Sheet 11

Create the directory sheet11 and make it your working directory.

Example 1. a) Go to my homepage and save the file data13.csv to your working directory. The first row is the time data and the second row is the corresponding signal data. Load the data (see sheet 2, Example 1 if you don't remember how) letting `t` be the first row and `y` the second. W are going to use wavelets to filter the noise. To get an understanding of how the details D_k are the various spikes that get wider and wider as k gets bigger and bigger, we plot the signal and the detail D_k on the same axes. First we do it for Haar wavelets. Enter the following—it will be a model you can easily modify for subsequent problems. First choose the level to which you wish to decompose:

```
n = 1;
```

Enter the wavelet family to use (db1 in this case):

```
fam = ‘db1’;
```

Decompose the signal with the Discrete Wavelet Transform (DWT):

```
[c,L] = wavedec(y,n,fam);
```

Now have Matlab compute the detail D_n:

```
Dn = wrcoef(’d’,c,L,fam,n);
```

Plot D_n with the signal

```
hold on
plot(t,y,’k’,’linewidth’, 1)
plot(t,Dn,’r’,’linewidth’,1)
m = sprintf(’ D_% d’,n);
title([fam m],’fontsize’,16,’fontweight’,’bold’)
hold off
```

Change n and try to see which value gives wiggles that are just wider than the noise in the signal.

b) Let's you decided $n = 5$. Then that means to filter the signal, you want to throw out the details D_1 through D_5 and use the approximation A_5 as the filtered signal. We throw out the details you decided in part a):

```
An = wrcoef(’a’,c,L,fam,n);
```

Now plot the signal and the filtered signal on the same graph so that it's easy to see what's going on:

```
hold on
plot(t,y,‘k’,‘linewidth’,.5)
plot(t,An,‘r’,‘linewidth’,2)
legend(’Signal’,’Filtered Signal’)
hold off
```

Change the value of n until you get the best looking plot. Did it match with the choice you made in part a)?

c) Repeat all this using the family `db3`—you should see that this family does a great job of filtering the signal.

Example 2. Repeat this analysis for the data in the file data14.csv. This time start with the family `db2`.

Example 3. Recall the data in data17.csv from example 2 on Sheet 7 had a spike—there you filtered it out using the DFT. Now that you understand what is going on, let us cut to the chase and filter the signal without plotting the details on the same graph as in the previous examples. Thus this time just plot the approximation `An` and the signal on the same graph. I suggest you use `linewidth` of 1 in each plot.

Play around with the level n until your filtered signal looks good. You might want to zoom in on the spike on the plot to see if it is really gone—remember you can do this using the `axis` command.

Example 4. Go to my homepage and download the file noisydata1.csv. It is a file containing a noisy signal. Proceed as you did in the preceeding example. Try using different wavelet families until you get a good plot, starting with `db2`.

Example 5. In the last example, the noisy signal `y` was generated from the signal with no noise in the file no-noise-data1.csv. Load it and define the second row as the variable `y1`. Plot the filtered signal on the same graph as the signal without noise so you can see how well the filtering worked with the particular choice of `dbm` and the level n you made above. Also include the relative error:

```
fprintf(‘Relative Error is %f\n ’,norm(An-y1)/norm(y1))
```

Try different levels.

Example 6. Go to my homepage and save the data file noisydata3.csv of a noisy signal to your working directory. Repeat the analysis of the preceeding example to filter out the noise. Then see how well you did by comparing to the true signal found in no-noise-data3.csv.