1

IMACS Matrices

Gilbert Strang

October 26, 2000

Abstract

So much of mathematics is involved with the representation of functions. A central example in pure and applied mathematics is the Fourier series. Its discrete version is computed by the Fast Fourier Transform, which is the most important algorithm of the last century (and we report briefly on its approximation by a "shift and add" binary transform). The Fourier basis is terrific – but imperfect. The basis functions are global instead of local, and they give poor approximation at a discontinuity (Gibbs phenomenon). New functions are being developed for interpolation and approximation and compression and many other applications.

Four properties we want are: local basis, easily refined, fast to compute, good approximation by a few terms. Splines and finite elements achieve the first three, but they don't allow compression; if we remove terms the approximation fails. So we turn to the wavelet construction to permit compression of data – which is needed in so many modern applications where the volume of data is overwhelming. Wavelets have two types of basis functions, one for averages and the other (the wavelets themselves) for details at all scales. When the details are not necessary they can be compressed away to leave a smoothed signal. That construction has entered the new IEEE standards for signal processing. We will explain how the construction is achieved with two filters (where one filter would fail). In matrix terms we get a banded block-Toeplitz matrix with a banded inverse.

Finally we discuss the localized eigenvectors that appear when a few entries are changed in a familiar tridiagonal Toeplitz matrix.

Key words: Toeplitz matrices, cosine transform, wavelets, filters, perturbations.

AMS subject classifications: 65T50, 42C40, 65T60.

*Department of Mathematics, M.I.T. Cambridge MA 02139, gs@math.mit.edu, http://www-math.mit.edu/~gs

16th IMACS World Congress (© 2000 IMACS)

1 Introduction

This talk is for people who like matrices. I don't mean every matrix in the world. If you like all matrices indiscriminately, that might be too much. I really mean structured matrices — the kind that appear in solving PDEs, and the kind that appear in signal processing, and the kind that appear in control theory. These are IMACS matrices!

Many of those matrices are Toeplitz, or nearly Toeplitz (affected by boundary conditions), or even circulant (from periodic boundary conditions). Their eigenvectors are often oscillatory. For circulants, the eigenvectors are the columns of the Fourier matrix (DFT matrix) — which is the one overwhelmingly important complex matrix. The eigenvectors of simple second difference matrices are discrete sines or discrete cosines. They go into the DST and DCT; these are very useful bases. Those eigenvector matrices are often orthogonal, and not sparse but full. Since a lot of applied mathematics is about choosing and also changing bases, we need to multiply by these matrices.

Let me pause a moment. I have touched on several of the themes of this lecture, and it may be useful to list them:

1. One small part is to report new "shift and add" developments in the implementation of the DFT and DCT. A new matrix problem appears: Approximation by binary matrices.

2. A larger part is the discussion of good bases — and particularly to explain wavelet bases. I would like to demystify the wavelet construction. The Discrete Wavelet Transform comes from a simple idea, that block Toeplitz matrices and their inverses can both be banded (and therefore very fast to compute).

3. The third part is to discuss the eigenvalues and eigenvectors of nearly Toeplitz matrices. In this talk, "nearly" does not refer to slow variation down the diagonal, produced by variable coefficients in a differential equation. We allow large changes in a Toeplitz matrix, but those modifications must be widely spaced (and low rank). What is the form of the eigenvectors that appear when a few entries of a structured matrix are modified? Our answer will be: Those new eigenvectors are highly localized.

   Note the use of the word "modification". A perturbation is usually something small. A modification can be large. Physicists are interested in random changes along the diagonal of a structured matrix, but our changes are most often in the interior.

   We devote a section to each of these three topics. The first reports on work in the signal processing literature. The second is adapted from the textbook [6] and lecture notes [9] on wavelets. The third comes from a joint paper [10] with my students Xiangwei Liu and Susan Ott that is being prepared for publication.

2 Transforms by Shifts and Adds

This brief section is included to call attention to recent results on "low power matrix multiplication". A possible application is to execute signal processing on board a satellite. The available power is very limited, but the advantages of compression before relaying images to the Earth are highly significant. There are other applications to computing in a mobile environment.

Two immediate choices for a "fast binary transform" are the DFT and the DCT. The Discrete Cosine Transform is extremely useful in image processing, where its Neumann boundary conditions [6, 7] offer an extra level of continuity in comparison with a periodic (circular or cyclic) transform. The DFT matrix is complex, the DCT is real, and both matrices are full:

$$F_{jk} = \exp(2\pi ijk/N) \quad \text{and} \quad C_{jk} = \cos(\pi j(k+\frac{1}{2})/N) \; j,k = 0,\ldots,N-1$$

(with the exception that $C_{0k} = 1/\sqrt{2}$). We concentrate here on the DCT. Very accurate matrix approximations have been developed by Tran [11, 12] and by Chen, Oraintara and Nguyen [1, 2]. The result announced in [1] is a surprising 8 by 8 integer DCT with a complexity of 45 adds and 18 shifts.

Their construction begins with an exact factorization of the DCT matrix into a product that involves simpler orthogonal matrices (and especially plane rotations). Normally, an N by N orthogonal matrix can be factored into $N(N-1)/2$ rotations, but the special structure of the DCT matrix allows a much shorter and more efficient factorization:

$$C = (5 \;\; \text{rotations}) \;\; (\text{permutation}) \;\; (\text{Hadamard matrix})$$

The permutation matrix is a bit-reversal that reorders the components to 0,4,2,6,1,5,3,7. The Hadamard matrix requires only additions, because its entries are ± 1. Then each rotation (the angles are $-\pi/8$, $3\pi/8$, $3\pi/8$, $7\pi/16$, and $3\pi/16$) is approximated by a binary (shift and add) matrix transform, starting from this factorization:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \frac{\cos\theta-1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\cos\theta-1}{\sin\theta} \\ 0 & 1 \end{bmatrix}$$

Breaking each rotation into these three "lifting steps", we have a total of 15 off-diagonal multipliers. Those are floating-point numbers. To reach a pure "shift and add" implementation, those numbers are quantized into the form $k/2^b$. The papers [1, 2] give careful descriptions of the whole construction. (The multiplier $\frac{(\cos\theta-1)}{\sin\theta}$ exceeds 1 in magnitude when $\frac{\pi}{2} < \theta \leq \pi$. A different factorization is used in this range.)

My purpose here is to call your attention to this family of new and interesting questions about matrices. We have bases that we know to be good (DFT and DCT). They are important enough to be optimized for very fast and accurate implementation.

3 Wavelets and Good Bases

In the past, signal processing was a topic that stayed almost exclusively in electrical engineering. It was only the specialists who applied lowpass filters to remove high frequencies from digital signals. The experts could cancel unwanted noise. They could compress the signal and then reconstruct. It took two-dimensional experts to do the same for images.

The truth is that everyone now deals with digital signals and images (involving large amounts of data). We all need to understand signal processing — *sampling, transforming, and filtering.*

A filter is the most important operation in signal processing. It acts on a signal to produce a modified signal. Usually some frequency components of the input signal are reduced; it is remarkable how simply this can be done. When the filter is FIR (*finite* impulse response), each output sample $y(n)$ is just a linear combination of a *finite* number of input samples.

The simplest example is a moving average

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)\,.$$

This filter combines each sample $x(n)$ with the previous sample $x(n-1)$. The weights in the linear combination are the filter coefficients $\frac{1}{2}$ and $\frac{1}{2}$. The filter is *time-invariant* because those coefficients are constant for all time. The filter is *causal* because it involves no future samples like $x(n+1)$. The effect $y(n)$ never comes earlier than its causes $x(n)$ and $x(n-1)$. Thus we have a causal linear time-invariant FIR system.

A causal FIR filter of higher order N has coefficients $h(0), h(1), \ldots, h(N)$. Notice that there are $N+1$ coefficients; this is the *length* of the filter. These coefficients stay fixed for all time so the filter is time-invariant.

At each time step, the $N+1$ coefficients multiply $N+1$ samples from the input signal — the current sample $x(n)$, the previous sample $x(n-1)$, continuing back to the sample $x(n-N)$. This weighted combination of input values produces the output $y(n)$:

$$y(n) = h(0)x(n) + h(1)x(n-1) + \cdots + h(N)x(n-N)\,.$$

This is the action of the filter in the time domain. We may write it compactly as a sum from $k=0$ to $k=N$:

$$y(n) = \sum_{k=0}^{N} h(k)x(n-k)\,.$$

A filter is a discrete convolution! It is the fundamental operation for discrete time-invariant systems. To implement this convolution in hardware, we only need three building blocks: *unit delay, multiplier*, and *adder*.

How does a filter look in the frequency domain? The basic rule is that ***a convolution becomes a multiplication***. We multiply the Fourier series $H(\omega)$ with coefficients $h(k)$ and the Fourier series $X(\omega)$ with coefficients $x(m)$.The result is the series $Y(\omega)$ with coefficients $y(n)$. In the frequency domain, the output Y is the input X multiplied by the response function H:

$$\begin{aligned} \sum y(n)e^{-in\omega} &= \left(\sum h(k)e^{-ik\omega}\right)\left(\sum x(m)e^{-im\omega}\right) \\ Y(\omega) &= H(\omega)X(\omega) \end{aligned}$$

To get $e^{-in\omega}$ *in the product of* $e^{-ik\omega}$ *with* $e^{-im\omega}$*, we must have* $k+m=n$. That is exactly what we see in the convolution. The indices k and $n-k$ add to n. The products $e^{-ik\omega}$ and $e^{-i(n-k)\omega}$ multiply to give $e^{-in\omega}$.

This "convolution rule" is more than just algebra because $Y = HX$ has a valuable scientific meaning:

Each component $X(\omega)$ of the input is amplified by the filter response $H(\omega)$ (or $H(e^{i\omega})$) at that frequency ω.

Now filter the input signal $\delta(n)$, an impulse. In the time domain, the convolution has only one term $k = n$:

$$y(n) = \sum h(k)\delta(n-k) = h(n)\,.$$

In the frequency domain, the rule $Y(\omega) = H(\omega)X(\omega)$ reduces to $Y(\omega) = H(\omega)$. As predicted, this matches $y(n) = h(n)$ in the time domain. ***The transform of the impulse response*** $\{h(n)\}$ ***is the frequency response*** $H(\omega)$**.** This is the function that describes the filter in the frequency domain:

$$H(\omega) = \sum_{k=0}^{N} h(k)e^{-ik\omega} = h(0) + h(1)e^{-i\omega} + \cdots + h(N)e^{-iN\omega}\,.$$

A filter is lowpass when low frequencies have $H(\omega) \approx 1$. They pass through the filter. High frequencies have $H(\omega) \approx 0$, so $H(\omega)X(\omega)$ is very small at these frequencies. The output is a *smoothed* version of the input. If we want to keep high frequencies, we use a highpass filter.

3.1 Wavelet Transforms

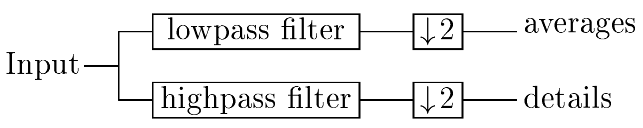
A lowpass filter greatly reduces the high frequency components, which often represent noise in the signal. For some purposes that is exactly right. But suppose we want to *reconstruct* the signal. We may be storing it or transmitting it or operating on it, but we don't want to lose it. In this case we can use *two filters*, highpass as well as lowpass. That generates a "**filter bank**," which sends the signal through two or more filters.

The filter bank structure leads to a *Discrete Wavelet Transform*. This has become a guiding idea for so many problems in signal analysis and synthesis. In itself the transform is lossless! Its inverse (the synthesis step) is another transform of the same type — two filters that are fast to compute. Between the DWT and the inverse DWT we may compress and transmit the signal. This sequence of steps, *transform then compress then reconstruct*, is the key to more and more applications.

The word *wavelet* is properly associated with a *multiresolution into different scales*. The simplest change of scale comes from downsampling a signal — keeping only its even-numbered components $y(2n)$. This sampling operation is denoted by the symbol $\downarrow 2$:

$$(\downarrow 2)\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} y(0) \\ y(2) \end{bmatrix}.$$

Information is lost. But you will see how *doubling* the length by using two filters, then *halving* each output by $(\downarrow 2)$, can give a lossless transform. The input is at one time scale and the two half-length outputs are at another scale (an octave lower).



The discrete wavelet transform: averages and details.

Note that an input of even length L produces two outputs of length $L/2$, after downsampling. The lowpass filter H_0 and the highpass filter H_1 originally maintain length L, when we deal suitably with the samples at the ends (possibly by extending the signal to produce the extra components that the filter needs). When the redundancy from $2L$ outputs is removed by $(\downarrow 2)$, the overall filter bank is L by L. It is a change of basis!

To simplify the theory we often pretend that $L = \infty$. This avoids any difficulty with the samples at the ends. In reality signals have finite length.

The wavelet idea is to repeat the filter bank. The lowpass output becomes the input to a second filter bank. The computation is cut in half because this input is half length. Typical applications of the wavelet transform go to four or five levels. We could interpret this multiscale transform as (quite long) filters acting on the very first inputs. But that would miss the valuable information stored in the outputs (averages and details) along the way. Those outputs are the coefficients of the input when expressed in the wavelet basis.

3.2 The Haar Transform

We now choose one specific example of a filter bank. At first there is no iteration (two scales only). Then we iterate to multiple scales. The example is associated with the name of Alfred Haar. It uses the *averaging filter* (moving average) and the *differencing filter* (moving difference). They combine naturally into the most basic example of a filter bank.

The two filters are denoted by H_0 (lowpass) and H_1 (highpass):

$$\begin{array}{lll} y_0 = H_0 x & \text{is the averaging filter} & y_0(n) = \frac{1}{2}(x(n-1) + x(n)) \\ y_1 = H_1 x & \text{is the differencing filter} & y_1(n) = \frac{1}{2}(x(n-1) - x(n)) \,. \end{array}$$

Suppose the input signal is zero except for four samples $x(1) = 6$, $x(2) = 4$, $x(3) = 5$, $x(4) = 1$. This input vector is $x = (6, 4, 5, 1)$. *We are looking for its coefficients in the Haar wavelet basis.* Those will be four numbers, $y_0(2)$ and $y_0(4)$ from subsampling the lowpass output together with $y_1(2)$ and $y_1(4)$ from highpass.

In reality we would not compute the odd-numbered components $y(1)$ and $y(3)$ since they are immediately destroyed by $(\downarrow 2)$. But we do it here to see the complete picture. Take averages and differences of $x = (6, 4, 5, 1)$:

$$\text{Averages} \quad \begin{array}{rcl} y_0(1) & = & 3 \\ y_0(2) & = & 5 \\ y_0(3) & = & 4.5 \\ y_0(4) & = & 3 \\ y_0(5) & = & 0.5 \end{array} \qquad \tfrac{1}{2}\,(\text{Differences}) \quad \begin{array}{rcr} y_1(1) & = & -3 \\ y_1(2) & = & 1 \\ y_1(3) & = & -0.5 \\ y_1(4) & = & 2 \\ y_1(5) & = & 0.5 \end{array}$$

You might notice that the sum of the y_0 vector and the y_1 vector is the input $x = (6, 4, 5, 1)$ with a unit delay (to $x(n-1)$). This comes from a simple relation (average + difference) that is special to Haar:

$$\frac{1}{2}(x(n-1) + x(n)) + \frac{1}{2}(x(n-1) - x(n)) = x(n-1) \,.$$

It is more important to notice that the *differences tend to be smaller than the averages.* For a smooth input this would be even more true. So in a compression step, when we often lose information in the highpass coefficients, the loss is small using the wavelet basis. Here is a first look at the whole compression algorithm:

$$\textit{signal } x \xrightarrow[\text{[lossless]}]{\qquad} \begin{matrix}\textit{wavelet}\\ \textit{coefficients}\end{matrix} \xrightarrow[\text{[lossy]}]{\qquad} \begin{matrix}\textit{compressed}\\ \textit{coefficients}\end{matrix} \xrightarrow[\text{[lossless]}]{\qquad} \begin{matrix}\textit{compressed}\\ \textit{signal}\end{matrix}\ \widehat{x}$$

At this point we have eight or even ten coefficients in y_0 and y_1. They are redundant! They came from only four samples in x. *Subsample* by $\downarrow 2$ to keep only the even-numbered components:

$$\begin{aligned} y_0(2) &= 5 \qquad & y_1(2) &= 1 \\ y_0(4) &= 3 \qquad & y_1(4) &= 2\,. \end{aligned}$$

Those are the four "first-level wavelet coefficients" of the signal. The inverse transform (which is coming in the next section) will use those coefficients to reconstruct x. That will be the synthesis step. Computing the coefficients was the analysis step:

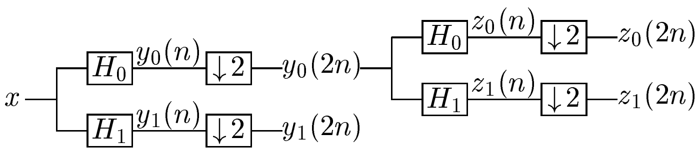
Analysis:	Find the wavelet coefficients (separate the signal into wavelets)
Synthesis:	Add up wavelets times coefficients (to reconstruct the signal).

It is like computing Fourier coefficients, and then summing the Fourier series. For wavelets, the *analysis filter bank* (H_0 and H_1 followed by $\downarrow 2$) computes the coefficients. The *synthesis filter bank* (this inverse wavelet transform will involve upsampling by $\uparrow 2$ and two filters) sums the wavelet series.

Now go from the lowpass $y_0(2) = 5$ and $y_0(4) = 3$ to the next scale by computing *averages of averages and differences of averages*:

$$z_0(2) = \frac{5+3}{2} = 4 \qquad z_1(2) = \frac{5-3}{2} = 1\,.$$

This completes the iteration of the Haar analysis bank. We can see the three scales (fine, medium, and coarse) in a block diagram that shows the tree of filters with subsampling:



Effectively, z comes from downsampling by 4. The vector $z_0(2n)$ contains the *low-low* coefficients, averages of averages. The high-low vector $z_1(2n)$ is also $\frac{1}{4}$ the length of the original signal. The highpass vector $y_1(2n)$ is half the original length, and $\frac{1}{4} + \frac{1}{4} + \frac{1}{2} = 1$ (this is critical length sampling).

You will ask, why not take averages and differences also of this first highpass output $y_1(2n)$? That is certainly possible. A "wavelet packet" might choose to do it. But the basic wavelet tree assumes that the highpass coefficients are small and not worth the additional effort. They are candidates for compression. For a typical long smooth signal, iteration to four or five scale levels will further decorrelate the samples. Iterations beyond that are generally not worthwhile.

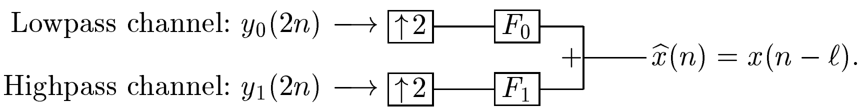
Summary: The input is $x = (6, 4, 5, 1)$. The wavelet coefficients are

$$(\text{low-low } z_0, \text{ high-low } z_1, \text{ high } y_1) = (4, 1, 1, 2).$$

The special point of the wavelet basis is that you can pick off the highpass details (1 and 2 in y_1), before the coarse details in z_1 and the overall average in z_0.

3.3 Reconstruction by the Synthesis Bank

Other analysis banks don't have this simple "two at a time" block structure, but the basic principle still applies. We will show how filtering followed by downsampling is inverted by **upsampling followed by filtering**. This sequence of inverse operations (and the notation) is displayed in the following diagram. *This is the synthesis bank:*



The goal is to recover the input exactly, when no compression has been applied to the wavelet transform $z(n)$. The synthesis bank does recover x but with ℓ delays (ℓ depends on the filter bank). Long delays are not desirable, but causal filters cannot avoid some delay. The filter only looks backward.

First, we show how the combination of upsampling and filtering recovers the input to the Haar filter (with one delay!).

$$(\uparrow 2)\, y_0(2n) = \begin{bmatrix} y_0(0) \\ 0 \\ y_0(2) \\ 0 \end{bmatrix} \longrightarrow \boxed{\begin{array}{c}\text{sum}\\ \text{filter } F_0\end{array}} \longrightarrow \begin{bmatrix} y_0(0) \\ y_0(0) \\ y_0(2) \\ y_0(2) \end{bmatrix}$$

$$(\uparrow 2)\, y_1(2n) = \begin{bmatrix} y_1(0) \\ 0 \\ y_1(1) \\ 0 \end{bmatrix} \longrightarrow \boxed{\begin{array}{c}\text{difference}\\ \text{filter } F_1\end{array}} \longrightarrow \begin{bmatrix} y_1(0) \\ -y_1(0) \\ y_1(2) \\ -y_1(2) \end{bmatrix}$$

$$+ \;\longrightarrow\; \begin{bmatrix} x(-1) \\ x(0) \\ x(1) \\ x(2) \end{bmatrix}.$$

The choice of synthesis filters F_0 and F_1 is directly tied to the analysis filters H_0 and H_1 (since the two filter banks are inverses). We will write down the rule for F_0 and F_1, and show that it succeeds (**perfect reconstruction**). F_0 comes from *alternating the signs* in H_1. This takes the highpass H_1 into a lowpass filter F_0. Similarly, the highpass F_1 comes from alternating the signs of the coefficients in H_0. An example will make the point more clearly:

$$h_0 = \frac{1}{8}(-1, 2, 6, 2, -1) \qquad f_0 = \frac{1}{2}(1, 2, 1)$$

$$h_1 = \frac{1}{2}(1, -2, 1) \qquad f_1 = \frac{1}{4}(1, 2, -6, 2, 1).$$

The coefficients of h_1 and f_1 add to zero; a zero-frequency signal (a constant DC signal) is killed by these highpass filters. The coefficients of h_0 add to 1 and the coefficients of f_0 add to 2. The filters $h_0 = (\frac{1}{2}, \frac{1}{2})$ and $f_0 = (1, 1)$ in the Haar example followed these rules.

Since you like matrices, let me express the analysis bank (the filters h_0 and h_1) as a matrix multiplication:

$$A = \begin{bmatrix} -1 & 2 & 6 & 2 & -1 & & & & \\ & & -1 & 2 & 6 & 2 & -1 & & \\ & & & & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & -2 & 1 & 0 & & & & \\ & & 0 & 1 & -2 & 1 & 0 & & \\ & & & & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

The double shift comes from the downsampling step (the matrix for an ordinary filter has rows shifted by one, so it is a constant-diagonal "Toeplitz matrix"). In the usual model, the input and output vectors are infinite, with $-\infty < n < \infty$, so the matrix A is doubly infinite. The lowpass filter (normalized by $\frac{1}{8}$ or not) produces half of A and the highpass filter produces the other half.

To deal with a finite-length input we will assume periodicity. If $x(n+6) = x(n)$, then the last three columns of A are moved back into the first three columns. The 6×6 matrix becomes finite and square:

$$A_6 = \begin{bmatrix} -1 & 2 & 6 & 2 & -1 & 0 \\ -1 & 0 & -1 & 2 & 6 & 2 \\ 6 & 2 & -1 & 0 & -1 & 2 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & -2 \end{bmatrix}.$$

We have double shifts and wraparound in the rows of A_6. Periodicity allows this finite matrix to display the crucial properties of the wavelet transform. In practice we might be dealing with 1024 samples $x(n)$, and the matrix becomes A_{1024}. The rows still only have five or three nonzeros. Ax is computed in 8×1024 operations (multiplications and additions).

To display the synthesis filters we put their coefficients into the *columns* of a matrix S_6 or S_{1024}, again with the double shifts. Alternating signs will turn $1, -2, 1$ into a lowpass filter $1, 2, 1$. Similarly $-1, 2, 6, 2, -1$ becomes a highpass filter $1, 2, -6, 2, 1$ (with coefficients adding to zero).

$$S_6 = \frac{1}{16} \begin{bmatrix} 0 & 0 & 2 & 1 & 1 & -6 \\ 1 & 0 & 1 & 2 & 0 & 2 \\ 2 & 0 & 0 & -6 & 1 & 1 \\ 1 & 1 & 0 & 1 & -6 & 1 \\ 0 & 2 & 0 & 1 & -6 & 1 \\ 0 & 1 & 1 & 0 & 2 & 2 \end{bmatrix}.$$

This matrix produces the inverse wavelet transform. *A direct calculation verifies that* $AS = I$. This is true for 6×6 matrices, and it will remain true for N by N. Notice a remarkable fact: *Both matrices A and S have only $4N$ nonzeros.* The inverse transform (the synthesis step using S) is as fast as the analysis step using A. It is not usual for a sparse matrix to have a sparse inverse, but the wavelet construction makes this happen.

The wavelet basis vectors appear in the columns of S. Any matrix-vector multiplication Sv yields a combination of these columns. If the original input is a vector x, then the analysis step produces the coefficients $v = Ax$ in the wavelet expansion. The synthesis is:

$$x = S(Ax) = \text{ sum of (basis vectors) times (wavelet coefficients)}$$

We are using the all-important fact for square matrices that $AS = I$ implies $SA = I$.

The above equation can be interpreted as an ordinary matrix multiplication (combination of the columns of S) or equivalently as a change of basis. The columns gives the wavelet basis and Ax gives the coefficients of x in this basis. These coefficients are the numbers produced by the analysis bank.

We can verify that this analysis-synthesis filter bank gives perfect reconstruction with $\ell = 3$ delays: The output is $x(n-3)$. Suppose the input is a sinusoid $x(n) = e^{in\omega}$ at frequency ω. Then H_0 and H_1 become multiplications by their response functions $H_0(e^{i\omega})$ and $H_1(e^{i\omega})$. The combination of $(\downarrow 2)$ followed by $(\uparrow 2)$ introduces zeros in the odd-numbered components, which means that the **alias frequency** $\omega+\pi$ appears together with ω:

$$(\uparrow 2)\,(\downarrow 2)\,\{e^{in\omega}\} = \begin{bmatrix} e^{in\omega} \\ 0(\text{odd}\,n) \\ e^{i(n+2)\omega} \\ 0(\text{odd}\,n) \end{bmatrix} = \frac{1}{2}\left[\{e^{in\omega}\} + \{e^{in(\omega+\pi)}\}\right].$$

The factor $e^{in\pi}$ is $+1$ for even n and -1 for odd n. This aliased frequency $\omega+\pi$ is appearing in both channels of the filter bank. It has to be cancelled at the last step by F_0 and F_1. This was the reason behind the alternating signs (between h_0 and f_1 and between h_1 and f_0). Those alternations introduce powers of $e^{i\pi}$ in the frequency response functions:

$$\begin{aligned} F_0(e^{i\omega}) &= \frac{1}{2}(1+2e^{-i\omega}+e^{-2i\omega}) \\ &= \frac{1}{2}(1-2e^{-i(\omega+\pi)}+e^{-2i(\omega+\pi)}) \\ &= 2H_1(e^{i(\omega+\pi)}). \end{aligned}$$

Similarly, $F_1(e^{i\omega}) = -2H_0(e^{-i(\omega+\pi)})$. Now follow the pure exponential signal through analysis and synthesis, substituting these expressions for F_0 and F_1 when they multiply the signal (the last step in the filter bank):

$$\begin{aligned} H_0(e^{i\omega})e^{in\omega} &\longrightarrow \boxed{\downarrow 2}\!-\!\boxed{\uparrow 2} \longrightarrow \frac{1}{2}\left[H_0(e^{i\omega})e^{in\omega} + H_0(-e^{i\omega})e^{in(\omega+\pi)}\right] \\ &\longrightarrow \boxed{F_0} \longrightarrow H_1(-e^{i\omega})H_0(e^{i\omega})e^{in\omega} + H_1(-e^{i\omega})H_0(-e^{i\omega})e^{in(\omega+\pi)} \end{aligned}$$

$$\begin{aligned} H_1(e^{i\omega})e^{in\omega} &\longrightarrow \boxed{\downarrow 2}\!-\!\boxed{\uparrow 2} \longrightarrow \frac{1}{2}\left[H_1(e^{i\omega})e^{in\omega} + H_1(-e^{i\omega})e^{in(\omega+\pi)}\right] \\ &\longrightarrow \boxed{F_1} \longrightarrow -H_0(-e^{i\omega})H_1(e^{i\omega})e^{in\omega} - H_0(-e^{i\omega})H_1(-e^{i\omega})e^{in(\omega+\pi)}. \end{aligned}$$

These are the outputs from the low and high channels. When we add, the alias terms with frequency $\omega+\pi$ cancel out. The choice of F_0 and F_1 gave "alias cancellation."

The condition for perfect reconstruction comes from the terms involving $e^{in\omega}$. Those don't cancel! The sum of these terms should be $e^{i(n-\ell)\omega}$, which produces the

$$\textbf{PR Condition:} \qquad H_1(-e^{i\omega})H_0(e^{i\omega}) - H_0(-e^{i\omega})H_1(e^{i\omega}) = e^{-i\ell\omega}.$$

coefficient of this lowpass filter $P(\omega) = H_1(-e^{i\omega})H_0(e^{i\omega})$ *is nonzero*. That nonzero coefficient is the ℓ^{th}. We now verify that the example satisfies this PR condition:

$$\begin{aligned} P(e^{i\omega}) &= \frac{1}{4}(1+2e^{-i\omega}+e^{-2i\omega})\frac{1}{8}(-1+2e^{-i\omega}+6e^{-2i\omega}+2e^{-3i\omega}-e^{-4i\omega}) \\ &= \frac{1}{32}(-1+9e^{-2i\omega}+16e^{-3i\omega}+9e^{-4i\omega}-e^{-6i\omega}). \end{aligned}$$

The coefficients of the product filter P are $-1, 0, 9, 16, 9, 0, -1$ divided by 32. The middle coefficient is $\frac{16}{32} = \frac{1}{2}$. *The zeros in the other odd-numbered coefficients give perfect reconstruction.* And it is the particular coefficients in $-1, 0, 9, 16, 9, 0, -1$ that make P a powerful lowpass filter, because it has a *fourth-order zero* at the top frequency $\omega = \pi$. Factoring this special polynomial produces

$$P(e^{i\omega}) = \left(\frac{1+e^{-i\omega}}{2}\right)^4 (-1+4e^{-i\omega}-e^{-2i\omega}).$$

Roughly speaking, the power $(1+e^{-i\omega})^4$ makes P a good lowpass filter. The final factor is needed to produce zeros in the first and fifth coefficients of P.

The PR condition applies to the product P and not the separate filters H_0 and H_1. So any other factorization of $P(e^{i\omega})$ into $H_1(-e^{i\omega})H_0(e^{i\omega})$ gives perfect reconstruction. Here are two other analysis-synthesis pairs that share the same product $P(e^{i\omega})$:

Biorthogonal
6/2

$$\begin{aligned} h_0 &= \frac{1}{16}(-1,1,8,8,1,-1) \quad & f_0 &= (1,1) \\ h_1 &= \frac{1}{2}(1,-1) & f_1 &= \frac{1}{8}(-1,1,-8,8,-1,1) \end{aligned}$$

Orthogonal
4/4

$$\begin{aligned} h_0 &= \frac{1}{8}\left(1+\sqrt{3},3+\sqrt{3},3-\sqrt{3},1-\sqrt{3}\right) \\ f_0 &= \frac{1}{4}\left(1-\sqrt{3},3-\sqrt{3},3+\sqrt{3},1+\sqrt{3}\right) \\ h_1 &= \frac{1}{8}\left(1-\sqrt{3},-3+\sqrt{3},3+\sqrt{3},-1-\sqrt{3}\right) \\ f_1 &= \frac{1}{4}\left(-1-\sqrt{3},3+\sqrt{3},-3+\sqrt{3},1-\sqrt{3}\right) \end{aligned}$$

The second one is *orthogonal* because the synthesis filters are just *transposes* (time-reversed flips) of the analysis filters. When the inverse is the same as the transpose, a matrix or a transform or a filter bank is called orthogonal. If we wrote out the infinite matrix H_{bank} for this analysis pair, we would discover that F_{bank} is essentially the transpose. (There will be a factor of 2 and also a shift by 3 diagonals to make the matrix lower triangular and causal again. These shifts are responsible for the $\ell = 3$ system delays.)

The algebra of PR filter banks was developed by Smith-Barnwell and iteration of those filter banks led to wavelets in the 1980s. The great paper of Ingrid Daubechies [3] gave the theory of orthogonal wavelets, and the first wavelets she constructed came from the coefficients $(1+\sqrt{3},3+\sqrt{3},3-\sqrt{3},1-\sqrt{3})$ given above.

The other filter banks, not orthogonal, are called *biorthogonal*. Synthesis is biorthogonal to analysis. The rows of H_{bank} are orthogonal to the columns of F_{bank}—except that $(\textit{row } k)\cdot(\textit{column } k) = 1$. This is always true for a matrix and its inverse:

$$HF = I \quad \text{means} \quad (\textit{row } j \text{ of } H)\cdot(\textit{column } k \text{ of } F) = \delta(j-k)\,.$$

The rows of H and columns of H^{-1} are biorthogonal. The analysis and synthesis wavelets that come from infinite iteration will also be biorthogonal.

3.4 Scaling Functions and Refinement Equation

May I write down immediately the most important equation in wavelet theory? It is an equation in continuous time, and its solution $\phi(t)$ is the "**scaling function**." This fundamentally new and fascinating equation is the **refinement equation** or **dilation equation**:

$$\phi(t) = 2\sum_{k=0}^{N} h(k)\phi(2t-k)$$

The factor 2 gives the right side the same integral as the left side, because each $\phi(2t-k)$ has half the area (from compression to $2t$) and $\sum h(k) = 1$. It is the inner factor 2, the one that rescales t to $2t$, that makes this equation so remarkable.

If $\phi(t)$ is a solution, so is any multiple of $\phi(t)$. The usual normalization is $\int \phi(t)\, dt = 1$. That integral extends over all time, $-\infty < t < \infty$, but we actually find that the solution $\phi(t)$ is zero outside the interval $0 \le t < N$. **The scaling function has compact support.** This localization of the function is one of its most useful properties.

Example 1 *Suppose $h(0) = h(1) = \frac{1}{2}$ as in the averaging filter. Then the refinement equation for the scaling function $\phi(t)$ is*

$$\phi(t) = \phi(2t) + \phi(2t-1)\,.$$

The solution is the unit box function, *which equals 1 on the interval $0 \le t < 1$ (and elsewhere $\phi(t) = 0$). The function $\phi(2t)$ on the right side of the equation is a* half-box, *reaching only to $t = \frac{1}{2}$. The function $\phi(2t-1)$ is a* shifted half-box, *reaching from $t = \frac{1}{2}$ to $t = 1$. Together, the two half-boxes make up the whole box, and the refinement equation is satisfied.*

Notice in this example the two key operations of classical wavelet theory:

Dilation of the time scale by 2 or 2^j:
The graph of $\phi(2t)$ is compressed by 2.

Translation of the time scale by 1 or k:
The graph of $\phi(t-k)$ is shifted by k.

The combination of those operations, from t to $2^j t$ to $2^j t - k$, will produce a family of wavelets $w(2^j t - k)$ from a single wavelet $w(t)$. That wavelet comes from the coefficients of the *highpass filter*:

$$w(t) = 2 \sum h_1(k)\phi(2t-k).$$

Solution of the refinement equation. Suppose the lowpass output goes back in as input. The original input is a box function, and we get $\phi^{(i+1)}(t)$ from $\phi^{(i)}(t)$:

$$\textit{Filter and rescale:} \quad \phi^{(i+1)}(t) = 2 \sum_{k=0}^{N} h(k)\phi^{(i)}(2t-k)\,.$$

This is the **cascade algorithm**. If it converges (see [8]), so that $\phi^{(i)}(t)$ approaches $\phi(t)$ as $i \to \infty$, then that limit satisfies the refinement equation.

We can explicitly solve the refinement equation in the frequency domain, for each separate ω. The equation connects the scales t and $2t$, so its Fourier transform connects the frequencies ω and $\omega/2$:

$$\text{The transform of } \phi(2t-k) \text{ is } \int \phi(2t-k)e^{-i\omega t}\, dt = \tfrac{1}{2}e^{-i\omega k/2}\widehat{\phi}(\tfrac{\omega}{2}).$$

Then the refinement equation $\phi(t) = 2\sum h(k)\phi(2t-k)$ transforms to

$$\widehat{\phi}(\omega) = \left(\sum h(k)e^{-i\omega k/2}\right)\widehat{\phi}\left(\frac{\omega}{2}\right) = H\left(\frac{\omega}{2}\right)\widehat{\phi}\left(\frac{\omega}{2}\right)\,.$$

By iteration (which is the cascade algorithm!), the next step gives the factor $H(w/4)$, and $\widehat{\phi}(\omega) = H(\frac{\omega}{2})H(\frac{\omega}{4})\widehat{\phi}(\frac{\omega}{4})$. Infinite iteration gives the *infinite product formula*

$$\widehat{\phi}(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right) \qquad (\text{since } \widehat{\phi}(0) = 1)\,.$$

3.5 Good Basis Functions

This section returns to one of the fundamental ideas of linear algebra — a basis. It often happens that one basis is more suitable than another for a specific application like compression. The whole idea of a *transform* (this paper concentrates on the Fourier transform and wavelet transform) is exactly a change of basis. Each term in a Fourier series or a wavelet series is a basis vector, a sinusoid or a wavelet, times its coefficient. A change of basis gives a *new representation of the same function.*

Remember what it means for the vectors $w_1, w_2, \ldots, w_n$ to be a basis for an n-dimensional space $\mathbf{R}^n$:

1. The w's are linearly independent.
2. The $n \times n$ matrix W with these column vectors is invertible.
3. Every vector x in $\mathbf{R}^n$ can be written in exactly one way as a combination of the w's:

$$x = c_1 w_1 + c_2 w_2 + \cdots + c_n w_n\,.$$

Here is the key point: those coefficients $c_1, \cdots, c_n$ completely describe the vector. In the original basis (standard basis), the coefficients are just the samples $x(1), \ldots, x(n)$. In the new basis of w's, the same x is described by $c_1, \ldots, c_n$. It takes n numbers to describe each vector and it *also* takes a choice of basis:

$$x = \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix}_{\text{standard basis}} \qquad \text{and} \qquad x = \begin{bmatrix} z_0(2) \\ z_1(2) \\ y_1(2) \\ y_1(4) \end{bmatrix}_{\text{Haar basis}}\,.$$

The point of wavelets is that both steps, analysis and synthesis, are executed quickly by filtering and subsampling. Let me extend this to repeat a more general comment on good bases. Orthogonality gives a simple relation between the transform and the inverse transform (they are "transposes" of each other). But I regard two other properties of a basis as more important in practice than orthogonality:

Speed: The coefficients $c_1, \ldots, c_n$ are fast to compute.

Accuracy: A small number of basis vectors and their coefficients can represent the signal very accurately.

For the Fourier and wavelet bases, the speed comes from the FFT and FWT: Fast Fourier Transform and Fast Wavelet Transform. The FWT is exactly the filter bank tree that we just illustrated for the Haar wavelets.

The accuracy depends on the signal! We want to choose a basis appropriate for the class of signals we expect. (It is usually too expensive to choose a basis adapted to each individual signal.) Here is a rough guideline:

For *smooth* signals with no jumps, the *Fourier basis* is hard to beat.
For *piecewise smooth* signals with jumps, a *wavelet basis* can be better.

The clearest example is a step function. The Fourier basis suffers from the *Gibbs phenomenon*: oscillations (ringing) near the jump, and very slow $\frac{1}{n}$ decay of the Fourier coefficients c_n. The wavelet basis is much more localized, and only the wavelets crossing the jump will have large coefficients. Of course the Haar basis could be perfect for a step function, but it has slow decay for a simple linear ramp. Other wavelets of higher order can successfully capture a jump discontinuity in the function or its derivatives.

To emphasize the importance of a good basis (an efficient representation of the data), we will list some important choices. Each example is associated with a major algorithm in applied mathematics.

1. Fourier series (sinusoidal basis)

2. Finite Element Method (piecewise polynomial basis)

3. Spline Interpolation (smooth piecewise polynomials)

4. Radial Basis Functions (functions of the distance $\|x - x_i\|$ to interpolation points x_i in d dimensions)

5. Legendre polynomials, Bessel functions, Hermite functions, ... (these are orthogonal solutions of differential equations)

6. Wavelet series (functions $w(2^j x - k)$ from dilation and translation of a wavelet).

The next good basis to be invented will be another significant step in applied mathematics.

4 Localized Eigenvectors from Widely Spaced Matrix Modifications

This section is about the eigenvalues and eigenvectors of familiar structured matrices, after changes in a small number of entries. The actual changes need not be small, so we refer to them as modifications rather than perturbations. The *number* of changes is small relative to the size of the matrix, because the modifications are required to be "widely spaced". They occur in entries that are far apart. They produce new eigenvectors that are localized near the components that correspond to the modified rows. By knowing the approximate form of those eigenvectors, we also determine a very close (and simple) approximation to the eigenvalues.

Imagine a large number of nodes around a circle. Edges go only to the two neighbors of every node. Each row of the adjacency matrix A of this cyclic graph has two 1's. The matrix is a circulant with 1's on the first subdiagonal and superdiagonal, coming from the neighbors to the left and right. Now add a few edges going "across" the circle, so that the nodes involved are widely spaced. The modified graph has an adjacency matrix (symmetric if the added edges are undirected, but this is not required) with 1 in the i,j entry when an edge connects node i to node j. A typical example of our work is to find the "new" eigenvalues and eigenvectors of this modified matrix.

The author mentioned in *SIAM News* (April 2000) the simplest case of this example. Only one undirected edge crosses the circle, from node i to a distant node j. This added edge modifies A by $a_{ij} = a_{ji} = 1$, in other words by a widely spaced submatrix with entries from

$$B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The two new 1's in the modified matrix are far from the main diagonal. *The two new eigenvalues are almost exactly* $\sqrt{5}$ *and* $-\sqrt{5}$. The corresponding eigenvectors show a sum or difference of two spikes, as in Figure 1, centered at the positions i and j connected by the "shortcut edge". The

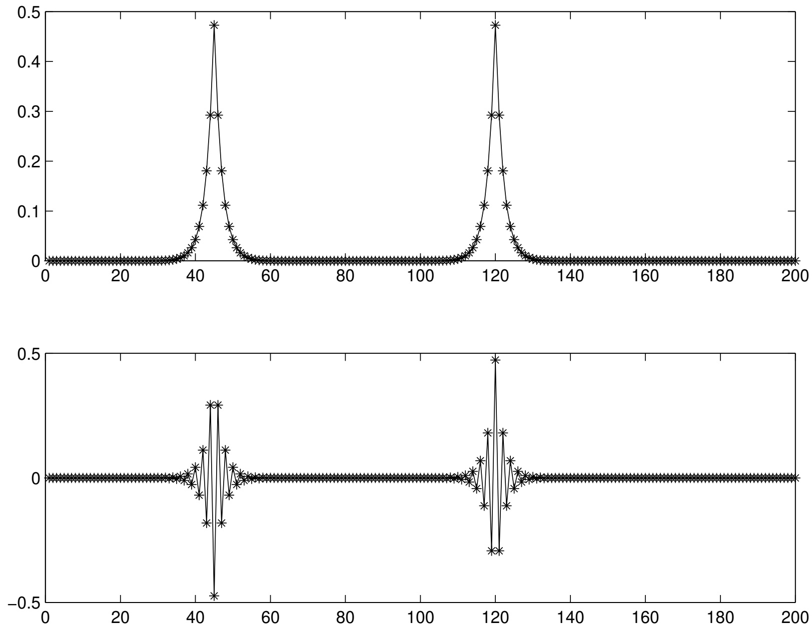


Figure 1: The eigenvectors for the maximal and minimal eigenvalues of the adjacency matrix of a 200-node cycle with an edge added between nodes $i = 45$ and $j = 120$

remaining eigenvalues stay in the interval $[-2, 2]$ that contains all eigenvalues of the original A. Their eigenvectors still oscillate like the original eigenvectors, but orthogonality to the new ones produces the pinching that is illustrated by Figure 2.

This brief report in *SIAM News* brought suggested proofs from three friends, Beresford Parlett and Bill Trench and Jackie Shen. All four approaches are different! Shen connected the problem to the theory of perturbed Schrödinger operators, and we believe that our work can be seen as a small contribution (possibly not new) to that established theory. We stay with this example, to find the following formula linking the (nearly exact) new eigenvalues λ to the eigenvalues μ of B:

$$\lambda = \operatorname{sign}(\mu)\ \sqrt{4 + \mu^2}$$

The rank two perturbation from one undirected edge and the matrix B above has $\mu = 1$ and -1, confirming that $\lambda = \sqrt{5}$ and $-\sqrt{5}$. In the two localized eigenvectors, the heights of the "spikes" are given by the eigenvectors $(1, 1)$ and $(-1, 1)$ of B. We also determine the ratio t between neighboring

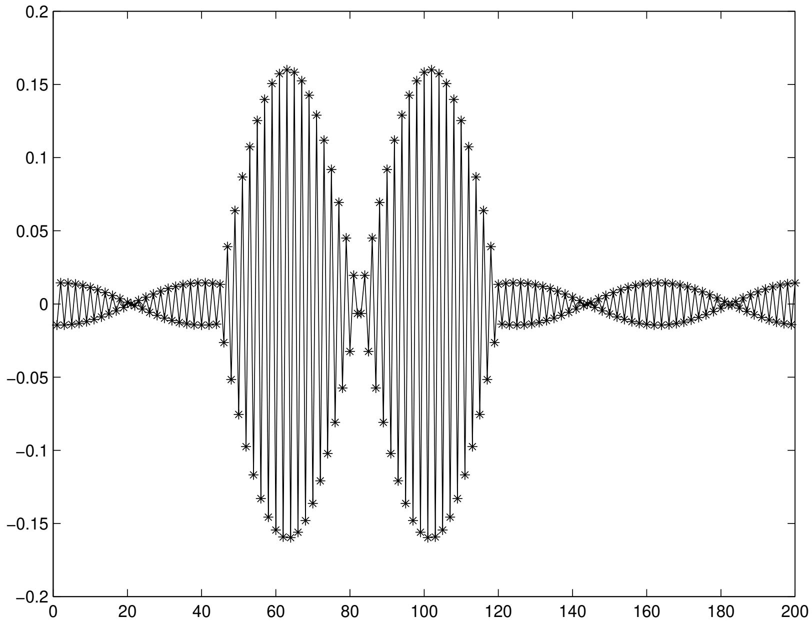


Figure 2: A typical eigenvector corresponding to an eigenvalue in the range of $[-2, 2]$ for the perturbed adjacency matrix.

entries near the spikes (a smaller t means a sharper spike and a more localized eigenvector). This pattern extends to any widely spaced modification by a nonsingular B.

Later sections of the full paper [10] will extend the theory beyond the circle of nodes and its particular adjacency matrix A. We mention that an infinite string of nodes would give the same results, or even a finite string with a tridiagonal matrix A, provided the modifications occur far from the ends of the string (the first and last rows of A). The Laplacian matrix of a graph (a circle or a tree or an N-dimensional grid) is another important source of examples.

4.1 The Model Problem

We start with an infinite line of nodes (the graph has a node at every integer). Its adjacency matrix A has 1's on the first subdiagonal and first superdiagonal: $a_{i,i-1} = a_{i,i+1} = 1$ for $-\infty < i < \infty$. The modification of A will be governed by an M by M matrix B, which need not be symmetric. We choose M widely spaced indices $r_1 < ... < r_M$; the differences between these indices all exceed a number $L >> 1$. Then the i, j entry of B is added to the r_i, r_j entry of A. By a terrible abuse of notation, we call the modified matrix $A + B$. Our problem is to estimate the "new" eigenvalues and eigenvectors after the modification:

$$(A + B)x = \lambda x$$

The key is that we expect each eigenvector x to be a sum of M spikes. For a given eigenvector, suppose the spike centered at the r_k entry of x has height h_k. The "spike ratio" between neighboring entries is denoted by t. Then the j-th component of this eigenvector has the form

$$x_j = \sum_{k=1}^{M} t^{|j-r_k|} h_k$$

When $j = r_k$, the k-th term reduces to h_k as desired. The ratio t will be different for different eigenvectors (t will depend on λ).

Now substitute this form for x into $(A+B)x = \lambda x$. We distinguish the special rows $j = r_1, ..., r_M$ from the other rows (ordinary unmodified rows). In those ordinary rows there is no contribution from B. The matrix A has 1's to the left and right of the diagonal, which produce an extra factor of t and $1/t$ in every spike. The eigenvalue equation in the ordinary rows becomes:

$$(t + \frac{1}{t})x_j = \lambda x_j$$

In each special row $j = r_k$, where B has an effect, the equation is

$$2th_k + (Bh)_k = \lambda h_k + O(t^L)$$

You see the change. For the k-th spike, centered on this special row, both 1's in A produce a factor t (thus $2t$). All the other spikes in x are of order t^L in this entry, because the special rows are far apart. For the same reason x_j on the right side equals $h_k + O(t^L)$.

Suppose we ignore the error $O(t^L)$. Then the previous equation says that the vector h of spike heights is an eigenvector of B. If that eigenvector has an eigenvalue μ, we have

$$2t + \mu = \lambda = t + \frac{1}{t}$$

Compare the left side with the right side, to find $t + \mu = 1/t$. This quadratic equation yields

$$t = \frac{1}{2}(-\mu \pm \sqrt{4 + \mu^2})$$

Choose the plus-minus sign to agree with the sign of μ, so that t also has that sign and $|t| < 1$. Then substitute t into λ to find

$$\lambda = 2t + \mu = \text{sign}(\mu)\ \sqrt{4 + \mu^2}$$

This is the (approximate) relation between the new eigenvalue λ of $A+B$ and the eigenvalue μ of B. We want to prove that the error in λ is of the same order $O(t^L)$ as the terms that were dropped.

Theorem 1 *If μ is a non-repeated non-zero eigenvalue of the M by M matrix B, with eigenvector h of norm one, then λ and x are within $O(t^L)$ of an exact eigenvalue-eigenvector pair for the (infinite) modified matrix $A + B$.*

This applies to every $x > 0$, including our x above. We need to do all eigenvalues and to allow negative entries in B.

Example 1 Suppose we change only a single entry on the main diagonal from 0 to 1. The modifying matrix is just $B = [1]$. For the (infinite) modified matrix, the localized eigenvector is exact! The eigenvalue is $\sqrt{5}$ and the spike ratio is the golden mean $t = 1/2(-1+\sqrt{5})$. If the single 1 is the $(0,0)$ entry, the j-th component of the eigenvector is $t^{|j|}$.

For a finite matrix, this eigenvalue-eigenvector pair is only approximate. The approximation is very accurate as soon as the modified entry moves a few positions from the ends of the diagonal.

Example 2 Connect three widely spaced nodes i,j,k by three undirected edges. In this case the modifying matrix is

$$B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Its eigenvalues are $\mu = 2, -1, -1$. The eigenvalues of the large matrix $A + B$ are approximately $\lambda = \sqrt{4+2^2} = \sqrt{8}$ and $\lambda = -\sqrt{4+(-1)^2} = -\sqrt{5}$ (twice).

The eigenvector $h = (1,1,1)$ of the small matrix is correctly reflected in the eigenvector of $A+B$ for $\lambda = \sqrt{8}$. It is very nearly a sum of three equal spikes.

The other eigenvalue $\mu = -1$ is repeated. The eigenvalue $\lambda = -\sqrt{5}$ is nearly but not exactly repeated. Theorem 1 cannot apply as it stands to the eigenvectors, because the small matrix has a plane of eigenvectors for $\mu = -1$. Since $\lambda = -\sqrt{5}$ is not exactly repeated, there is no corresponding plane for $A + B$.

To extend this example, suppose the modification adds a complete graph on M nodes to the starting graph (which is still an infinite line of nodes connected only to their neighbors). The M by M matrix B has 0's on the diagonal and 1's everywhere else. Its largest eigenvalue $\mu = M - 1$ has eigenvector $h = (1, 1, ..., 1)$. Then the modified matrix $A + B$ has largest eigenvalue $\lambda = \sqrt{4 + (M-1)^2}$ with $x =$ sum of equal spikes.

By including three separate applications of matrices in one lecture, we hope that every reader will have found an IMACS matrix to like.

References

[1] Y. Chen, S. Oraintara and T. Nguyen, Video Compression Using Integer DCT, *IEEE Int. Conf. on Image Processing* (2000).

[2] Y. Chen, S. Oraintara and T. Nguyen, Integer Discrete Cosine Transform (IntDCT), submitted *IEEE Trans. Signal Processing* (2000). See also Integer Fast Fourier Transform (IntFFT), preprint (2000).

[3] I. Daubechies, Orthonormal Bases of Compactly Supported Wavelets, *Comm. Pure Appl. Math.* **41** (1988) 909–996.

[4] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, 1993.

[5] R. DeVore and B. Lucier, Wavelets, *Acta Numerica* **1** (1991) 1–56.

[6] K.R. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantages, Applications*, Academic Press (1990).

[7] G. Strang, The Discrete Cosine Transform, *SIAM Review* **41** (1999) 135–147.

[8] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press (1996).

[9] G. Strang, *Signal processing for everyone*, Computational Mathematics Driven by Industrial Problems, CIME short course (1999) Springer Lecture Notes in Mathematics, to appear.

[10] G. Strang, X. Liu and S. Ott, Localized Eigenvectors from Widely Spaced Matrix Modifications, to appear (2000).

[11] T. D. Tran, A Fast Multiplierless Block Transform for Image and Video Compression, *IEEE Int. Conf. on Image Processing*, Kobe, Japan (1999).

[12] T. D. Tran, The BinDCT: Fast Multiplierless Approximation of the DCT, *IEEE Signal Processing Letters*, **7**, (2000) 141–145.

[13] M. Unser and T. Blu, Fractional splines and wavelets, *SIAM Review* **42** (2000).