Wavelet Shrinkage and Denoising

Instructions: Email the requested figure as a .fig or .tif file to tleise@amherst.edu.

In this lab we will work through further important results by David Donoho and collaborators that demonstrate how to effectively apply wavelet analysis to noisy data, building on what we did in a previous lab.¹

As in the previous lab, suppose you are measuring some process by sampling a value once a minute, resulting in a vector \( y \) of measured values. We want to find \( v \), the “true” signal, under the assumption of additive noise: \( y = v + e \). The simplest approach is to assume the noise vector \( e \) is Gaussian white noise: each component is independently drawn from a normal distribution (bell-shaped curve) with mean zero and some unknown variance \( \sigma^2 \).

We apply a wavelet transform to generate approximation \( (a_{j,k}) \) and detail \( (b_{j,k}) \) coefficients. Presumably, most of the “true” signal \( v \) goes into the \( a_{j,k} \), while the noise vector \( e \) gets separated into the \( b_{j,k} \). Iterating the wavelet transform several times should further filter out noise, removing it from the approximation into the detail coefficients. One strategy to denoising a signal is to “shrink” the values of the coefficients toward zero using a shrinkage function \( s_\lambda(x) \):

\[
 s_\lambda(x) = \begin{cases} 
 x - \lambda & \text{if } x > \lambda, \\
 0 & \text{if } -\lambda \leq x \leq \lambda, \\
 x + \lambda & \text{if } x < \lambda, 
\end{cases}
\]

The inverse transform is then applied to the altered coefficients to obtain the denoised signal.

**Exercise 1** Create signal and noise vectors as in the previous lab. Also create a shrinkage function:

```matlab
function s=shrinkage(x,lambda)
s=(abs(x)>=lambda).*sign(x).*(abs(x)-lambda);
```

Test your function to make sure it works correctly.

**Exercise 2** We want to iterate the D4 transform, so create a Matlab function to do each iteration:

```matlab
function [aj,bj]=WT1D(a,p)
% a is an array of (j+1)-level approximation coefficients
% p is an array of scaling coefficients
% Returns j-level approximation and detail coefficients

L=length(p);p=p(:); % ensures is column vector
N=length(a);a=a(:)';% ensures is row vector
h=flipud(p); % flips upside down to obtain scaling filter
g=(-1).^(1:L)'.*(1:L).^.*p; % wavelet filter

a=[a a(1:L-2)]; % periodic boundary condition
for k=1:N/2
    idx=2*k-1;
c(k,:)=a(idx:idx+L-1); % part of signal that the filter hits to generate each coefficient
end
```

¹Adapted from Example 4.2 on pages 316-319 of the Ruch & Van Fleet book *Wavelet Theory.*
aj=c*h;
bj=c*g;

Apply 3 iterations of WT1D to the noisy heavisine signal using the D4 wavelet:

\[ p = \left[ 1+\sqrt{3}; 3+\sqrt{3}; 3-\sqrt{3}; 1-\sqrt{3} \right]/\sqrt{2} \]

**Exercise 3** Save the detail coefficients in an array \texttt{details=[b8; b9; b10]}. Apply the shrinkage function to the details vector, using a \texttt{lambda} value of your choice, e.g., 0.5. Compare the original and shrunk detail vectors by plotting on the same figure (don’t need to submit).

**Exercise 4** Now we want to invert the transform using the new detail coefficients and the unchanged \texttt{a8}, so we need an inversion function.

```matlab
function a=IWT1D(aj,bj,p)
% aj and bj are arrays of j-level coefficients, p is an array of scaling coefficients
% Returns array a of (j+1)-level approximation coefficients
L=length(p);p=p(:);
N=length(aj);
aj=aj(:)';aj=[aj(N-L/2+2:N) aj];
bj=bj(:)';bj=[bj(N-L/2+2:N) bj];

h=flipud(p); % scaling filter
g=(-1).^(1:L)'.*p; % wavelet filter

c=zeros(N,L/2);d=zeros(N,L/2);
for k=1:N
c(k,:) = aj(k:k+L/2-1);
d(k,:) = bj(k:k+L/2-1);
end

a=zeros(2*N,1);
a(1:2:end)=c*flipud(h(1:2:L))+d*flipud(g(1:2:L));
a(2:2:end)=c*flipud(h(2:2:L))+d*flipud(g(2:2:L));
```

**Exercise 5** What is the best choice for \( \lambda \) in this wavelet shrinkage algorithm? In a set of classic papers, Donoho and Johnstone showed that a good choice is often the \textit{universal threshold} \( \lambda^{\text{univ}} \):

\[
\lambda^{\text{univ}} = \hat{\sigma} \sqrt{2 \ln(M)},
\]

where \( M \) is the length of the \texttt{details} vector and \( \hat{\sigma} \) is an estimate of the noise level, as we calculated using the median absolute deviation in the previous lab: \( \hat{\sigma} = \text{MAD(details)}/0.6745 \).

Calculate \( \lambda^{\text{univ}} \) for the original \texttt{details} vector, apply the shrinkage function with that value for \texttt{lambda}, then invert the iterated transform to obtain a denoised signal \( w \).

Submit a labeled 3-subplot figure showing \( y \), \( v \), and \( w \). Indicate in your email what values you found for \( \hat{\sigma} \) and \( \lambda^{\text{univ}} \).