The MATLAB functions that may be useful for this assignment:

- **dbwavf**: Design of Daubechies’ CQF filters
- **dwt**: Fast (Discrete) wavelet transform (i.e., decomposition)
- **idwt**: Inverse wavelet transform (i.e., reconstruction)
- **fft**: (Fast) Fourier transform
- **ffts**: Rearranging the output of **fft**

The description of the above functions is available at MATLAB Helpdesk, file:///s/matlab/help/helpdesk.html

Comments about **fft**: Given a signal \( x \) with \( N \) samples, the **fft** computes the values of \( X \) at \( N \) points equally spaced on the interval \([0, 2\pi]\). Thus, if your signal \( x \) consists of, say, only 100 samples, but you would like to get values of \( X \) at, say, 500 points, you should simply extend \( x \) by adding 400 zeros at its end (the command \( x(500)=0 \) should do it, right?)

Another comment concerns the plotting of \( X \): Since \( X \) is usually complex-valued, and since MATLAB has its own rules concerning the plotting of such graphs, you are advised, as the cheapest solution, to plot \(|X|\).

1. Using **dbwavf**, create the Daubechies’ filters \( d_1, d_2, d_4, d_7 \) (Comment: in terms of the algorithm presented in class, the \( d_k \) filter is the one obtained by expanding

\[
(\cos^2(\omega/2) + \sin^2(\omega/2))^{2k-1}.
\]

The filter itself has \( 2k \) non-zero coefficients. Some people enumerate Daubechies’ filters according to the number of non-zero coefficients in the filter). Verify that the above four filters are all CQFs. (You may want to write a MATLAB function **CheckIfCQF** analogous to the function **CheckIfTight** from your previous assignment that accepts one input vector \( h \) and returns **true** if it is CQF.
and false otherwise.) Using `fft` and `fftshift`, plot the magnitudes of the masks $D_1$, $D_2$, $D_4$, $D_7$.

2. Compute 4 levels of the discrete wavelet transform using the function `dwt`, the Daubechies filter of length 4 (i.e., $d_2$), and the input signal

$$x(n) = \begin{cases} n - 16 & \text{if } 16 \leq n \leq 272 \\ 528 - n & \text{if } 272 \leq n \leq 528 \\ 0 & \text{otherwise} \end{cases}$$

of length 544. Plot all coarse approximations ($x_{j,0}$ in class’ notation) and details ($x_{j,1}$, respectively). Describe the results. Now use `idwt` to compute the reconstructed signal $y$. Plot $y$ and explain any difference between $x$ and $y$.

Repeat the same process for the Daubechies filter of length 16.

3. Write a MATLAB function `dframet`

```matlab
function [ca, cd1, cd2] = dframet(x, h0, h1, h2)
```

that accepts an input signal $x$, a low-pass filter $h_0$, and two high-pass filters $h_1$, $h_2$ and performs a single-level decomposition of the signal, obtaining the refinable function coefficient vector $ca$ and two detail coefficient vectors $cd1$ and $cd2$ (i.e., the three requisite signals are obtained by convolving $x$ with each of the given filters and then downsampling).

Apply this function to compute 2 levels of the discrete “frame transform” of the signal in problem 2 with the filters

$$h_0 = \left( \frac{1}{4}, \frac{2}{4}, \frac{1}{4} \right), \quad h_1 = \left( \frac{\sqrt{2}}{4}, 0, -\frac{\sqrt{2}}{4} \right), \quad h_2 = \left( -\frac{1}{4}, \frac{2}{4}, -\frac{1}{4} \right).$$

Turn in the code you wrote, and the plots of the various signals.

Do you recognize the filters?

4. Write a MATLAB function `MyCascade`

```matlab
function [phi_i, xval] = MyCascade(h, I)
```

that accepts the filter $h$ and the number of iterations $I$, and returns the values $\phi_I$ of the $I$-th approximation to the refinable function at the $2^{-I}$-integer points of the mesh $xval$ starting with the signal $\phi(0)$ (see below). The left end point of the $xval$ is zero, the right endpoint is the first $2^{-I}$-integer where $\phi_I$ vanishes.

Here, the initial function is the centered hat function whose values at the integers are

$$\phi(0)(n) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{otherwise}. \end{cases}$$

To recall, $\phi_{i+1}(t) = \sum_n 2h(n)\phi_i(2t - n)$.

For example, `[phi_i, xval] = MyCascade([0.1 0.9], 1)` should return
phi =
0   0.2000  1.8000  0
xval =
0   0.5000  1.0000  1.5000

Apply your function to the filters
\[ h = \left( \frac{-1, 3, 3, -1}{4} \right), \quad f = \left( \frac{1, 3, 3, 1}{8} \right). \]

Choose the number of iterations large enough to make conclusions about the resulting refinable functions.

Then, test your code against Daubechies’ filters ‘db3’ and ‘db9’ (again, tune the number of iterations so that you obtain a good resolution of the refinable function).

Turn in your code, the plots, and any (visual) conclusions about the four refinable functions involved, and about the convergence of the cascade algorithm for these cases.