the image. This results in a requirement

of a lot of data to represent an image which leads to practical problems. Image files are time consuming to transmit. Anyone who has viewed pictures on the internet would agree. In most images, though, there are regions of little or no variation. The goal of wavelets is to take advantage of these somehow, and come up with a more economical way to store the matrices that represent the images.

There is a wavelet scheme for transforming and compressing digital data. These data can represent samples of a function or a matrix of gray levels (like in a picture image), for a couple of examples. The wavelet scheme is the same for either type of data. The main goal of wavelets is to transform data into a form where areas of low activity in the original data set become easy to find in the transformed version. This method can be applied to plotting functions because data strings allow us to identify uniformly sampled functions. Consider a string of eight data values: 10,10,20,16,24,32,32,48. This could be uniform samples of a function, or a row of an 8 x 8 pixel image, or something else. The processing of this data that I am going to explain is called averaging and differencing. This process can be generalized to any string of length equal to a power of two, and if the string length is not equal to a power of two we can just add zeros to the end of the string to satisfy this requirement. Now, back to our string of eight data values: 10,10,20,16,24,32,32,48. The rows of the following matrix show the steps that lead up to the final result of the averaging and differencing process:

$$\begin{bmatrix} 10 & 10 & 20 & 16 & 24 & 32 & 32 & 48 \\ 10 & 18 & 28 & 40 & 0 & 2 & -4 & -8 \\ 14 & 34 & -4 & -6 & 0 & 2 & -4 & -8 \\ 24 & -10 & -4 & -6 & 0 & 2 & -4 & -8 \end{bmatrix}.$$

The first row of the table is the original data string. If we think of the first row as four pairs of numbers, the first four numbers of the second row are the averages of the pairs in the first row. Continuing, the first two numbers in the third row are the averages of the four averages in the second row, taken as pairs, and the first entry in the last row is the average of the preceding two computed averages in the third row. The other numbers in each row measure the deviations from the averages. The last four numbers in the

second row are what we get when we subtract the first four averages from the first numbers of the pairs that made these averages. If we subtract 10, 18, 28, 40 from 10, 20, 24, 32, respectively, we get 0,2,-4,-8. These numbers are called *detail coefficients* and are repeated in every consecutive row on the table. The third and fourth numbers in the third row are from subtracting the first and second entries in that row from the first numbers of the pairs that are at the beginning of row two. If we subtract 14,34 from 10, 28 we get -4,-6. All computed detail coefficients are then repeated in each consecutive row after the row in which they are first computed. Finally, the second number in the last row, -10, is the detail coefficient from subtracting the overal average of 24 from the 14 that starts row three. The last average in the table, the first number of the fourth row, is the overall average of the original eight numbers. This overall average does not have an effect on the shape of any type of plot of the data: it just positions the data vertically. The seven detail coefficients are what determines the shape of the data. What we have done is transformed the original string of numbers into a new string of numbers. If we wanted to get the original string back, we would just work backwards. This is great because it means we didn't lose any information when we transformed our string of numbers. We also can slightly change the transformed version and still get a close approximation to the original data. When I say slightly change, I mean taking advantage of the areas that have low activity. The transformed string has one detail coefficient of 0 which came from the adjacent 10's in the original string. This suggests an area of low activity. The next smallest detail coefficient is 2. If we reset this to zero, it would make the last row look like 24, -10, -4, -6, 0, 0, -4, -8. If we work our way back to get the new string of numbers that would have come from our new last row, the table would look like this:

$$\begin{bmatrix} 10 & 10 & 18 & 18 & 24 & 32 & 32 & 48 \\ 10 & 18 & 28 & 40 & 0 & 0 & -4 & -8 \\ 14 & 34 & -4 & -6 & 0 & 0 & -4 & -8 \\ 24 & -10 & -4 & -6 & 0 & 0 & -4 & -8 \end{bmatrix}.$$

The first row in this new table is an approximation to the original data. Now let's plot the original data string and the approximation to the original data string using the numbers in the string as our y-values

with corresponding x-values of $\frac{1}{8}, \frac{2}{8}, \ldots, \frac{8}{8}$, respectively.

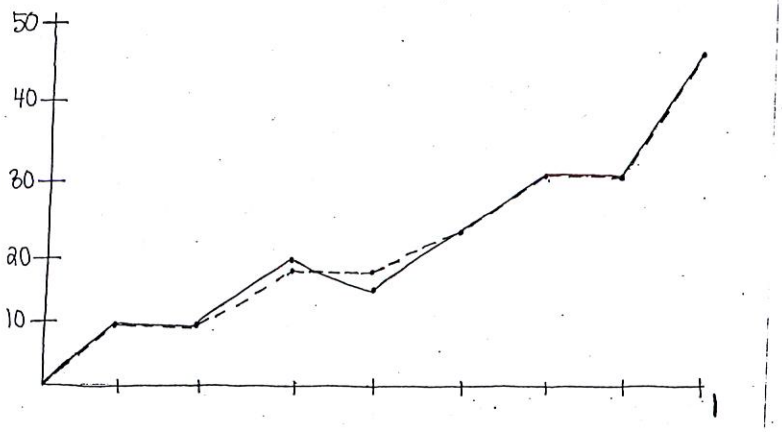
plot

of

10 10 20 16 24 32 32 48

and

10 10 18 18 24 32 32 48



You can see that there is a difference, but it is hardly noticeable. Even if we change a few more detail coefficients to 0, the new approximation to the original data set will still be very good. Let's change the -4's to 0 as a further demonstration. The last row would look like 24,-10,0,-6,0,0,0,-8. Again, working back to get the new string of numbers that would result from our new last row, the table would look like:

$$\begin{bmatrix} 14 & 14 & 14 & 14 & 28 & 32 & 32 & 48 \\ 14 & 14 & 28 & 40 & 0 & 0 & 0 & -8 \\ 14 & 34 & 0 & -6 & 0 & 0 & 0 & -8 \\ 24 & -10 & 0 & -6 & 0 & 0 & 0 & -8 \end{bmatrix}$$

. Also, let's plot the original and the second approximation to the original the same way we just did with the original and the first approximation.

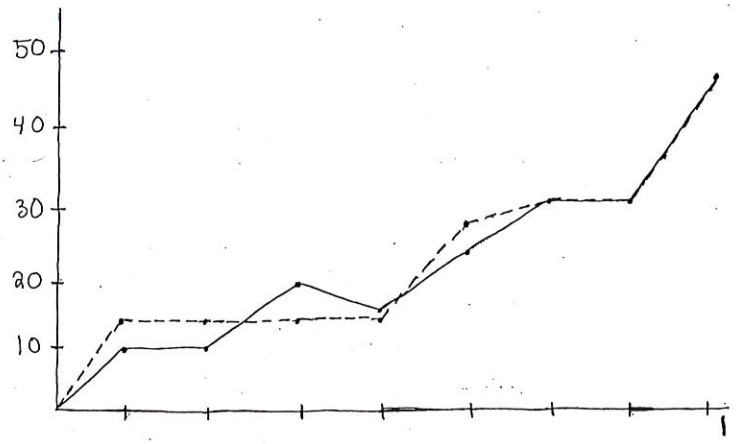
plot

of

10 10 20 16 24 32 32 48

and

14 14 14 14 28 32 32 48



This shows us that even though we are down to minimal data (only five numbers have been kept from the original last row using thresholding), the resulting approximation to the original data set is still very good. This is how the compression scheme works in general. Start with a data string and transform the string as we did above, then decide on a nonnegative threshold value ϵ. Reset to zero any detail coefficient whose absolute value is less than or equal to ϵ. This should lead to a relatively sparse data string that has a good proportion of zeros and is compressible when it comes to storing the data string. If we let $\epsilon = 0$, this process is called *lossless compression* because no information is lost and we can get our original string back. If we let $\epsilon > 0$, this process is called *lossy compression* because some, although potentially little, information will be lost and we will get an approximation of the original string based on the newly transformed string. Even after resetting a significant proportion of the detail coefficients to zero, we still generally get decent approximations to the original string of data. It is hard to see how useful this scheme is when we just use a string of eight numbers. Imagine, though, that we are working with a string of length $2^{10} = 1024$. After ten applications of the averaging and differencing process we would get a string with one overall average and 1023 detail coefficients. Now a substantial reduction of required storage space can be achieved if many of these detail coefficients are set to zero. And we can still always work our way back up to a good approximation of the original string.

Averaging and differencing can also be done with matrices, although for large data sets using matrices is not the most efficient approach. We will go through an example just to see how the matrix approach would work. Let A_1, A_2, A_3 be the following matrices, respectively:

$$\begin{bmatrix} 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & -0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & -0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & -0.5 \end{bmatrix}, \begin{bmatrix} 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The three-step transformation that we did previously when we went from the string of eight data 10, 10, 20, 16, 24, 32, 32, 48 to the approximation string 24, -10, -4, -6, 0, 2, -4, -8 can be carried out in terms of the following matrix equations:

$(10, 18, 28, 40, 0, 2, -4, -8) = (10, 10, 20, 16, 24, 32, 32, 48) \cdot A_1$;

$(14, 34, -4, -6, 0, 2, -4, -8) = (10, 18, 28, 40, 0, 2, -4, -8) \cdot A_2$;

$(24, -10, -4, -6, 0, 2, -4, -8) = (14, 34, -4, -6, 0, 2, -4, -8) \cdot A_3$.

The matrix multiplications carry out the task of averaging and differencing. And furthermore, the results of these three matrix equations can be accomplished by the single equation:

$(24, -10, -4, -6, 0, 2, -4, -8) = (10, 10, 20, 16, 24, 32, 32, 48) \cdot A_1 \cdot A_2 \cdot A_3$.

From basic linear algebra, since the columns of each A_i matrix are mutually orthogonal (from using the standard dot product), but no column is entirely zeros, this means that each of these matrices is invertible. The inverses reverse the three averaging and differencing steps that lead to the approximation string and give us the original string of data back. Doing so would look like:

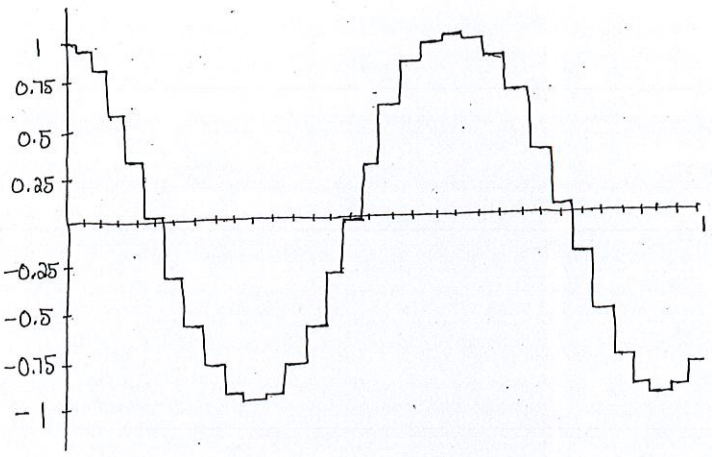
$(10, 10, 20, 16, 24, 32, 32, 48) = (24, -10, -4, -6, 0, 2, -4, -8) \cdot A_3^{-1} \cdot A_2^{-1} \cdot A_1^{-1}$.

This routine can be done in general to construct any corresponding $2^r \times 2^r$ matrices $A_1, A_2, \ldots, A_r$ needed to work with strings of data of length 2^r.

Now we can use some of the standard concepts and notations used in the general study of wavelets to give the previous example of averaging and differencing a firmer mathematical foundation. To do so, we present an alternative vector space description of our discrete, 8-member data sets. A *vector space* V is a set of elements (called vectors) that is closed under addition and scalar multiplication, on which the following axioms are satisfied for every **u**, **v**, and **w** in V and for all numbers r and s: a) $\mathbf{u}+\mathbf{v}=\mathbf{v}+\mathbf{u}$ b) $(\mathbf{u}+\mathbf{v})+\mathbf{w}=\mathbf{u}+(\mathbf{v}+\mathbf{w})$ c) there is a special member **0** (called the **zero vector**) of V such that $\mathbf{u}+\mathbf{0}=\mathbf{u}$ for all **u** in V d) for every member **u** of V there is a **additive inverse**, **-u**, in V such that $\mathbf{u}+(-\mathbf{u})=\mathbf{0}$ e) $r(\mathbf{u}+\mathbf{v})=r\mathbf{u}+r\mathbf{v}$ f) $(r+s)\mathbf{u}=r\mathbf{u}+s\mathbf{u}$ g) $(rs)\mathbf{u}=r(s\mathbf{u})$ h) $1\mathbf{u}=\mathbf{u}$. When all these properties are satisfied then V is called a vector space, and the members of V are called vectors. Moving on, we can identify data strings with a certain class of step functions. A string of length k is identified with the step function on [0,1] which (potentially) changes at $k-1$ equally spaced x-values and which uses the string entries as its step heights. These step functions can in turn be thought of as linear combinations of dyadically dilated and translated unit step functions on [0,1). For example, the string of y-values that comes from uniformly sampling the function f(x) = cos(10x) thirty-two times in [0,1] is identified with the step function plotted in this graph:

y = cos(10x)

step function



Consider the Haar scaling funtion: $\phi(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \textit{otherwise.} \end{cases}$ This function satisfies a *scaling equation* of the form $\phi(x) = \sum_{i \in \mathbb{Z}} c_i \varphi(2x - i)$, where in our case the only nonzero c_i's are $c_0 = c_1 = 1$, which makes our function look like $\phi(x) = \phi(2x) + \phi(2x - 1)$. For every i such that $0 \le i \le 7$, we get an induced (dyadically) dilated and translated scaling function of the form $\phi_i^3(x) = \phi(2^3 x - i)$. These eight functions form a basis for the vector space V_3 of piecewise constant functions on [0,1) with possible breaks at $\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \ldots, \frac{7}{8}$, where $\phi_0{}^3$ equals 1 on $[0, \frac{1}{8})$ only and is 0 elsewhere, $\phi_1{}^3$ is 1 on $[\frac{1}{8}, \frac{2}{8})$ only, $\phi_2{}^3$ is 1 on $[\frac{2}{8}, \frac{3}{8})$ only, and so on.

Now consider the element $10\phi_0{}^3 + 10\,\phi_1{}^3 + 20\phi_2{}^3 + 16\phi_3{}^3 + 24\phi_4{}^3 + 32\phi_5{}^3 + 32\phi_6{}^3 + 48\phi_7{}^3 \in V_3$, which is just another way of expressing our earlier data string 10, 10, 20, 16, 24, 32, 32, 48. We now have a step function representation of our data string. Any string of length eight can be identified with an element of V_3. We can describe the averaging and differencing scheme from earlier in terms of this version of data strings. To do so, we again need vector spaces and a few more definitions. Let $v_1, v_2, \ldots, v_n$ be vectors in a vector space V. The set S of all linear combinations of $v_1, v_2, \ldots, v_n$ is called the *span* of $v_1, v_2, \ldots, v_n$. We can also say these vectors *span* S. A *basis* for a vector space V is a set B of

vectors of V such that a) B is linearly independent and b) B spans V. Now, similarily to before, the four functions ϕ_i^2 defined by $\phi_i^2(x) = \phi(2^2x - i)$, for $0 \leq i \leq 3$, form a basis for the vector space V_2 of piecewise constant functions on [0,1) with possible breaks at $\frac{1}{4}$, $\frac{2}{4}$, $\frac{3}{4}$. And the two functions ϕ_i^1 defined by $\phi_i^1(x) = \phi(2^1x - i)$, for $0 \leq i \leq 1$, form a basis for the vector space V_1 of piecewise constant functions on [0,1) with a possible break at $\frac{1}{2}$. And finally, $\phi_0^0 = \phi$ itself is a basis for the vector space V_0 of constant functions on [0,1). This $\phi(t)$ is sometimes referred to as the father Haar wavelet. Note that there is a subspace relationship among these V_i vector spaces such that $V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3$.

We can identify the various averages derived in our earlier example of scheming with elements of these V_i vector spaces. To do so, we treat these averages as lower-resolution versions of the original string. This allows us to match 10, 18, 28, 40 with $10\phi_0^2 + 18\,\phi_1^2 + 28\,\phi_2^2 + 40\,\phi_3^2$, and 14 and 34 with $14\,\phi_0^1 + 34\,\phi_1^1$, and lastly 24 with $24\phi_0^0 = 24\phi$. Now we just need to find a new interpretation for the detail coefficients. This is where the wavelets come into play. Consider the *inner product* $\langle f, g \rangle = \int_0^1 f(t)g(t)\,dt$ defined on V_i. Two functions in V_i are *orthogonal* if and only if their inner product on [0,1] equals 0; that is, if the product function captures equal areas on each side of the horizontal axis. For every $j = 0, 1, 2$, we define the wavelet space W_j to be the orthogonal complement of V_j in V_{j+1}, so that we get the orthogonal direct sum decomposition of V_{j+1}: $V_{j+1} = V_j \oplus W_j$. Then we have the following relationships: $V_3 = V_2 \oplus W_2 = V_1 \oplus W_1 \oplus W_2 = V_0 \oplus W_0 \oplus W_1 \oplus W_2$.

Expressing step functions in V_3 in terms of these new bases brings us to the various detail coefficients

we came across before, which are known as wavelet coefficients.

The mother Haar wavelet is defined by $\psi(t) = \begin{cases} 1 & 0 \le t \text{ and } t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t \text{ and } t < 1 \\ 0 & \textit{otherwise} \end{cases}$. This new function is related to the father Haar wavelet by means of the equation $\psi(x) = \phi(2x) - \phi(2x-1)$. Then $\psi(x)$ is a basis for W_0 since ψ is orthogonal to ϕ. The four functions $\psi_i^2(x) = \psi(2^2x - i)$, for $0 \le i \le 3$, form a basis for W_2 because they are orthogonal to the corresponding functions $\phi_i^2 (0 \le i \le 3)$ which form a basis for the subspace V_2 of V_3. (We shall explore this orthogonality in greater detail a bit later in the paper.) Similarily, the two functions ψ_i^1 defined by $\psi_i^1(x) = \psi(2^1x - i)$, for $0 \le i \le 1$, form a basis for W_1.

Note the correspondence between the standard basis $\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$ for $\mathbb{R}^8$ and the basis $S_3 = \left\{ \phi_0^3, \phi_1^3, \phi_2^3, \phi_3^3, \phi_4^3, \phi_5^3, \phi_6^3, \phi_7^3 \right\}$ for V_3, and the correspondence between the non-standard basis

$$\left\{\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix},\begin{bmatrix}1\\1\\1\\1\\-1\\-1\\-1\\-1\end{bmatrix},\begin{bmatrix}1\\1\\-1\\-1\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\1\\1\\-1\\-1\end{bmatrix},\begin{bmatrix}1\\-1\\0\\0\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\\-1\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\1\\-1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\-1\end{bmatrix}\right\}$$ for $\mathbb{R}^8$ and the basis

$B_3 = \left\{\phi, \psi, \psi_0^{\,1}, \psi_1^{\,1}, \psi_0^{\,2}, \psi_1^{\,2}, \psi_2^{\,2}, \psi_3^{\,2}\right\}$ for V_3. We want to work with the basis B_3 instead of the more obvious basis S_3 because it is the wavelets that allow us to do the differencing and detailing, which in turn allows for the compression of data.

Using the notation from above, the three steps in the averaging and differencing transformation in the preceding section correspond to the following chain of identities:

$$10\phi_0^{\,3} + 10\,\phi_1^{\,3} + 20\phi_2^{\,3} + 16\phi_3^{\,3} + 24\phi_4^{\,3} + 32\phi_5^{\,3} + 32\phi_6^{\,3} + 48\phi_7^{\,3}$$

$$= 10\phi_0^{\,2} + 18\phi_1^{\,2} + 28\phi_2^{\,2} + 40\phi_3^{\,2} + 0\,\psi_0^{\,2} + 2\,\psi_1^{\,2} - 4\,\psi_2^{\,2} - 8\,\psi_3^{\,2}$$

$$= 14\phi_0^{\,1} + 34\phi_1^{\,1} - 4\,\psi_0^{\,1} - 6\,\psi_1^{\,1} + 0\,\psi_0^{\,2} + 2\,\psi_1^{\,2} - 4\,\psi_2^{\,2} - 8\,\psi_3^{\,2}$$

$$= 24\phi_0^{\,0} - 10\,\psi_0^{\,0} - 4\,\psi_0^{\,1} - 6\,\psi_1^{\,1} + 0\,\psi_0^{\,2} + 2\,\psi_1^{\,2} - 4\,\psi_2^{\,2} - 8\,\psi_3^{\,2}\,.$$

This fully-transformed version consists of one overall average and seven wavelet coefficients. This is simply a decomposition with respect to a very special basis. When we got rid of the smallest detail coefficients earlier (by setting them to zero) to get a string that was more sparse, all we were doing was setting some of the wavelet coefficients to zero. In our first compression example, we approximated

$$10\phi_0^{\,3} + 10\,\phi_1^{\,3} + 20\phi_2^{\,3} + 16\phi_3^{\,3} + 24\phi_4^{\,3} + 32\phi_5^{\,3} + 32\phi_6^{\,3} + 48\phi_7^{\,3}$$

$$= 24\phi_0^{\,0} - 10\,\psi_0^{\,0} - 4\,\psi_0^{\,1} - 6\,\psi_1^{\,1} + 0\,\psi_0^{\,2} + 2\,\psi_1^{\,2} - 4\,\psi_2^{\,2} - 8\,\psi_3^{\,2}$$

by the element $24\phi_0^{\,0} - 10\,\psi_0^{\,0} - 4\,\psi_0^{\,1} - 6\,\psi_1^{\,1} + 0\,\psi_0^{\,2} + 0\,\psi_1^{\,2} - 4\,\psi_2^{\,2} - 8\,\psi_3^{\,2}$. This idea can be

extended in the following way: for each nonnegative integer j, let V_j be the vector space of piecewise constant functions on [0,1) with possible breaks at $\frac{1}{2^j}, \frac{2}{2^j}, \frac{3}{2^j}, \ldots, 1-\frac{1}{2^j}$. Then the 2^j functions ϕ_i^j defined by $\phi_i^j(x) = \phi(2^j x - i)$, $0 \le i \le 2^j - 1$, form a basis for V_j. We then get an infinite ascending chain of vector spaces $V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3 \subseteq \ldots \subseteq V_j \subseteq V_{j+1} \subseteq \ldots$, each of which is an inner product space with respect to the inner product $\langle f, g \rangle = \int_0^1 f(t)g(t)dt$. The wavelet space W_j is then defined to be the orthogonal complement of V_j in V_{j+1}. The functions $\psi_i^j(x) = \psi(2^j x - i)$ for $0 \le i \le 2^j - 1$ form a basis for W_j and for any j we have

$$V_j = V_{j-1} \oplus W_{j-1} = V_{j-2} \oplus W_{j-2} \oplus W_{j-1} = \ldots = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \ldots \oplus W_{j-2} \oplus W_{j-1}.$$

This means that if we were to, say, work with a string of length 256, it would be equivalent to working in the larger space V_8, where $V_8 = V_0 \oplus W_0 \oplus W_1 \oplus \ldots \oplus W_6 \oplus W_7$.

At this point we have introduced the entire Haar-wavelet family: the father ϕ, the mother ψ, the daughters ($\psi_n^k(t) = \psi(2^n t - k)$), and the sons ($\phi_n^k(t) = \phi(2^n t - k)$). We will now show in greater detail how the mother and daughter wavelets are derived from the father and son wavelets by means of the orthogonal decomposition theorem. The *orthogonal decomposition theorem* says if W is a finite-dimensional subspace of an inner product space V, then any $v \in V$ can be written uniquely as $v = w + w^{\perp}$, where $w \in W$ and $w^{\perp} \in W^{\perp}$. (This theorem can be represented by $V = W \oplus W^{\perp}$.) The vector spaces V_1 and V_2 have the "natural" bases $S_1 = \{\phi_0^1, \phi_1^1\}$ and $S_2 = \{\phi_0^2, \phi_1^2, \phi_2^2, \phi_3^2\}$, respectively. Now, according to the theorem, any element of V_1 can be written as the sum of elements from V_0 and

$V_0{}^{\perp}$. But neither $\phi_0{}^1$ or $\phi_1{}^1$ is orthogonal to the basis element ϕ for V_0. So how can $\phi_0{}^1$ (in S_1) be written as the sum of ϕ and an element $\mathbf{w}^{\perp}$ from $V_0{}^{\perp}$? By utilizing the *Gram-Schmidt process*, it turns out that $\mathbf{w}^{\perp}$ is the *residual element* (in $V_0{}^{\perp}$) defined as $\phi_0{}^1$ minus the projection of $\phi_0{}^1$ onto ϕ.

Thus, $\mathbf{w}^{\perp} = \phi_0{}^1 - proj_{\phi}\phi_0{}^1 = \phi_0{}^1 - \frac{\langle \phi_0{}^1, \phi \rangle}{\langle \phi, \phi \rangle}\phi = \phi_0{}^1 - \frac{1}{2}\phi = \frac{1}{2}\psi$. Thus $\phi_0{}^1 = \frac{1}{2}\phi + \frac{1}{2}\psi$,

where $\phi \in V_0$ and $\psi \in V_0{}^{\perp}$.

Now let's check for $\phi_1{}^1 \in S_1$: how can $\phi_1{}^1$ (in S_1) be written as the sum of ϕ and an element $\mathbf{w}^{\perp}$ from $V_0{}^{\perp}$? Similarly to the previous calculation, we have

$\mathbf{w}^{\perp} = \phi_1{}^1 - proj_{\phi}\phi_1{}^1 = \phi_1{}^1 - \frac{\langle \phi_1{}^1, \phi \rangle}{\langle \phi, \phi \rangle}\phi = \phi_1{}^1 - \frac{1}{2}\phi = -\frac{1}{2}\psi$. So we have $\phi_1{}^1 = \frac{1}{2}\phi + -\frac{1}{2}\psi$. Since the two elements of the "natural" basis S_1 for vector space V_1 can be written in terms of ϕ and ψ, it is evident that we would take $B_1 = \{\phi, \psi\}$ as an alternative basis - the wavelet basis - for V_1, where $\phi \in V_0$ and $\psi \in V_0{}^{\perp}$. (Note that $V_0{}^{\perp}$ is spanned by ψ.) Again apply the orthogonal decomposition theorem, since $V_1 \subseteq V_2$, we have $V_2 = V_1 \oplus V_1{}^{\perp}$. Taking B_1 as the basis for V_1, what forms the basis for $V_1{}^{\perp}$? If we project each of the standard basis elements for V_2 onto V_1 we will see what the residuals are. First we will look at $\phi_0{}^2 \in S_2$: the projection of $\phi_0{}^2$ onto V_1 is

$\frac{\langle \phi_0{}^2, \phi \rangle}{\langle \phi, \phi \rangle}\phi + \frac{\langle \phi_0{}^2, \psi \rangle}{\langle \psi, \psi \rangle}\psi = \frac{\frac{1}{4}}{1}\phi + \frac{\frac{1}{4}}{1}\psi = \frac{1}{4}\phi + \frac{1}{4}\psi$. The residual is in $V_1{}^{\perp}$ and is

$\phi_0{}^2 - \frac{1}{4}\phi - \frac{1}{4}\psi = \frac{1}{2}\psi_0{}^1$. Second, $\phi_1{}^2 \in S_2$: the projection of $\phi_1{}^2$ onto V_1 is

$\frac{\langle \phi_1{}^2, \phi \rangle}{\langle \phi, \phi \rangle}\phi + \frac{\langle \phi_1{}^2, \psi \rangle}{\langle \psi, \psi \rangle}\psi = \frac{\frac{1}{4}}{1}\phi + \frac{\frac{1}{4}}{1}\psi = \frac{1}{4}\phi + \frac{1}{4}\psi$. The residual is in $V_1{}^{\perp}$ and is

$\phi_1^{\,2} - \frac{1}{4}\phi - \frac{1}{4}\psi = -\frac{1}{2}\psi_0^{\,1}$. Third, $\phi_2^{\,2} \in S_2$: the projection of $\phi_2^{\,2}$ onto V_1 is

$\frac{\langle \phi_2^{\,2}, \phi \rangle}{\langle \phi, \phi \rangle}\phi + \frac{\langle \phi_2^{\,2}, \psi \rangle}{\langle \psi, \psi \rangle}\psi = \frac{\frac{1}{4}}{1}\phi + \frac{-\frac{1}{4}}{1}\psi = \frac{1}{4}\phi - \frac{1}{4}\psi$. The residual is in $V_1^{\perp}$ and is

$\phi_2^{\,2} - \frac{1}{4}\phi + \frac{1}{4}\psi = \frac{1}{2}\psi_1^{\,1}$. Fourth and lastly, $\phi_3^{\,2} \in S_2$: the projection of $\phi_3^{\,2}$ onto V_1 is

$\frac{\langle \phi_3^{\,2}, \phi \rangle}{\langle \phi, \phi \rangle}\phi + \frac{\langle \phi_3^{\,2}, \psi \rangle}{\langle \psi, \psi \rangle}\psi = \frac{\frac{1}{4}}{1}\phi + \frac{-\frac{1}{4}}{1}\psi = \frac{1}{4}\phi - \frac{1}{4}\psi$. The residual is in $V_1^{\perp}$ and is

$\phi_3^{\,2} - \frac{1}{4}\phi + \frac{1}{4}\psi = -\frac{1}{2}\psi_1^{\,1}$. $V_1^{\perp}$ is a two-dimensional vector space. We see that it is spanned by $\{\psi_0^{\,1}, \psi_1^{\,1}\}$. Basically, $V_0^{\perp}$ is spanned by ψ and when V_2 is written as $V_0 \oplus V_0^{\perp} \oplus V_1^{\perp}$, it turns out that $V_1^{\perp}$ is spanned by $\psi_0^{\,1}$ and $\psi_1^{\,1}$. We could keep this process going to find that the next generation of daughters are the last four elements of a basis B_3 which looks like $\{\phi, \psi, \psi_0^{\,1}, \psi_1^{\,1}, \psi_0^{\,2}, \psi_1^{\,2}, \psi_2^{\,2}, \psi_3^{\,2}\}$ and the process continues to spaces V_n with larger values of n.

While the Haar wavelets are the simplest to understand and work with, there are many other wavelet families. As was the case with the Haar wavelets, each family is generated by a father wavelet or scaling function ϕ. Examples of such families are the *hat wavelets,* the *quadratic Battle-Lemarie' wavelets,* the *cubic Battle-Lemarie' wavelets,* and the *Shannon wavelets.* Recall the relationship between the Haar father and mother wavelets: $\psi(x) = \phi(2x) - \phi(2x-1)$. The relationship between mother and father wavelets is unique to each family of wavelets and this relationship only applies to the Haar family. Another wavelet family worth mentioning is the *Mexican hat family.* This family is different than the others we have encountered so far because there is no simple form for the scaling function. There is,

however, a simple representation of the mother wavelet. It is important to note that in most wavelet families there is no simple closed-form representation of either the father or mother functions. Instead, the scaling functions are defined by certain properties (relations), and are then approximated.

We used the Haar wavelets above because for this wavelet family, ϕ and ψ are readily defined, a property that makes them ideal for demonstrating concepts. However, the Haar wavelets are not used in practice because they lack some important properties. Wavelets that are typically used in applications are constructed to satisfy certain criteria. The standard approach is to first build a *multiresolution analysis* (MRA) and then construct the wavelet family with the desired criteria from the MRA. A *multiresolution analysis* is a nested sequence $\cdots \subseteq V_{-1} \subseteq V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$ of subspaces of $L^2(\mathbb{R})$ (the space of all functions whose squares are integrable on $\mathbb{R}$) with a scaling function ϕ such that:

1) $\cup_{n\in\mathbb{Z}} V_n$ is dense in $L^2(\mathbb{R})$(if any element in a set A can be approximated as closely as we like by an element in a subset B of A),

2) $\cap_{n\in\mathbb{Z}} V_n = \{0\}$,

3) $f(t) \in V_n$ if and only if $f(2^{-n} \cdot t) \in V_0$, and

4) $\{\phi(t-k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis for V_0.

Sometimes, the fourth property makes it impossible for certain scaling functions to form a multiresolution analysis. It can happen, though, that a scaling function just needs to be normalized in order to obtain a multiresolution analysis.

A critical property of each scaling function follows from condition (4). Since $\{\phi(t-k)\}$ is an orthonormal basis for V_0, the set $\{\phi(2t-k)\}$ is an orthogonal basis for V_1. The fact that $\{\phi(2t-k)\}$

is a basis for V_1 means $\phi(t) \in V_0 \subseteq V_1$ can be written in the form $\phi(t) = \sum_k c_k \varphi(2t - k)$ for some constants c_k. This should look familiar as we saw this earlier. This equation is called a *dilation equation* and is crucial in the theory of wavelets. The constants $\{c_k\}$ are the *refinement coefficients*. The fact that a scaling function satisfies a dilation equation is a consequence of the multiresolution analysis. This equation provides enough information that one can proceed without knowing a specific formula for ϕ.

As was already mentioned, when working with a multiresolution analysis we often don't have a simple formula for the scaling function. In these situations, however, we usually do know the refinement coefficients. How does this information enable us to determine some characteristics of the scaling function? One of the standard techniques to deal with problems like this is a numerical one known as the *cascade algorithm.* This algorithm will provide approximations to the scaling function and is an example of a *fixed-point method.*

A *fixed point* of a function f is a value a such that $f(a) = a$. A simple example of a fixed point method follows, which shows how to determine the fixed point of the cosine function. Let $f(t) = \cos(t)$. To find a solution to the equation $\cos(t) = t$, start with a guess, t_0, of a fixed point. Let $t_1 = f(t_0) = \cos(t_0)$, then compute another number $t_2 = f(t_1) = \cos(t_1)$, and continue. In this way, construct a sequence $\{t_0, t_1 = f(t_0), t_2 = f(t_1), \ldots\ldots, t_{n+1} = f(t_n), \ldots\ldots\}$ that will (hopefully) converge to the fixed point of the cosine function. The cascade algorithm is a fixed-point method, except that instead of generating a sequence of numbers, it creates a sequence of functions. When given refinement coefficients, this algorithm creates a sequence of functions $\{f_i\}$ so that, for every value of t, $f_i(t) \rightarrow \phi(t)$ as $i \rightarrow \infty$.

Recall that ϕ satisfies the dilation equation above. If we let F be the function that assigns the expression

$F(\gamma)(t) = \sum_n c_n \gamma(2t - n)$ to any function γ, then we can consider ϕ as a fixed point of F.

We can use a computer algebra system (CAS) such as *Maple* to create a worksheet with loops that will perform the cascade algorithm and then plot the graph of the approximated scaling function. In the appendix, we make use of the dilation equations for the hat wavelets, the quadratic Battle-Lemarie' wavelets, and the cubic Battle-Lemarie' wavelets to approximate the scaling functions for each wavelet group.

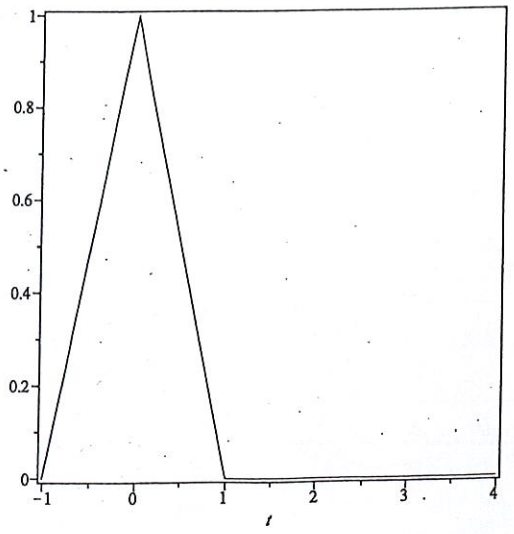
Now that we have a bit of a background in the topic of wavelets, we can see how useful they are. Since the foundation of wavelet theory has been put in place, the field of wavelets has grown quickly over the last decade. Although wavelets are best known for image compression, many researchers now are interested in using wavelets for pattern recognition. In weather forecasting, for example, wavelets could downsize the amount of data in the computer models that are now in use. Also, most of this data is redundant. The barometric pressure where we are is probably about the same as the barometric pressure a few miles away. If the weather models used wavelets, they could view the data the same way weather forescasters do, concentrating on the places where abrupt changes occur such as warm fronts and cold fronts. Wavelets have a future in the movies, as well. Because the wavelet transform is a reversible process, it is just as easy to build an image out of wavelets as it is to break it down into wavelet components. To draw a cartoon character, the animator would only have to specify where a few key points go and then do a reverse multiresolution analysis, making the character look like a real person and not a stick figure. This process could also be used in the video game industry, where they could eliminate the blocky look of today's graphics. What wavelets have done is taken the specific applied research that led to new theories and allowed scientists to look at new applications. Wavelets are an incredibly useful topic and have a very promising future.

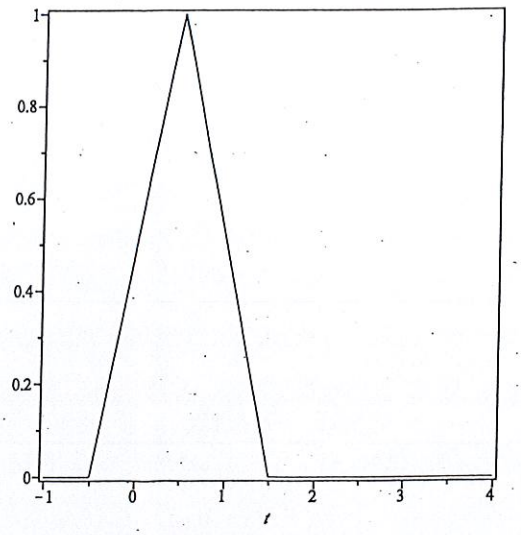
Appendix

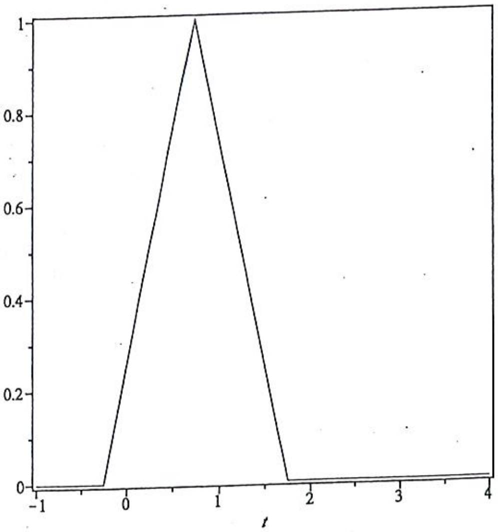
Cascade Algorithm

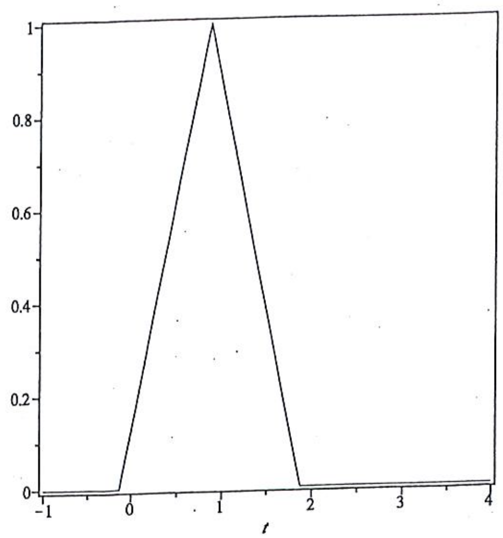
The scaling function for the hat wavelets satisfies the dilation equation

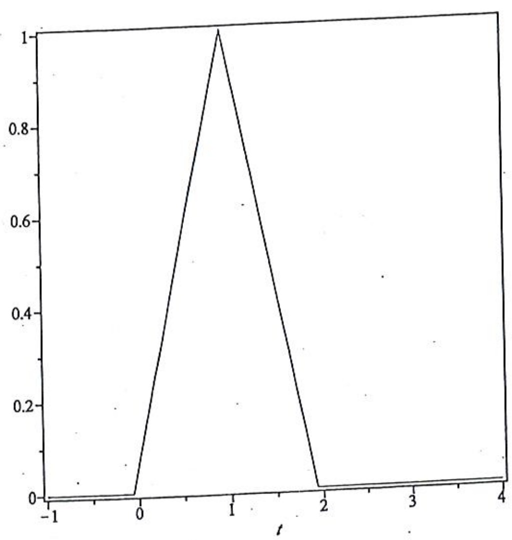
$\phi(t) = \frac{1}{2}\phi(2t) + \phi(2t-1) + \frac{1}{2}\phi(2t-2)$. Using the cascade algorithm, the approximating process looks like this with the finished product being the last graph:

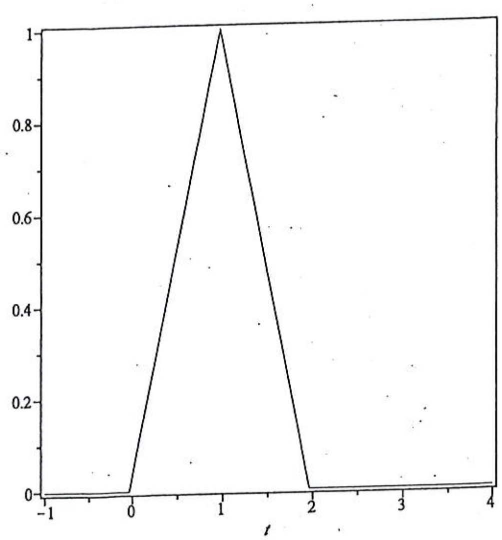


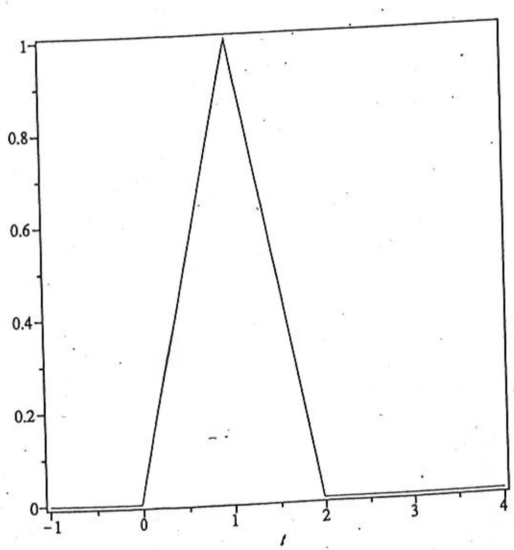


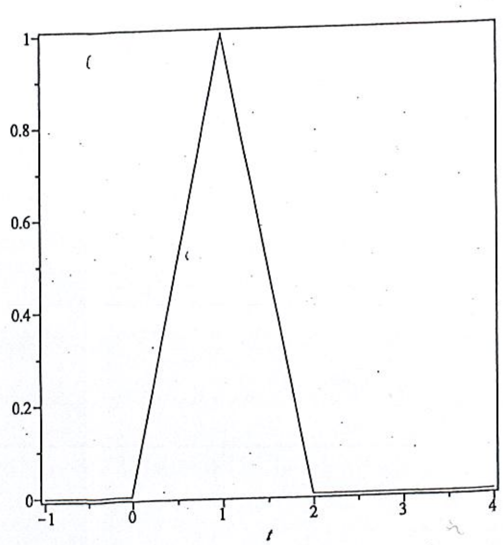

1
0.8
0.6
0.4
0.2
0
−1 0 1 2 3 4
t


1
0.8
0.6
0.4
0.2
0
−1 0 1 2 3 4
t


1
0.8
0.6
0.4
0.2
0
−1 0 1 2 3 4
t


1
0.8
0.6
0.4
0.2
0
−1 0 1 2 3 4
t

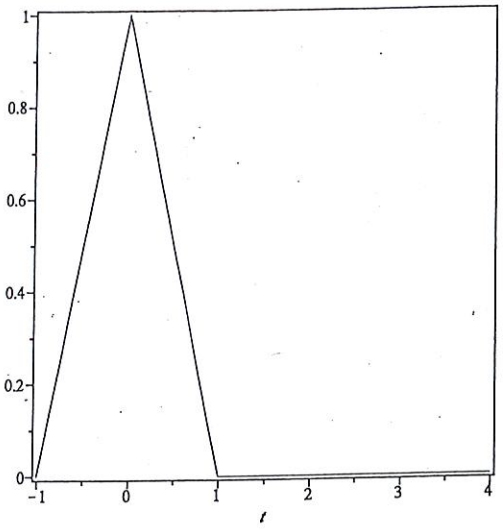

1
0.8
0.6
0.4
0.2
0
−1 0 1 2 3 4
t

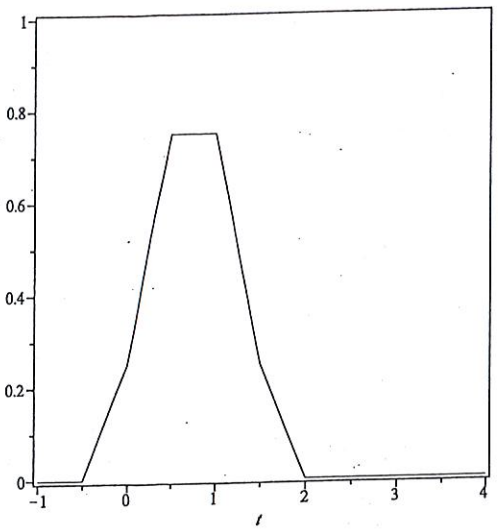

1
0.8
0.6
0.4
0.2
0
−1 0 1 2 3 4
t

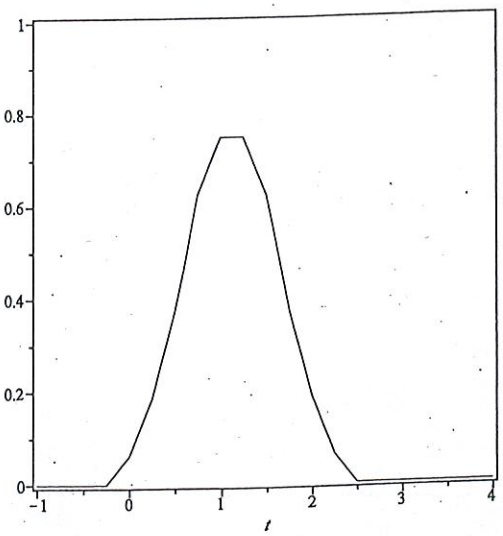
The quadratic Battle-Lemarie' scaling function satisfies the dilation equation

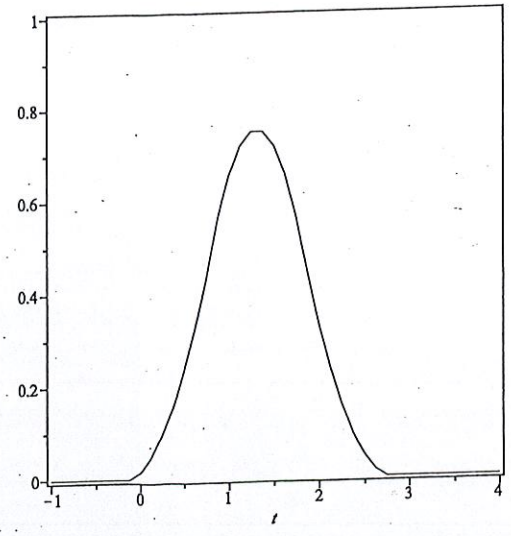
$$\phi(t) = \frac{1}{4}\phi(2t) + \frac{3}{4}\phi(2t-1) + \frac{3}{4}\phi(2t-2) + \frac{1}{4}\phi(2t-3).$$

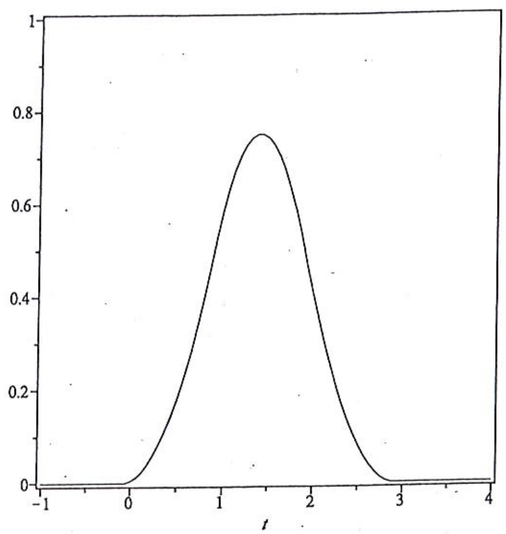
Using the cascade algorithm, the approximating process looks like this with the finished product being the last graph:

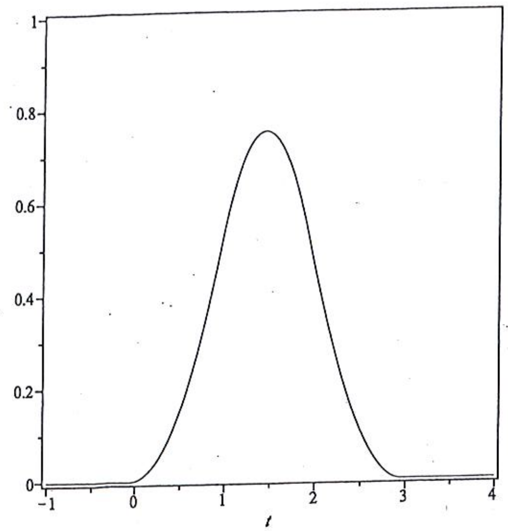


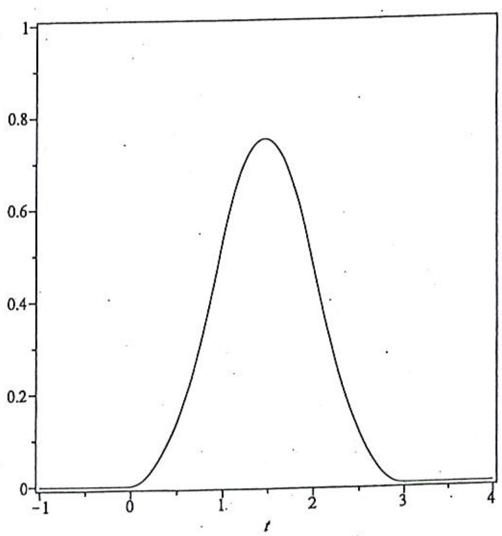


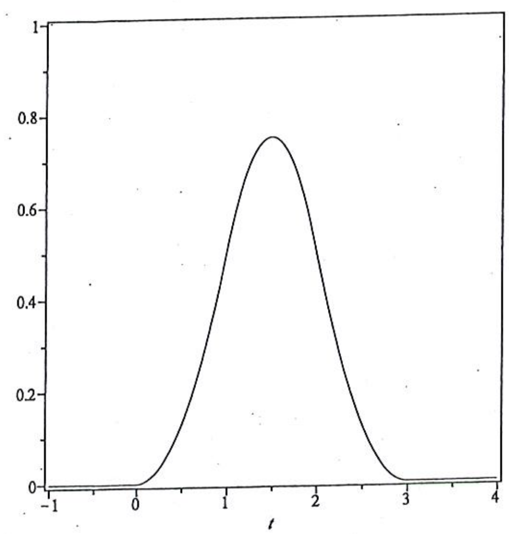


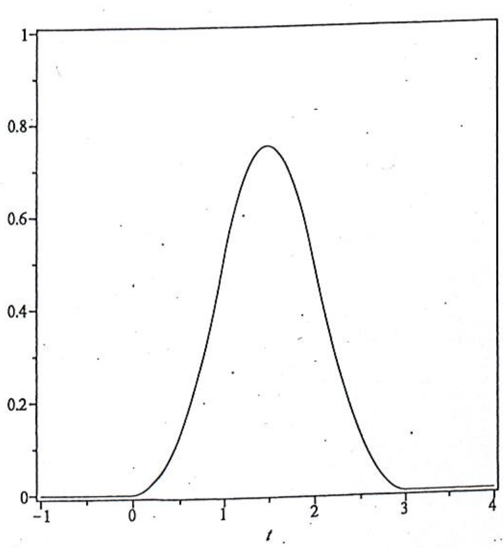



1
0.8
0.6
0.4
0.2
0
−1
0
1
2
3
4
t


1
0.8
0.6
0.4
0.2
0
−1
0
1
2
3
4
t


1
0.8
0.6
0.4
0.2
0
−1
0
1
2
3
4
t

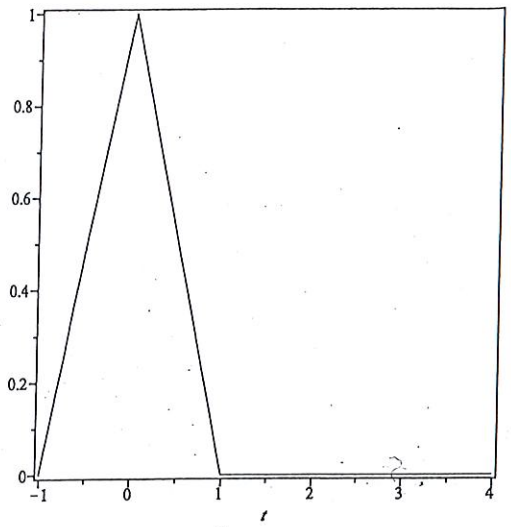

1
0.8
0.6
0.4
0.2
0
−1
0
1
2
3
4
t


1
0.8
0.6
0.4
0.2
0
−1
0
1
2
3
4
t

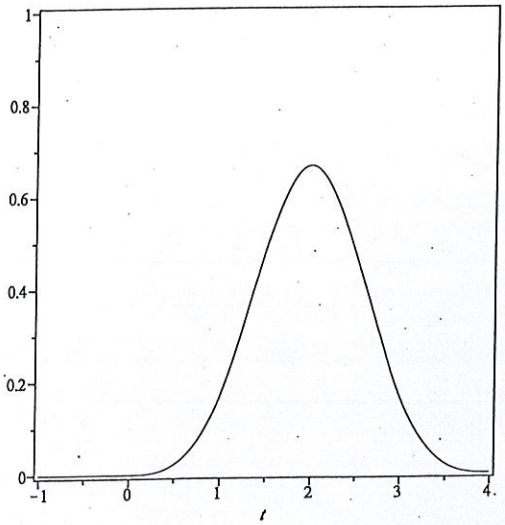
Finally, the cubic Battle-Lemarie' wavelet scaling function satisfies the dilation equation

$$\phi(t) = \frac{1}{8}\phi(2t) + \frac{1}{2}\phi(2t-1) + \frac{3}{4}\phi(2t-2) + \frac{1}{2}\phi(2t-3) + \frac{1}{8}\phi(2t-4)$$

Before performing the cascade algorithm we have a function that looks like:



After using the cascade algorithm, we have approximated a scaling function that looks like:



References

1. Aboudafel, Edward and Steven Schlicker. Discovering Wavelets. New York: John Wiley & Sons, Inc, 1999.

2. Hubbard, Barbara Burke. The World According to Wavelets. Wellesley: A K Peters, Ltd, 1996.

3. Mackenzie, Dana. "Wavelets: Seeing the Forest-and the Trees." National Academy of Sciences. December, 2001 <http://www.beyonddiscovery.org/content/view.article.asp?a=1952>.

4. Mulcahy, Colm. "Plotting and Scheming with Wavelets." Mathematics Magazine. Vol. 69, No. 5, December 1996.

5. Nievergelt, Yves. Wavelets Made Easy. New York: Birkhauser, 1999.

6. Polikar, Robi. "The Wavelet Tutorial Part 1." Fundamental Concepts & an Overview of the Wavelet Theory Second Ed. June, 1996 <http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html>.