Wavelet Shrinkage and Denoising

Instructions: Email the requested figure to tleise@amherst.edu by next Wed.

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. These exercises are adapted from Example 4.2 on pages 316-319 of *Wavelet Theory* by Ruch & Van Fleet .

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

We apply a wavelet transform to generate approximation ($a_{j,k}$) and detail ($b_{j,k}$) coefficients. Presumably, most of the "true" signal $\mathbf{v}$ goes into the $a_{j,k}$, while the noise vector $\mathbf{e}$ 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* $s_\lambda(x)$:

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

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

Exercise 1 The "heavisine" function was introduced for testing noise-reduction methods (a combination of a sine function and two Heaviside functions causing jump discontinuities). We'll use it in this lab to see how well the Haar transform does at extracting $\mathbf{v}$ from $\mathbf{y}$:

```
t=(0:2^11-1)'/2^11;
v=4*sin(4*pi*t)-sign(t-0.3)-sign(0.72-t);
e=0.5*randn(size(t));   % sigma=0.5
y=v+e;
figure;plot(t,y,'k.')
```

Also create a script for the shrinkage function:

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

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

```
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 p is a column vector
```

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

a=[a a(1:L-2)]; % repeat values so don't run off edge of vector
c=zeros(N/2,L);
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

aj=c*h;
bj=c*g;
```

Apply 4 iterations of `WT1D` to the noisy heavisine signal using the D4 wavelet:

```
p=[1+sqrt(3);3+sqrt(3);3-sqrt(3);1-sqrt(3)]/4/sqrt(2)
```

Exercise 3 Save the detail coefficients in an array `details=[b7; b8; b9; b10]`. Apply the shrinkage function to the `details` vector, using a `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 "shrunk" detail vectors and the unchanged `a7`, so we need an inversion function.

```
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 λ in this wavelet shrinkage algorithm? In a set of classic papers, Donoho and Johnstone showed that a good choice is often the *universal threshold* λ^{univ}:

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

where M is the length of the `details` vector and $\hat{\sigma}$ is an estimate of the noise level using the median absolute deviation:

```
MAD=median(abs(details-median(details)))
```

$\hat{\sigma}$=`MAD`/0.6745

Calculate λ^{univ} for the original `details` vector, apply the shrinkage function with that value for `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 λ^{univ}.