The Inverse Discrete Wavelet Transform*

Mark Eastaway

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 2.0†

Abstract

A walkthrough of the inverse discrete wavelet transform and the use of the r_idwt program.

1 Introduction to the Inverse Discrete Wavelet Transform (IDWT)

Once we arrive at our discrete wavelet coefficients, we need a way to reconstruct them back into the original signal (or a modified original signal if we played around with the coefficients). In order to do this, we utilize the process known as the inverse discrete wavelet transform.

Much like the DWT can be explained by using filter bank theory, so can the reconstruction of the IDWT. The process is simply reversed. The DWT coefficients are first upsampled (the approximation and the detail coefficients are handled separately) by placing zeros in between every coefficient, effectively doubling the lengths of each. These are then convolved with the reconstruction scaling filter for approximation coefficients (the reconstruction scaling filter is simply the original scaling filter that has been flipped left to right) and the reconstruction wavelet filter for the detail coefficients. These results are then added together to arrive at the original signal.

Similar to how we made the signal periodic before doing our DWT calculations on it, we must make our dwt coefficients periodic before convolving to obtain the original signal. This is done by simply taking the first N/2-1 coefficients from the DWT coefficients, and appending them to the end. Remember that N is the length of our scaling filter.

After the convolution and addition, to grab the part of the signal we want away from the convolution 'junk', we grab the coefficients from N to the length of the signal + N -1. This will give us our original signal.

NOTE: THIS MAY NOT BE CORRECT. WE BELIEVE THE THEORY IS SOUND BUT WE ARE ARRIVING AT ANOMOLOUS RESULTS AND BELIEVE THE REASON WHY IS DUE TO THIS PERIODIZATION. FURTHER DETAILS WILL BE ADRESSED FURTHER DOWN.

If you are looking for a graphical description of this process, simply look at the figure below:

*Version 1.1: Dec 17, 2008 3:14 pm -0600

†http://creativecommons.org/licenses/by/2.0/

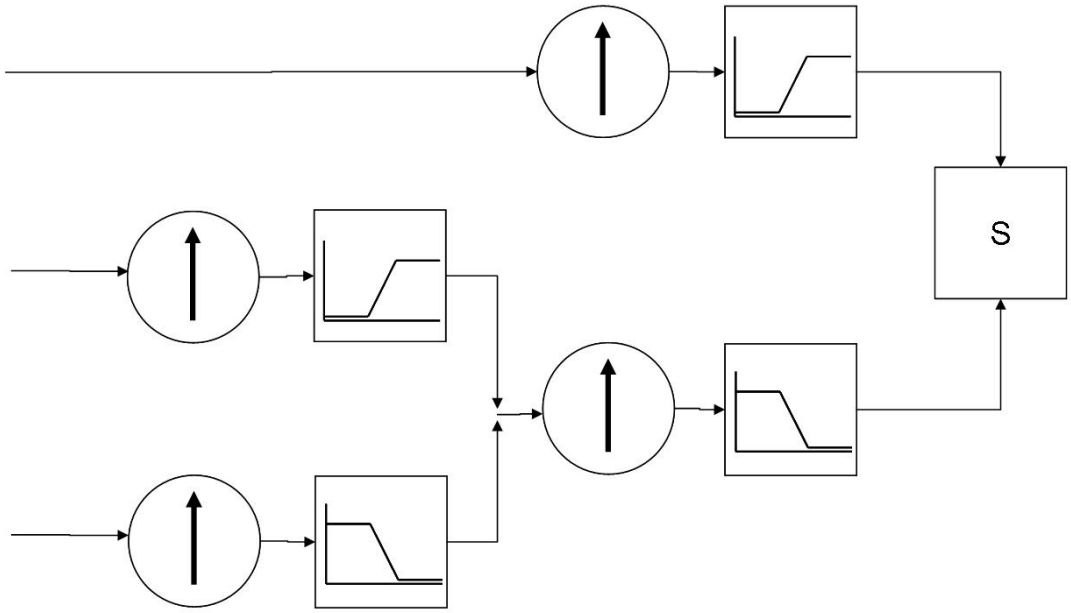

Figure 1

Please keep in mind that the pattern of reconstruction must match the pattern of the DWT's deconstruction, so in our case for multi level reconstruction we simply reconstruct the approximation coefficients in order from the finest scale to the coarsest scale. In the above figure, the lower paths are the approximation coefficients and the higher paths are the detail coefficients; also the junction of two arrowheads corresponds to an addition.

2 How to use our code (r_idwt.m)

x = r_idwt(fx,scaling,wavelet,scales,graphs)

The function is called using the r_idwt function name. The parameter fx is the DWT from which we wish to reconstruct the original signal. The parameter scaling is the hrn output given by our R_daub code, or the reconstructive scaling function of another wavelet function. The parameter wavelet is the wavelet coefficients given by the hr1n output of our R_daub code, or the reconstructive wavelet coefficients of any other wavelet function. The parameter scales is the amount of levels wanted in the IDWT. The parameter graphs is 0 for no graphs, 1 for graphs of the IDWT at each level. The output x is reconstructed original signal.

This code uses a loop to handle the possibility of multiple levels as opposed to recursion used in our DWT code. There are two main reasons for this: The first is that the recursion used in the DWT code was used primarily to show the recursive nature of the DWT at multiple levels. As that has been concretely shown there is no real reason to show it again with the IDWT as the idea is still the same.

Also, as explained above we had been running into errors with our reconstruction and tried to recreate an example code as accurately as possible, and the example code utilized a loop as well.

The code segments we believe contain the errors are presented below:

```
%current level
d  = fx(coeffperlev+1:2*coeffperlev);
%Upsamples the approximation coefficients (inserts 0 in between each
%coefficient) and also makes it period by appending the first N/2-1
%coefficients onto the end of the coefficents.
%****POTENTIAL HERE FOR THE ERROR, ALTHOUGH UNLIKELY*****
upc(1:2:2*coeffperlev+N) = [c c(periodiclength)];
%Peforms the same function for the detail coefficients
upd(1:2:2*coeffperlev+N) = [d d(periodiclength)];
%We convolve our detail coefficients with our wavelet filter and our
%approximation coefficients with our scaling filter and then combine
%them to arrive at the iDWT.  There will be convolution remnants in the
%beginning and end of this result.
%***NOTE THE ERROR ENCOUNTERED AND EXPLAINED IN THE DOCUMENTATION IS
%POTENTIALLY FOUND IN THIS STEP.
c  = conv(upc,h0) + conv(upd,h1);
%We want to isolate the reconstructed signal from all the convolution
%'junk' that surrounds it.  The elements from N (the length of our
%filter) to the double of our coefficients per level will grab the
%correct signal from our convolution result
%***IT IS LIKELY THAT THE TAIL END ERROR ENCOUNTERED COULD BE FROM THIS
%STEP, FROM AN INCORRECT FORMULA OF GRABBING
c  = c(N:N+2*coeffperlev-1);
```

Figure 2

The circled code is the code we believe is causing the error with reconstruction, with the shade of the circle noting our belief of which code is responsible (the darker the circle the more strongly we believe it is the erroneous code). Of course, it could potentially be all three pieces, our theoretical model, or something else entirely.

Examples (+Errors)

Let's go over some examples. We will use our DWT result from our last module (r_dwt).

`recon = r_idwt(dwt,hr0,hr1,1,0);`

This will reconstruct our DWT coefficients into our original signal. See below for the actual results:

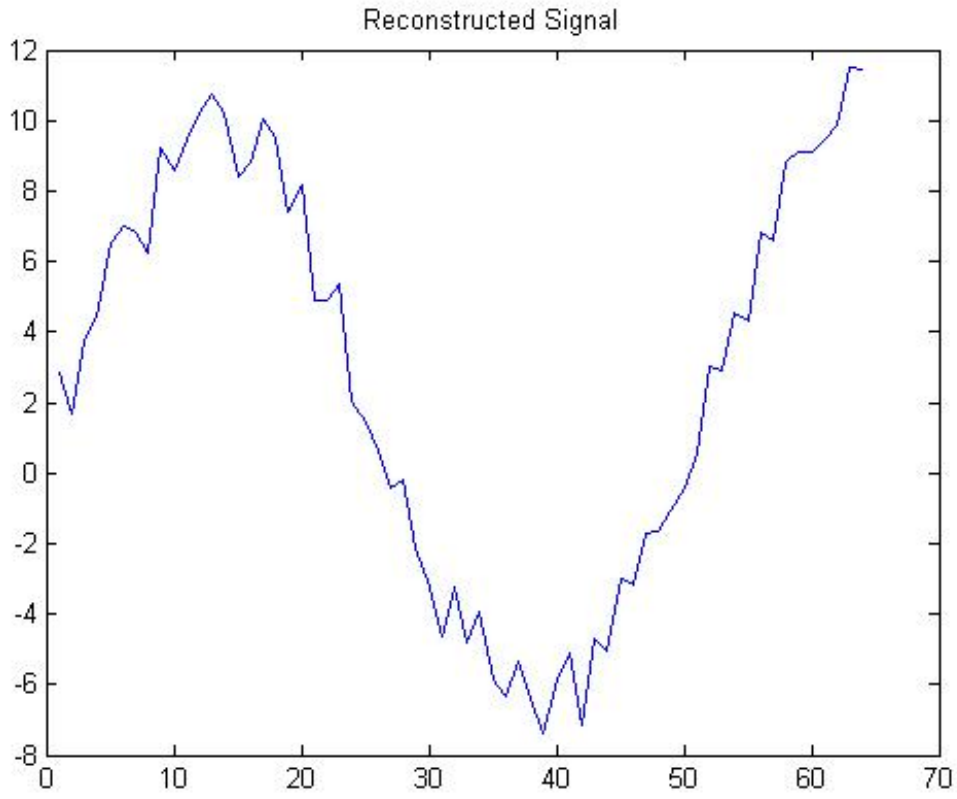

Figure 3

This looks fairly accurate! It appears that we have perfectly reconstructed our original signal, as shown below:

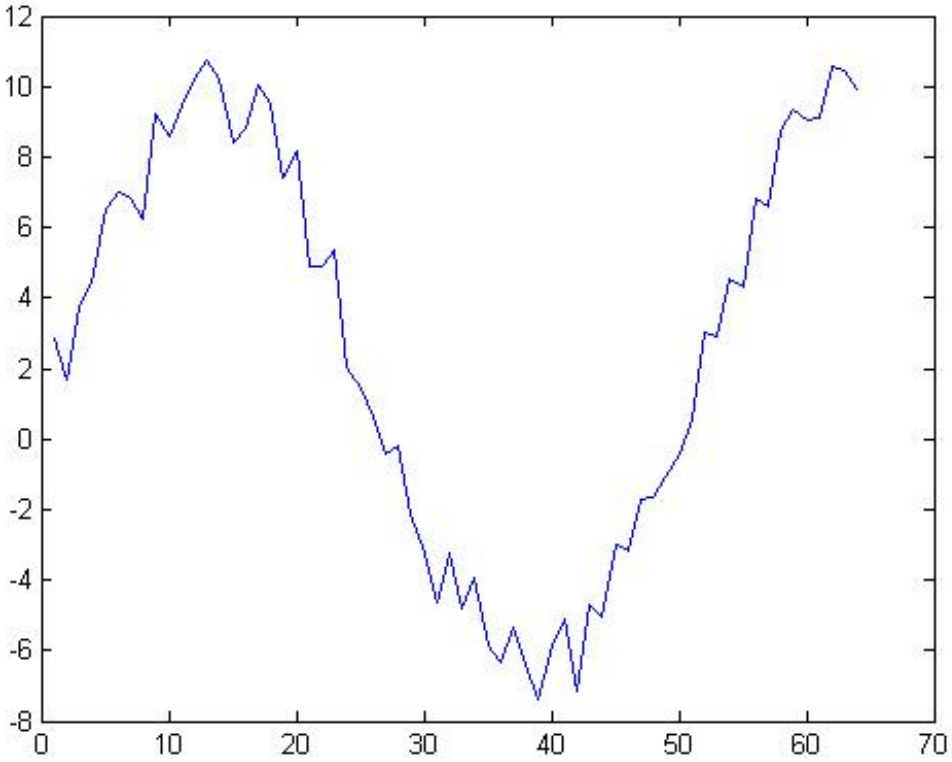

Figure 4

Oh no! Look at the last couple of coefficients (actually, the last 8...which corresponds to the length of our filters, hrmmmm)...we can see they are off from our original signal now:

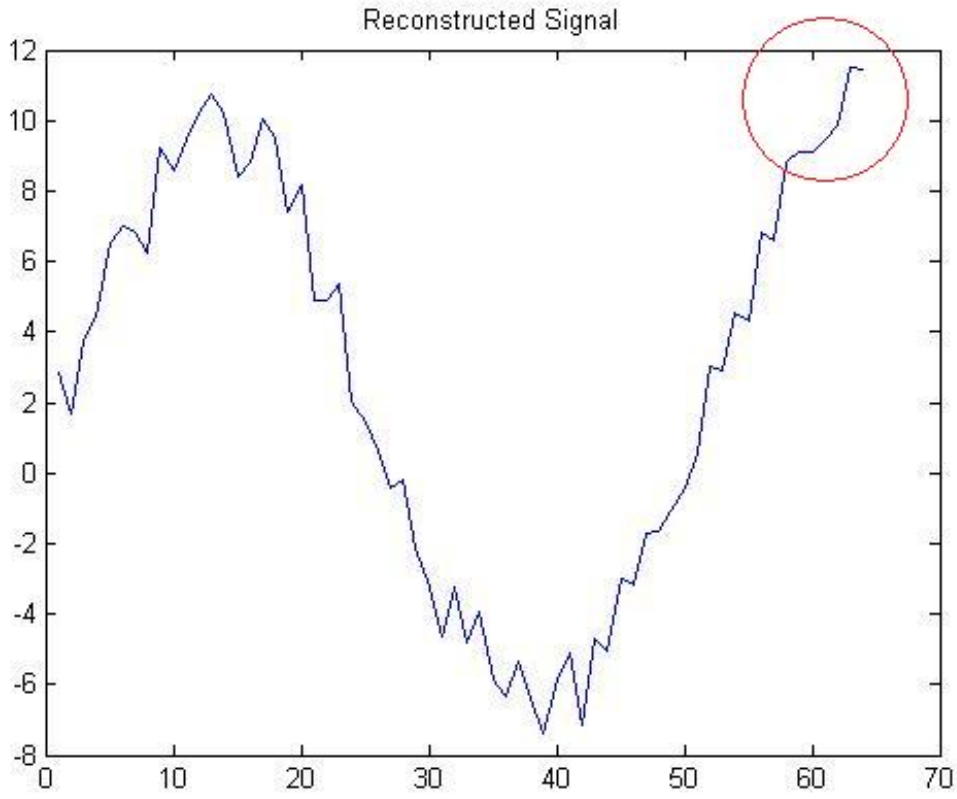

Figure 5

For an even better example, we'll look at a different signal:

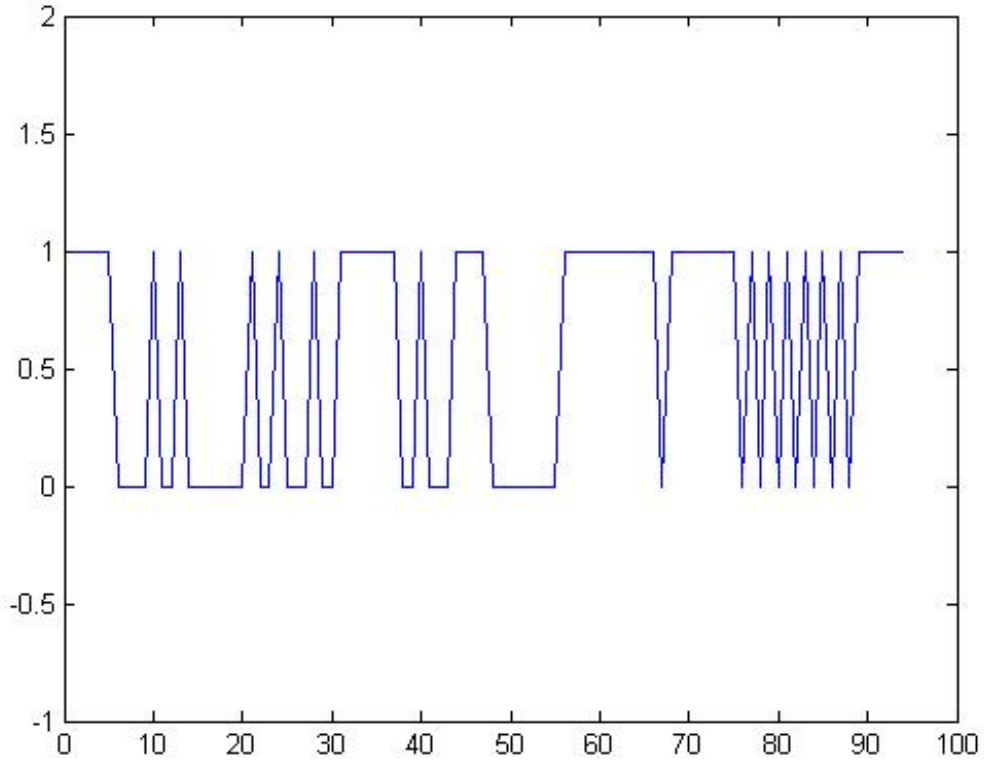

Figure 6

And its reconstruction:

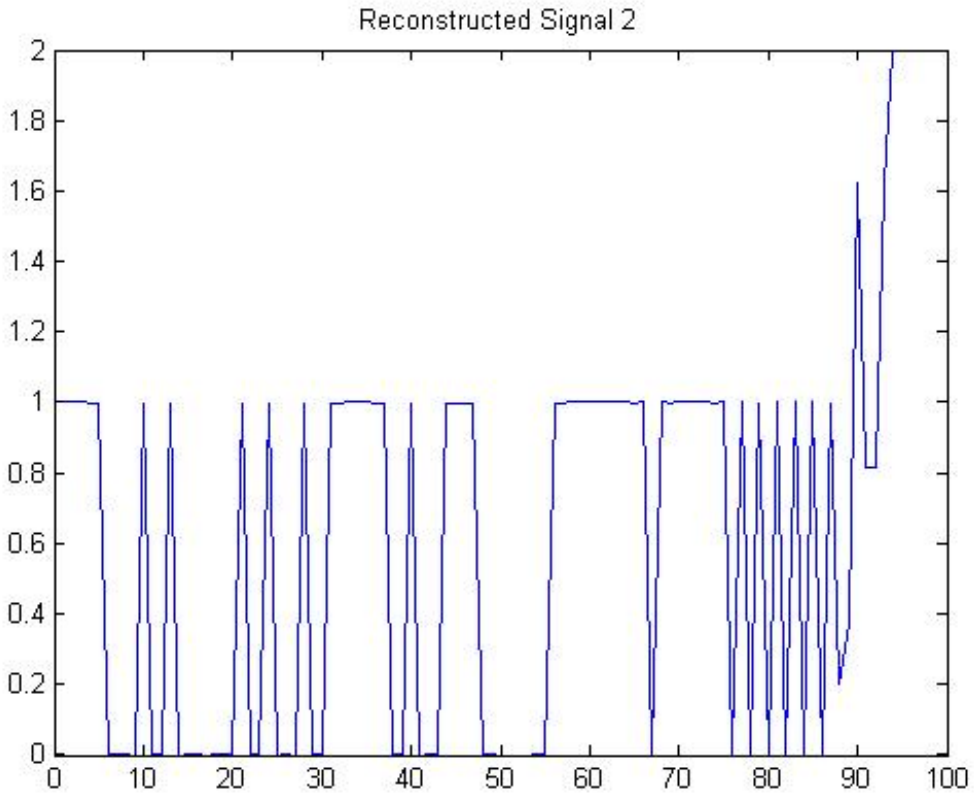

Figure 7

There is clearly something erroneous happening somewhere in our iDWT.
The problem is magnified when we do our second level iDWT (to our original noisy sinusoid signal):

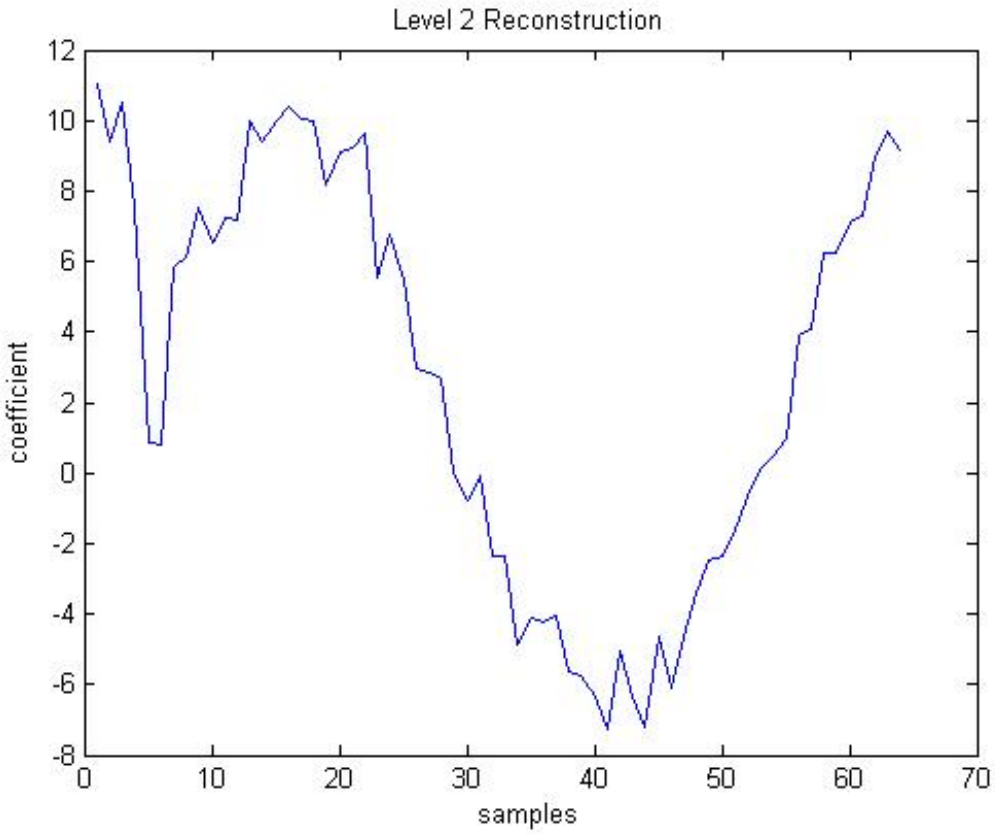

Figure 8

Now not only is it not as accurate, there are mistakes in the first 8 coefficients as well! This stems from our inaccurate results in our first IDWT (the end effects), and these are all convolution errors.