# Lecture 28

- Wavelets.

Just like the FFT, the wavelet transform is an operation that can be performed in a fast way. Operating on an “input vector” representing a sampled signal, it can be viewed, just like the FFT, as a matrix operation that “rotates” the data into a new vector. The operation has a well defined inverse.

The FFT expands the signal in a basis of sines and cosines. The wavelet transform uses a different basis of “mother” functions (also known as “wavelets”) that has different properties than sines and cosines.

Sines and cosines are functions that are “localized in frequency”, however they are not localized in space. Wavelets are localized in space and to a certain extent also in frequency.

There are many basis of wavelets. Their goal is to render transforms and operators sparse. Just like convolutions and other operations are easy when Fourier transformed other operations become simple when wavelet transformed.

Wavelet theory was pushed forward by Ingrid Daubechies (1983 and on).

keyword in over 20,000 articles by now.

An example of a wavelet basis is called “DAUB4”.

$$\begin{bmatrix}
c_0 & c_1 & c_2 & c_3 & & & & & & & \\
c_3 & -c_2 & c_1 & -c_0 & & & & & & & \\
 & & c_0 & c_1 & c_2 & c_3 & & & & & \\
 & & c_3 & -c_2 & c_1 & -c_0 & & & & & \\
\vdots & \vdots & & & & & \ddots & & & & \\
 & & & & & & & c_0 & c_1 & c_2 & c_3 \\
 & & & & & & & c_3 & -c_2 & c_1 & -c_0 \\
c_2 & c_3 & & & & & & & & c_0 & c_1 \\
c_1 & -c_0 & & & & & & & & c_3 & -c_2
\end{bmatrix}$$

Consider a matrix acting on a “signal vector”.

The first row just multiplies the four first components of the signal times four coefficients.

If the even rows did the same thing (shifted by one entry) then one would be doing an ordinary convolution, which we could do via FFT.

Here however, the even rows do a different multiplication. The end result is like if one had split the data into two sets of half the length and convolved them with different filter functions, interleaving the final result into a single set.

It is useful to think of the $c_0 \ldots c_3$ filter as a "smoothing" filter G that does a four-point average of the signal. The $c_3,-c_2,c_1,-c_0$ filter (let's call it H) is of a very different nature. It is designed to yield zero for a sufficiently smooth function (for instance by requesting that the sequence of c's have a certain number of vanishing moments).

In signal processing language, filters like G,H are called "quadrature mirror" filters. One filter smoothes out the signal whereas the other picks up whatever non-smooth ("detail") information the signal might have.

In the wavelet lingo, having a filter that has p vanishing moments is referred to as "approximation condition of order p".

Now, we are trying to create a "transform", not just a "filter", meaning we ought to be able to reconstruct the original signal from the transformed data. This is achieved by requiring that the matrix we introduced be orthogonal.

Requesting that the matrix be orthogonal imposes conditions on its coefficients,

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1$$

$$c_2 c_0 + c_3 c_1 = 0$$

If we further request that the transform be of "approximation condition order" equal to 2 (vanishing of first two moments of H), then we get,

$$c_3 - c_2 + c_1 - c_0 = 0$$

$$0c_3 - 1c_2 + 2c_1 - 3c_0 = 0$$

Which implies that the coefficients are

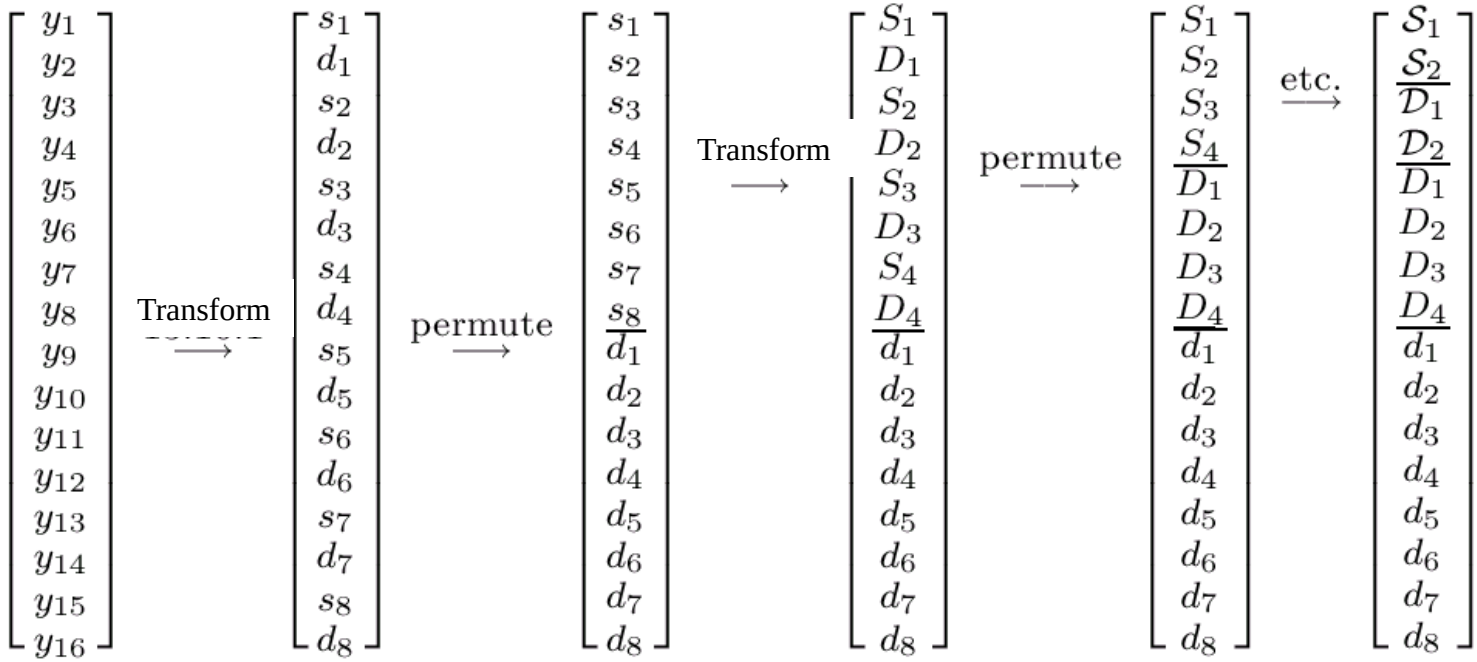
$$c_0 = (1 + \sqrt{3})/4\sqrt{2} \qquad c_1 = (3 + \sqrt{3})/4\sqrt{2}$$

$$c_2 = (3 - \sqrt{3})/4\sqrt{2} \qquad c_3 = (1 - \sqrt{3})/4\sqrt{2}$$

One can repeat the same story with more coefficients. DAUB6 uses six coefficients and sets to zero the first three moments. Daubechies book "Wavelets" (SIAM Press 1992) lists several more bases.

We are now ready to define a wavelet transform. The transform consists in applying the transformation matrix we proposed. One is left with a vector of length equal to the one defining the signal with alternating “smooth” and “detailed” components.

Then one goes ahead and applies the matrix again, to the N/2 length vector of smooth components. Then one goes again, this time to the N/4 vector of “smooth-smooth” components, and so on, until one is left with with only two smooth components. An image is worth 1000 words in this case...

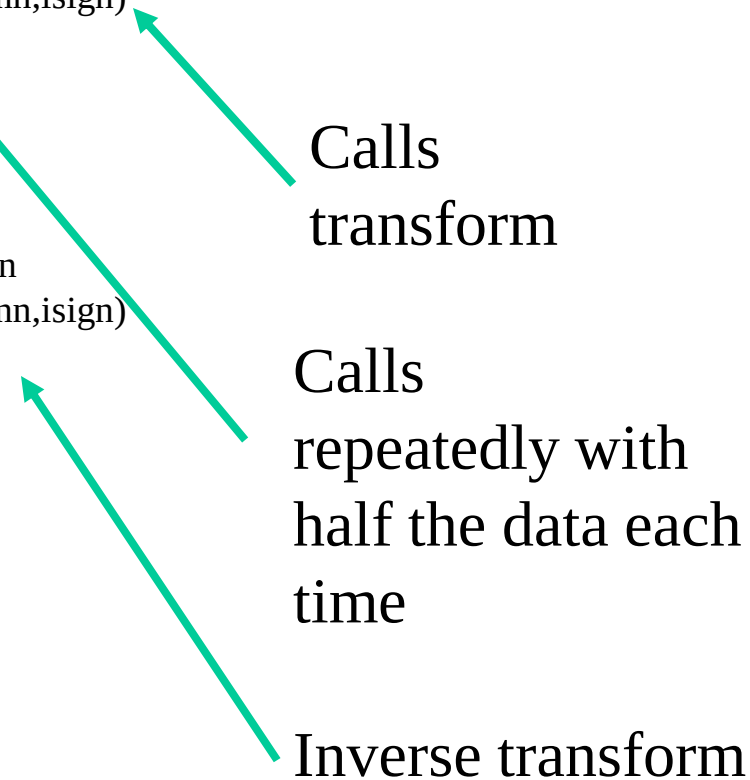


The output of the transform is therefore two smooth components and N-2 “detailed” components.

Since the transform is a product of orthogonal transformations, it is in itself an orthogonal transformation. It can therefore be “undone” by going through the pyramid we showed in the picture in the reverse order (using the inverse matrix for the transforms).

As we noted, the matrix has “wrap around” coefficients assuming the data is periodic. This implies that the first and last wavelet coefficients will contain information about both the beginning and the end of the signal. There is a way of modifying the matrix such that one eliminates this (and is left with a band-diagonal matrix), but we will not discuss it here.

```
 SUBROUTINE wt1(a,n,isign,wtstep)
      INTEGER isign,n
      REAL a(n)
      EXTERNAL wtstep
CU    USES wtstep
      INTEGER nn
      if (n.lt.4) return
      if (isign.ge.0) then
        nn=n
1       if (nn.ge.4) then
          call wtstep(a,nn,isign)
          nn=nn/2
        goto 1
        endif
      else
        nn=4
2       if (nn.le.n) then
          call wtstep(a,nn,isign)
          nn=nn*2
        goto 2
        endif
      endif
      return
      END
```



```
SUBROUTINE daub4(a,n,isign)
    INTEGER n,isign,NMAX
    REAL a(n),C3,C2,C1,C0
    PARAMETER (C0=0.4829629131445341,C1=0.8365163037378079,
   *C2=0.2241438680420134,C3=-0.1294095225512604,NMAX=1024)
    REAL wksp(NMAX)
    INTEGER nh,nh1,i,j
    if(n.lt.4)return
    if(n.gt.NMAX) pause 'wksp too small in daub4'
    nh=n/2
    nh1=nh+1
    if (isign.ge.0) then
      i=1
      do 11 j=1,n-3,2
        wksp(i)=C0*a(j)+C1*a(j+1)+C2*a(j+2)+C3*a(j+3)
        wksp(i+nh)=C3*a(j)-C2*a(j+1)+C1*a(j+2)-C0*a(j+3)
        i=i+1
11      continue
      wksp(i)=C0*a(n-1)+C1*a(n)+C2*a(1)+C3*a(2)
      wksp(i+nh)=C3*a(n-1)-C2*a(n)+C1*a(1)-C0*a(2)
    else
      wksp(1)=C2*a(nh)+C1*a(n)+C0*a(1)+C3*a(nh1)
      wksp(2)=C3*a(nh)-C0*a(n)+C1*a(1)-C2*a(nh1)
      j=3
      do 12 i=1,nh-1
        wksp(j)=C2*a(i)+C1*a(i+nh)+C0*a(i+1)+C3*a(i+nh1)
        wksp(j+1)=C3*a(i)-C0*a(i+nh)+C1*a(i+1)-C2*a(i+nh1)
        j=j+2
12      continue
    endif
    do 13 i=1,n
      a(i)=wksp(i)
13    continue
    return
    END
```

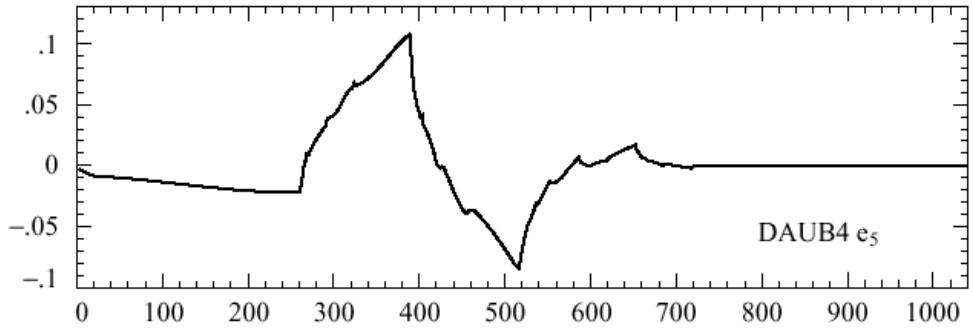
Smooth

Detail, shift to second half of output

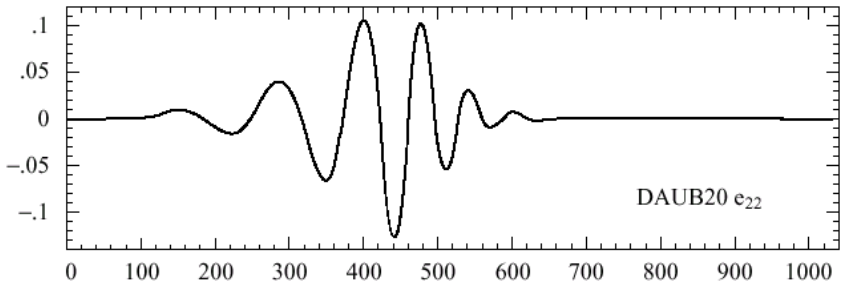
Inverse

Treat specially wrap-around points

What do these wavelets look like? Here is the inverse transform of a wavelet vector of length 1024 that is zero in all components except the fifth one, which is set to unity, using DAUB4,




This is a DAUB20 transform of a wavelet vector with non-vanishing 22nd component,

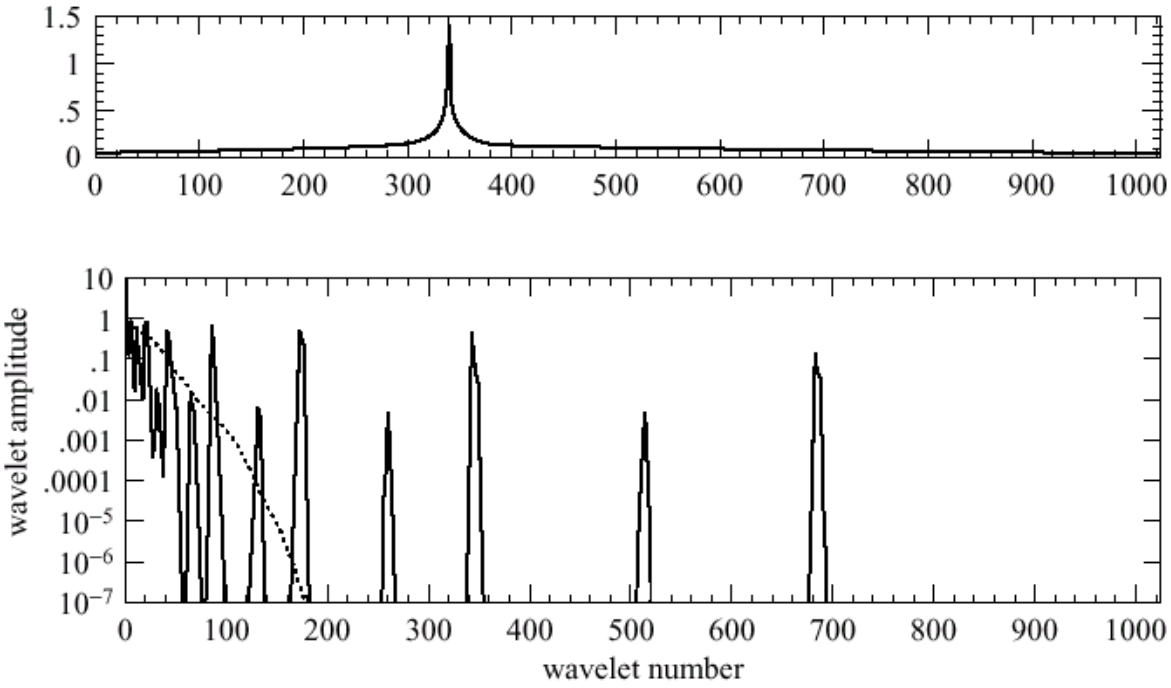



A basis of wavelet vectors consists in translations and scalings of these functions. Lower order wavelets have cusps in the function, higher order ones will show cusps in the derivatives.

One might be concerned about the presence of cusps. Can one represent "reasonable" functions in this basis? The answer is yes: cusps of the various basis elements can cancel each other to yield functions without cusps. That this is true is guaranteed by the fact that the transforms have certain moments equal to zero. At least for functions representing the powers of t involved in computing such moments, the wavelet is guaranteed to represent them exactly.

Standard mathematical operations are not "easier" in wavelet space (as say, a convolution was in Fourier space).
**Why are wavelets useful?** Because a transformed function can be **severely truncated** without significant loss of information.

In Fourier space a truncation is bad because one uses lots of Fourier components to account for the usual "locality" of most signal structures. In wavelet space one incorporates locality into the elements of the basis, so one does not need infinitely many of them to represent local functions.



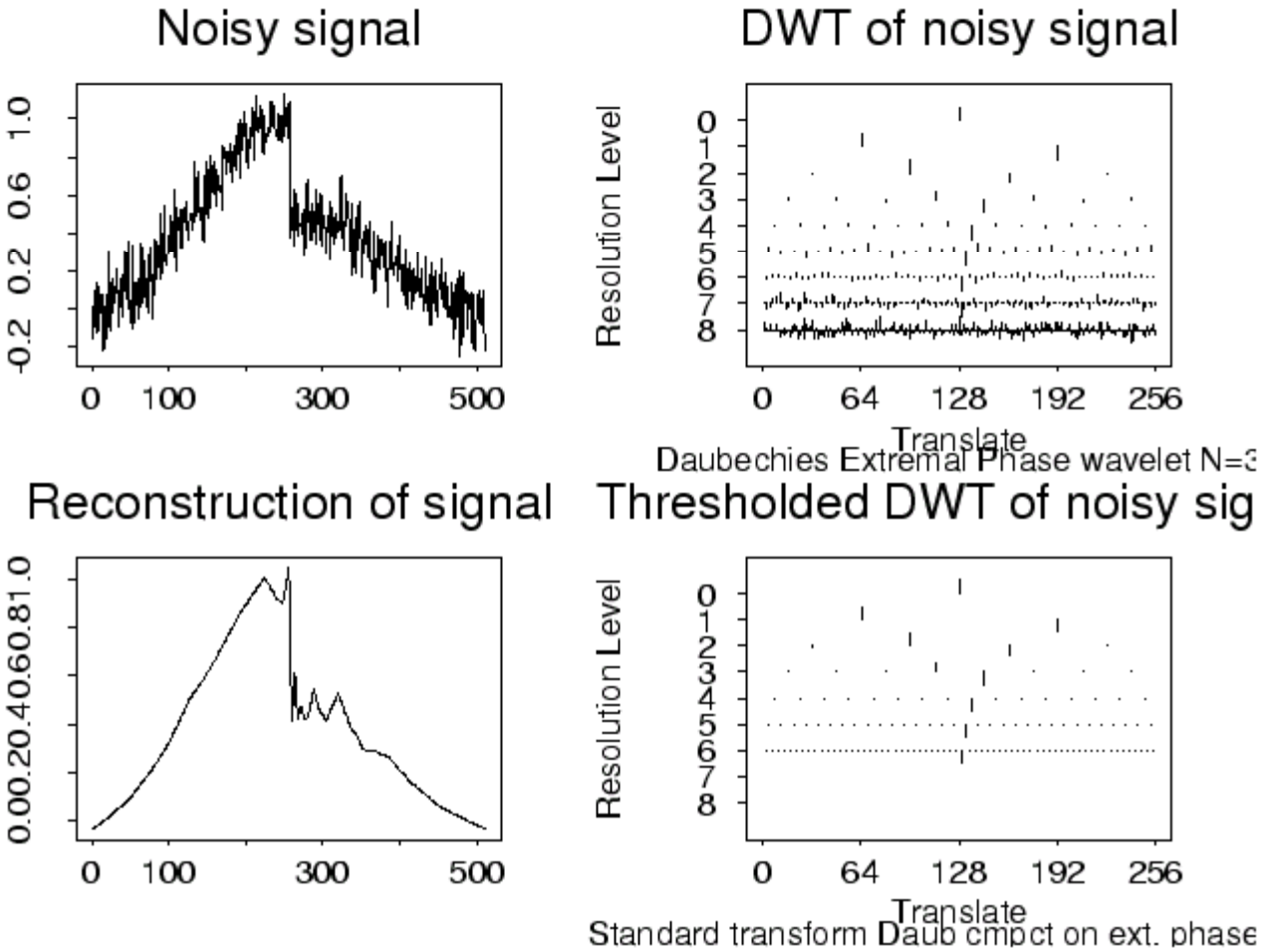

Here we show a signal with localized structure and its wavelet components.

The dotted curve represents the transform sorted in decreasing amplitude.

We see that about 130 of the components has amplitudes larger than $10^{-5}$. Therefore we could keep only 130 components out of the 1024 (provided we remember where they belong in wavelet number!) and lose information that in terms of power is only $10^{-12}$ of the signal.

Compact wavelets are better for dramatically cutting storage space of functions with discontinuities (like video images) whereas smooth wavelets are better to represent smooth functions at high accuracy (for instance solving integral or certain differential equations).

Wavelets are great for getting rid of noise. Since the wavelet transform of noise is largely noise, whereas that of a signal has large, distinct components (and we can do very well without many of them), one then only needs to do a wavelet transform, apply a threshold and inverse transform to “clean” a signal,




What are the diagrams on the right?

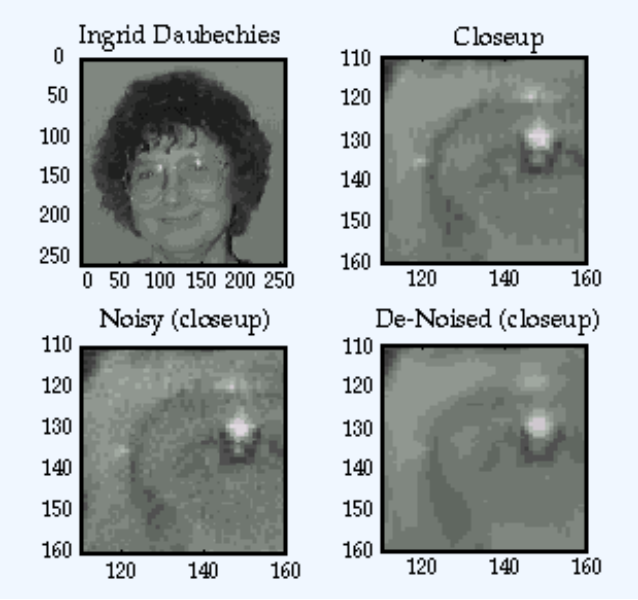
The diagrams on the right correspond to a different notation of the wavelet basis than the one we used. We noted that the various components of the basis of wavelets were obtained by shifting and scaling the “mother wavelet”. Some people refer to this explicitly,

The diagrams we saw correspond to thinking of the wavelet as an expansion

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{jk}\psi_{jk}(x)$$

Where the basis functions are denoted by “level” (amplitude) and translation (shift)

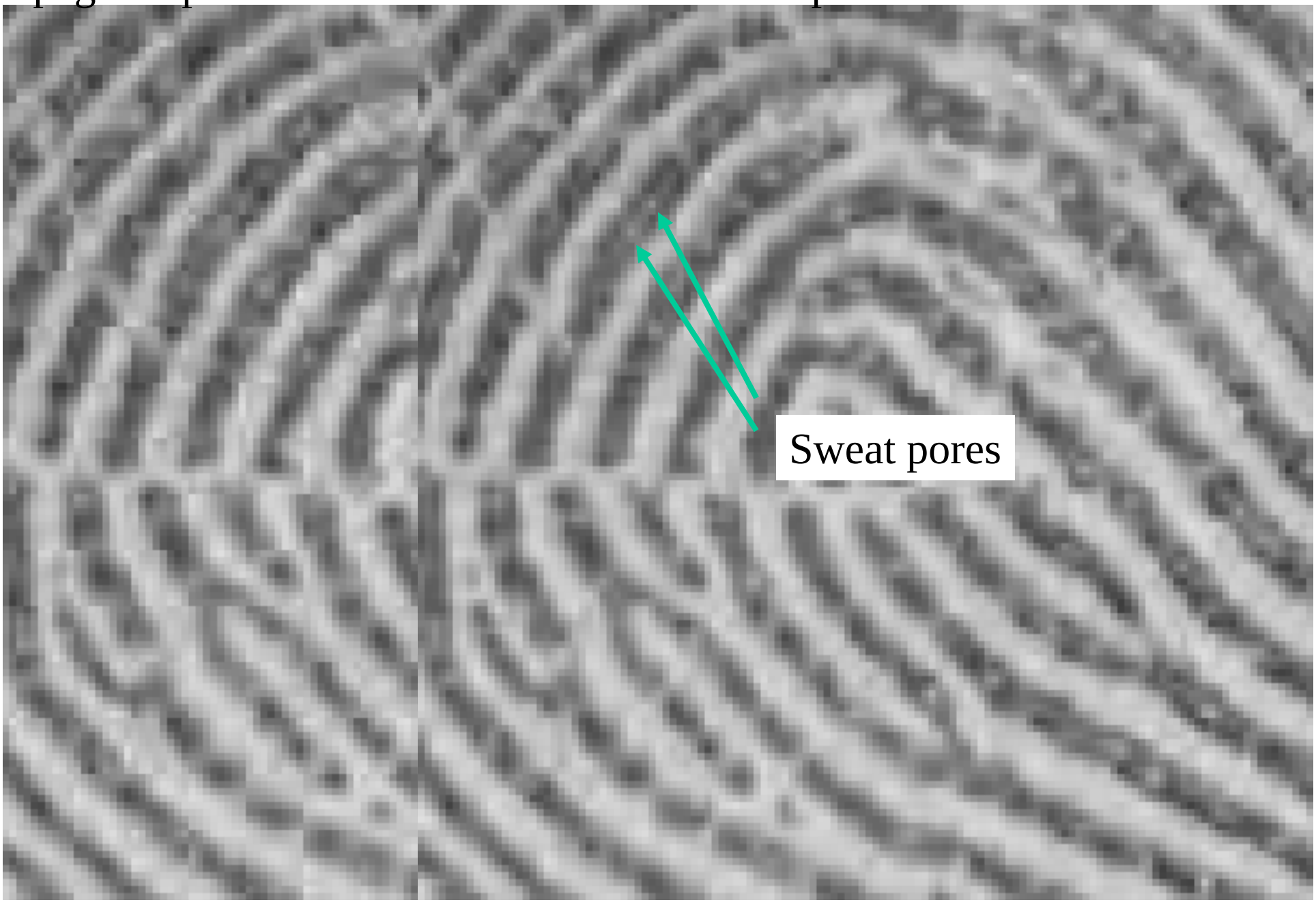
$$\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k)$$




www.wavelet.org

FBI fingerprint file (700k original) (200 million of them!)

Jpeg compressed to 45k Wavelet compressed to 45k



A promising area for wavelet transforms is the solution of very large systems of linear equations. The idea is to consider the matrix A of A.x=b as an “image”. If the “image” compresses well under the wavelet transform, the solution of the resulting (much sparser) system of equations will therefore be a good approximation to the original system.

Here again it is nice that the wavelet transform can be thought of as an orthogonal transformation. Starting from

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

We transform,

$$\widetilde{\mathbf{A}} \equiv \mathbf{W} \cdot \mathbf{A} \cdot \mathbf{W}^T, \qquad \widetilde{\mathbf{b}} \equiv \mathbf{W} \cdot \mathbf{b}$$

Where W is a one-dimensional wavelet operator. We then solve the much sparser system given by A-tilde and transform back.

# Summary

- By using bases that are localized in frequency and space, one can render operators sparse.
- This allows enormous economies of space and acceleration of speed.
- Applications in many areas.