But, that's just for the 1D case. Since we apply it to the rows and the columns, we have the 2D functions:

$$\begin{aligned}\phi\phi(x,y) &= \phi(x)\phi(y)\\ \psi\phi(x,y) &= \psi(x)\phi(y)\\ \phi\psi(x,y) &= \phi(x)\psi(y)\\ \psi\psi(x,y) &= \psi(x)\psi(y)\end{aligned}$$

These correspond to the four quadrants of an image: A, V, D, and H.

The long and the short of it is that we have four samples (let's say):

$$w = hs$$

w are the approximations of the wavelet coefficients, h some magic $N \times N$ matrix that represents a linear system, and s our samples.

$$\begin{bmatrix} w_A \\ w_H \\ w_V \\ w_D \end{bmatrix} = \begin{bmatrix} \phi\phi(x_{s_1}, y_{s_1}) & \phi\phi(x_{s_2}, y_{s_2}) & \phi\phi(x_{s_3}, y_{s_3}) & \phi\phi(x_{s_4}, y_{s_4}) \\ \phi\psi(x_{s_1}, y_{s_1}) & \phi\psi(x_{s_2}, y_{s_2}) & \phi\psi(x_{s_3}, y_{s_3}) & \phi\psi(x_{s_4}, y_{s_4}) \\ \psi\phi(x_{s_1}, y_{s_1}) & \psi\phi(x_{s_2}, y_{s_2}) & \psi\phi(x_{s_3}, y_{s_3}) & \psi\phi(x_{s_4}, y_{s_4}) \\ \psi\psi(x_{s_1}, y_{s_1}) & \psi\psi(x_{s_2}, y_{s_2}) & \psi\psi(x_{s_3}, y_{s_3}) & \psi\phi(x_{s_4}, y_{s_4}) \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} \tag{3}$$

This is just saying that we know that the wavelet coefficients are just scaled properly and include those terms.

The calling of ϕ and ψ are simply putting Figure 1 into a mathematical package.

2.2 The approximation algorithm

Yes, we know how to make the matrix: it's just full of ϕ and ψ in the approbate places. j is determined by the level of wavelet and k, l are determined by where we are in the square (we have to look at how many shifts are possible and determine if x, y is in that.

But the algorithm is a different story – we have to write down **how** to tell what j and k are.

2.2.1 j – the power variable

This is easy. We know how far we are away from the top left corner and what power of wavelet we're going for.

2.2.2 k – the shift variable

This is slightly more complex. This involves both the power (how many shifts are possible?) and the (x, y) position. But we know how far we are away from the upper left corner of the current quadrant (either in the x or y directions). We know that the number of shifts increases by one for each shift in position, making sense since the power is halfed each time.

2.3 Which indices are important?

Initially, I thought that the wavelet transform had to be recursive. That makes it really tricky for the indices – you would have to make `dwt_ind` functions, and keep doing it.

But after I talked to Ashkay, I learned that you can do the full wavelet transform on each row then each wavelet transform on each column. That makes this function trivial: it's just a matter of indexing.

2.4 The reconstruction

In the wavelet domain, we have a tree that corresponds to the image. An example is in Figure 2. The upper levels of the tree represent the lower frequency terms. Since there's no need to closely sample a low frequency term, we only sample where there are high frequencies.

We know that if any of these "branches" are close enough to zero ($|x| < \lambda$) that all of it's child branches are close enough to 0 as well. Therefore, we only look at where the branches are not close enough to zero ($|x| > \lambda$). As we go further down the branch, we form a better approximation of the wavelet transform.

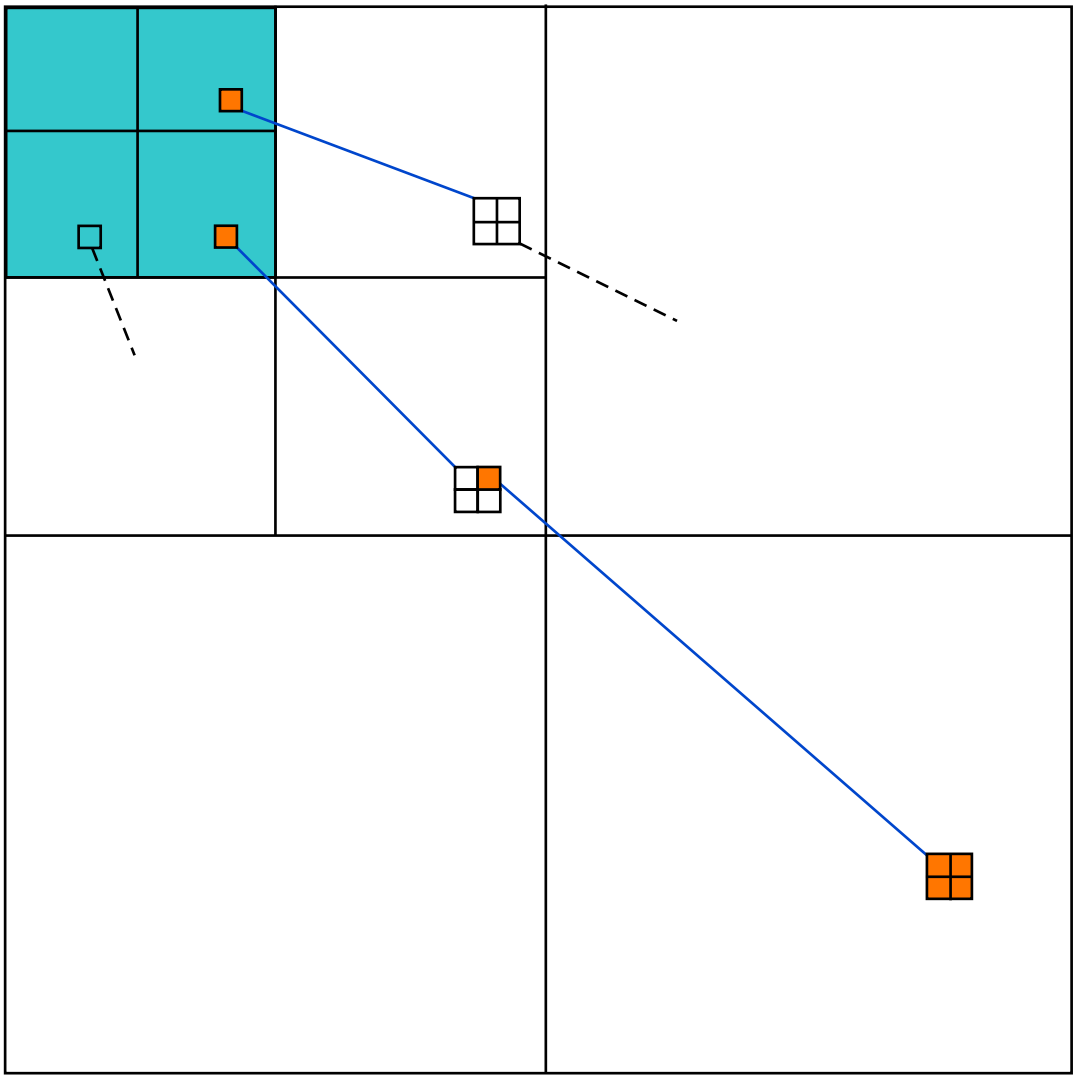

Figure 2: A visualization of our tree. The blue square is our current node. Only two of it's components are non-zero, meaning that there's more detail at lower levels we need to look at closer. We then build a better and better approximation as we need too.

Where we choose to look (or where we sample) is near the edges. The wavelet transform is zero

for a constant: it has no high frequency terms.