Introduction to the Haar transform

In this lab we will apply the coefficient formulas connecting scales j and $j+1$ to some examples to explore how the process of changing the resolution works. To explore how the discrete Haar transform works, we will start with a relatively high level of resolution and then see what happens as we decrease the level j and decompose the function into approximations and details.

Instructions: Email the requested figures as .fig or .tif files to tleise@amherst.edu by Wed Oct 28.

Suppose we sample a function f with support in [0,1) at points $t = k/2^J$, $k = 0, 1, \cdots, 2^J - 1$, for some positive integer J, and then record this data as a vector a_J.

The coefficients for approximation vectors a_j and detail vectors d_j are related by the following formulas:

$$a_j(k) = \frac{1}{\sqrt{2}}\Big(a_{j+1}(2k+1) + a_{j+1}(2k)\Big) \tag{1}$$

$$d_j(k) = \frac{1}{\sqrt{2}}\Big(a_{j+1}(2k+1) - a_{j+1}(2k)\Big) \tag{2}$$

That is, add consecutive pairs of numbers to generate the approximation vectors a_j, and take differences (odd-indexed number minus the even-indexed number) to generate the detail vectors d_j.

Exercise 1 Start by considering $a_3 = \{2, 1, 0, 1, 0, 0, 1, 3\}$ (indexing runs from $k = 0$ to 7). Use equations (1) and (2) to calculate by hand the sequences $a_2 = \{a_2(0), a_2(1), a_2(2), a_2(3)\}$ and $d_2 = \{d_2(0), d_2(1), d_2(2), d_2(3)\}$ from a_3. Next calculate $a_1 = \{a_1(0), a_1(1)\}$ and $d_1 = \{d_1(0), d_1(1)\}$ from a_2, then $a_0 = \{a_0(0)\}$ and $d_0 = \{d_0(0)\}$ from a_1. (You don't need to turn in these calculations.)

Exercise 2 Create a Matlab script that runs through this process for the sequence a_9 as follows:

```
J=9; t=(0:2^J-1)'/2^J;
a9=20*t.^2.*(1-t).^4.*cos(12*pi*t)+0.1*randn(size(t));
figure; stairs(t,a9)
```

You can apply the formulas commands like (1) and (2) through for-loops, or try more succinct array-based approaches like the following:

```
a8=(a9(1:2:end)+a9(2:2:end))/sqrt(2); % adds even-indexed to odd-indexed entries
```

Note that the theoretical expressions start indexing at $k = 0$, while Matlab arrays start indexing at 1. The above code takes the array `[a9(1) a9(3) a9(5) ...]` (representing $\{a_9(0), a_9(2), a_9(4), \ldots\}$, that is, the $a_9(2k)$ coefficients) and adds it to the array `[a9(2) a9(4) a9(6) ...]` (representing $\{a_9(1), a_9(3), a_9(5), \ldots\}$, that is, the $a_9(2k+1)$ coefficients). Be sure to ask me if it's not clear what the given code is doing or if you want help coding your ideas into Matlab.

Create a 3-by-2 figure with subplots showing a_8, d_8, a_7, d_7, a_6, and d_6, with identical y axis ranges to allow direct comparison. Clearly label each subplot to indicate which array is being plotted. I suggest using `stem` to plot the coefficients.

Exercise 3 The "energy" of a sequence is the sum of the squared coefficients. Find the energy in a_9 (something like `sum(a9.^2)`). Compare this to the energy of a_8 plus the energy of d_8–the sum should equal the energy of a_9. This conservation of energy as we decompose the function is another nice property resulting from using an orthonormal basis (and justifying all those pesky $1/\sqrt{2}$ factors). What do we have to add to the energy of a_7 to equal the energy of a_9 (in terms of the energy of other sequences)? What do we have to add to the energy of a_6 to equal the energy of a_9? Put your answer in the text of your email.

Exercise 4 Repeat Exercise 2 on the sequence

```
a9=(t<1/3)+(t>=411/512);
```

Submit the analogous figure to that in Exercise 2. What are the $d_j(k)$ coefficients revealing in this example? Put your answer in the title of your figure.