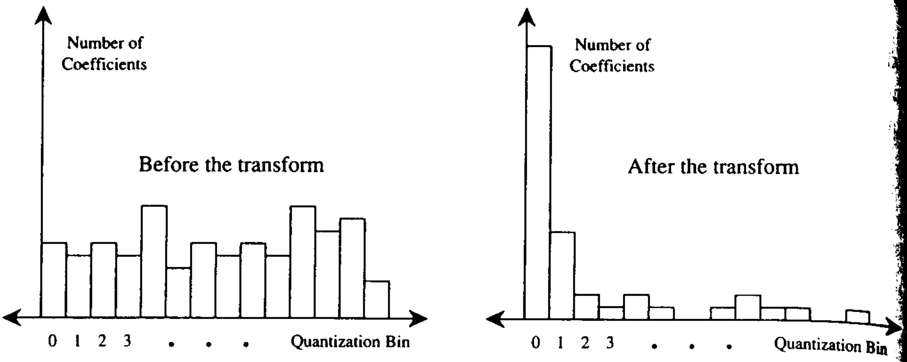


Figure 11.11: Number of coefficients per quantization bin, before and after an effective transform step.

functions in order to concentrate quantization error where it will be least visible. This is easy to do with a custom subband coding scheme, since synthesis functions within a subband are characterized by their spatial frequencies which in turn, to a large extent, determine their visibility. We can use fewer bins and coarser quantization in high-frequency subbands, where errors are less visible. But even for nonhuman audiences, such as automatic image detection, classification, and recognition machines, the quantization can be adjusted to preserve or even emphasize image features which are known to be important. In the FBI WSQ case, subbands which describe pore frequencies and ridge spacings are quantized more finely to assist subsequent automatic fingerprint identification systems, as well as human fingerprint examiners. For individual best-basis compression, it may be necessary to append a quantization table to the compressed image data. The JPEG standard [1] also allows individualizing the quantization table for each image, to improve its performance on exceptional images.

We may also vary the quantization by assigning different numbers of bins to different groups of coefficients, in proportion to how much variance those coefficients exhibit. We may compute the variance empirically for a group of coefficients from an individual image, such as all coefficients within one subband, or we may estimate the variance of each coefficient from a model of the class of images we plan to compress. In the first case, we must transmit a *bit allocation* table along with the compressed image, so we should take fairly large groups of coefficients to keep down the overhead. In the second case, the bit allocation scheme will be fixed and known to both transmitter and receiver, so there will be no overhead.

Redundancy removal

Our goal is to reduce the number of bits we must transmit or store, so after quantization we should code the sequence of bin indices for greatest efficiency. The third block ("Remove redundancy") replaces the stream of small integers with a more efficient alphabet of variable-length characters. In this alphabet the frequently occurring letters (like "0") are represented more compactly than rare letters.

The Huffman algorithm ([104], p.39ff) or one of its variants may be used for entropy coding. These treat the input as a sequence of independent Bernoulli trials with a known or empirically determined probability distribution function; they chose a new alphabet so as to minimize the expected length of the output. Our assumption is that the transform step has decorrelated the coefficients to the point that they are hard to distinguish from a sequence of independent random variables. This assumption is not valid if there are long runs of zeroes, but that can be fixed by introducing special code words for common run lengths. Such a technique is part of the FBI WSQ algorithm definition, since certain subbands are expected to be quantized to zero almost entirely.

11.2 Fast approximate factor analysis

There are many names for the algorithms described below: *factor analysis*, *principal component analysis*, *singular value decomposition*, and the *Karhunen–Loève transform* are some of the more common ones. It has both an algebraic interpretation, finding the eigenvalues of a matrix, and an analytic interpretation, finding the minimum of a cost function over the set of orthogonal matrices. The first interpretation is best when we need an exact solution, but it has high arithmetic complexity. The minimization point of view leads to a lower complexity approximate algorithm which gets close to the minimum but does not attain it.

Principal orthogonal decomposition can be used to solve two related problems: distinguishing elements from a collection by making d measurements, and inverting a complicated map from a p-parameter configuration space to a d-dimensional measurement space. In the case where d is more than 1000 or so, the classical $O(d^3)$ singular value decomposition algorithm becomes very costly, but it can be replaced with an approximate best basis method that has complexity $O(d^2 \log d)$. This can be used to compute an approximate Jacobian for a complicated map from $\mathbf{R}^p \rightarrow \mathbf{R}^d$ in the case $p \ll d$.

Consider the problem of most efficiently distinguishing elements from a collection by making d measurements. In general, we will need all d measured values to fully specify an element. However, it is possible to use less information if some of the measurements are correlated. For example, if the objects are parametrized by a small number $p \ll d$ of parameters, then the d measurements should separate them in a redundant fashion. That is, we can change basis locally in the d-dimensional measurement space to find just p combinations of measurements which change with the p parameters. This idea works even if there are many parameters but only p of them are relatively important.

Note the resemblance between the problem of distinguishing elements and the problem of inverting a complicated map from $\mathbf{R}^p$ to $\mathbf{R}^d$. In the first problem, we must find a discrete object given its description in $\mathbf{R}^d$. In the second, we must find the parameters in $\mathbf{R}^p$ from the description in $\mathbf{R}^d$. These problems are identical if the collection of objects is produced by evaluating the complicated map at discrete grid points in $\mathbf{R}^p$.

The combinations of measurements which root out the underlying parameters are called *principal (orthogonal) components* or *factors*; they have a precise meaning, and the well-known and well-behaved method of *singular value decomposition* or *SVD* produces them with arbitrary accuracy. However, SVD has a complexity that is asymptotically $O(d^3)$, making it impractical for problems larger than $d \approx 1000$. In this section, we will describe the classical principal factor algorithm, then give a lower complexity *approximate principal factor algorithm*. Finally, we will give example applications of the approximate algorithm to the two mentioned problems.

The Karhunen–Loève transform

The *principal orthogonal* or *Karhunen–Loève coordinates* for an ensemble $X = \{X_n \in \mathbf{R}^d : n = 1, \ldots, N\}$ correspond to the choice of axes in $\mathbf{R}^d$ which minimizes the volume of the variance ellipsoid defined by Figure 11.1. These axes should be the eigenvectors of the autocovariance matrix $A = E(\tilde{X} \otimes \tilde{X})$ of the ensemble, defined by Equation 11.8. Since this matrix is positive semidefinite, we can find an orthonormal basis for $\mathbf{R}^d$ consisting of eigenvectors, and these eigenvectors will have nonnegative real eigenvalues. Thus we are assured that the *Karhunen–Loève* basis exists for the ensemble X; this is the orthonormal basis of eigenvectors of A.

The Karhunen–Loève basis eigenvectors are also called *principal orthogonal components* or *principal factors*, and computing them for a given ensemble X is also called *factor analysis*. Since the autocovariance matrix for the Karhunen–Loève eigenvectors is diagonal, it follows that the Karhunen–Loève coordinates of the vectors in the sample space X are uncorrelated random variables. Let us denote these basis eigenvectors by $\{Y_n : n = 1, \ldots, N\}$, and let us denote by K the $d \times d$ matrix whose columns are the vectors $Y_1, \ldots, Y_N$. The adjoint of K, or K^*, is the matrix which changes from the standard coordinates into Karhunen–Loève coordinates;

this map is called the *Karhunen–Loève transform.*

Unfortunately, finding these eigenvectors requires diagonalizing a matrix of order d, which has complexity $O(d^3)$. In addition, even after already computing the Karhunen–Loève eigenvectors of an ensemble, updating the basis with some extra random vectors will cost an additional $O(d^3)$ operations since it requires another diagonalization.

Such a high order of complexity imposes a ceiling on the size of problem we can do by this method. In many cases of interest, d is very large and X spans $\mathbf{R}^d$, implying $N \geq d$. Thus even to find the coefficients of the autocovariance matrix requires $O(d^3)$ operations. At the present time, we are limited to $d \leq 10^3$ if we must use common desk top computing equipment, and $d \leq 10^4$ for the very most powerful computers.

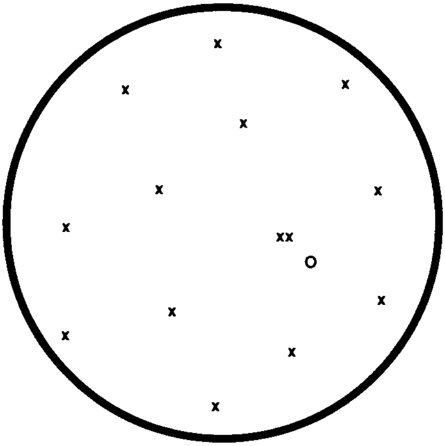
So we shall take another perspective. We shall pose the problem of finding the Karhunen–Loève eigenvectors as an optimization over the set of orthogonal transformations of the original ensemble X. The quantity to be maximized will be a *transform coding gain*, or the amount of compression we achieve by using another basis to represent the ensemble. This gain is increased if we can decrease the volume of the variance ellipsoid for the signal ensemble, and is thus maximized by the Karhunen–Loève transform. We get an approximate Karhunen–Loève transform from any efficient transform which significantly increases the gain, even if does not attain the maximum.

Alternatively, we could introduce a distance function on the set of orthonormal bases for $\mathbf{R}^d$, treat the Karhunen–Loève basis as a distinguished point in this set, and then try to find an efficiently computable basis which is close to the Karhunen–Loève basis.

A metric on the orthogonal matrices

Consider Figure 11.12, which schematically depicts all the orthonormal bases of $\mathbf{R}^d$. These can be identified with certain transformations of $\mathbf{R}^d$, namely the orthogonal $d \times d$ matrices. The points marked by "x" represent bases in some library of fast transforms. The point marked "o" represents the optimal, or Karhunen–Loève basis, for a given ensemble of vectors. The point marked "xx" represents the fast transform closest to the Karhunen–Loève bases.

Let U be an orthogonal $d \times d$ matrix, and write $Y = UX$ to signify that $Y_n = UX_n$ for each $n = 1, 2, \ldots, N$. Since U is linear, $E(Y) = E(UX) = UE(X)$, which will be zero if we started with $E(X) = 0$. Since U is orthogonal, it preserves sums of squares, so $\mathrm{Var}(X) = \mathrm{Var}(Y)$. Using wavelet packets or adapted local trigonometric functions, it is possible to build a library of more than 2^d fast transforms U of $\mathbf{R}^d$



x = fast transform bases; o = Karhunen–Loève basis; xx = best fast basis.

Figure 11.12: Orthonormal bases for $\mathbf{R}^d$.

to use for the "x" points. We will illustrate the construction using wavelet packets. These transforms are arranged in a structure that permits us to search for the one closest to the "o" point in $O(d \log d)$ operations. We will use a notion of closeness that is derived from the function minimized by the Karhunen–Loève transform.

As in [61], define the *transform coding gain* for an orthogonal matrix by the formula

$$G_{TC}(U) = \mathrm{Var}(UX)/\exp H(UX), \quad \text{where} \quad H(X) = \frac{1}{d}\sum_{i=1}^{d} \log \sigma(X)(i). \quad (11.11)$$

From this formula we see that $G_{TC}(UX)$ is maximized when $H(UX)$ is minimized. The quantity H has various interpretations. It is the entropy of the direct sum of d independent Gaussian random variables with variances $\sigma(X)(i)$, $i = 1, \ldots, d$. It is also equal to the logarithm of the volume of the variance ellipsoid (if we add $\log \omega_d$), so we see that minimizing $H(UX)$ or maximizing $G_{TC}(UX)$ is equivalent to minimizing the volume of the variance ellipsoid for the ensemble UX over all orthogonal matrices U.

Since the Karhunen–Loève transform is a global minimum for H, we will therefore say that the best approximation to the Karhunen–Loève transform from a library $\mathcal{U}$ of orthogonal matrices is the minimum of $H(UX)$ with the constraint $U \in \mathcal{U}$. We can define the *approximate factor analysis algorithm* to be the search through a library of orthonormal bases for the one whose H is closest to that of the Karhunen–Loève basis. If the library of bases is organized to facilitate a fast search, we will dare to call the result a *fast approximate Karhunen–Loève algorithm.*

The "closeness" of a basis U to the Karhunen–Loève basis K can be measured by computing the transform coding gain of U and subtracting that of K. This give us a *transform coding gain metric*:

$$dist_X(U,V) = |H(UX) - H(VX)|.$$

Notice that we get a different metric for each ensemble X. This is a degenerate metric on the orthogonal group, since it gives a distance of zero between bases which have the same transform coding gain for X. However, this technical point can be overcome by constructing a topological quotient space in which such bases are considered equivalent.

A minimum for $H(VX)$ is the Karhunen–Loève basis $V = K$, so minimizing $H(UX)$ over fast transforms U is equivalent to minimizing $dist_X(U,K)$: it finds the closest fast transform for this ensemble in the transform coding sense.

The entropy metric

If U is a $d \times d$ orthogonal matrix, then we can define its *entropy* to be

$$\mathcal{H}(U) = -\sum_{i=1}^{d}\sum_{j=1}^{d} |U(i,j)|^2 \log\left(|U(i,j)|^2\right). \qquad (11.12)$$

We have the two inequalities

$$0 \le \mathcal{H}(U) \le d \log d, \qquad (11.13)$$

which are elementary consequences of properties of the entropy of a probability distribution function and the fact that the squares of the elements of each of the d columns of U sum up to one. $\mathcal{H}$ is a functional on $\mathbf{O}(d)$, the compact Lie group of orthogonal linear transformations of $\mathbf{R}^d$. $\mathcal{H}$ can be used to define a distance function.

The *entropy metric* on the orthogonal group is the function

$$dist(U,V) = dist_{\mathbf{O}}(U,V) \overset{\text{def}}{=} \sqrt{\mathcal{H}(U^*V)}$$

defined for $U, V \in \mathbf{O}(d)$. This function adds up the entropy of the coordinates of the basis vectors for V when they are expanded in the basis U.

Note that $\mathcal{H}(I) = 0$ for the identity matrix I, and that $\mathcal{H}(U^*) = \mathcal{H}(U)$. These two properties of $\mathcal{H}$ imply that $dist(U,U) = 0$ and $dist(U,V) = dist(V,U)$. We also have the triangle inequality: $dist(U,V) \le dist(U,W) + dist(W,V)$.

11.2.1 The approximate KL transform

We now turn to the large library of rapidly computable orthonormal wavelet packet bases, constructed to take advantage of the rapid growth of the number of subtrees of a binary tree. We will fix our attention on wavelet packets; it is relatively easy to generalize this example to the other adapted wavelet approximate Karhunen–Loève expansions.

Let H and G be conjugate orthogonal QFs. Starting with a signal $x = \{x(j) : j = 0, 1, \ldots, d-1\}$ of $d = M2^L$ samples, we recursively apply H and G a total of L times to get a complete wavelet packet analysis down to level L. We arrange the resulting sequences into a binary tree, which now contains very many basis subsets. Each graph, for example, gives a different orthonormal basis.

Suppose that the sequence of coefficients in block f of level s of the tree for the signal X_n is called $\{\lambda_{sf}^{(n)}(p)\}$. We then sum the coefficients of the N *signal* trees into two *accumulator* trees:

- The tree of *means*, which contains $\sum_{n=0}^{N-1} \lambda_{sf}^{(n)}(p)$ in location p of block f at level s, and so on;

- The tree of *squares*, which contains $\sum_{n=0}^{N-1} \left[\lambda_{sf}^{(n)}(p)\right]^2$ in location p of block f at level s, and so on.

Computing all the coefficients of all the blocks in an L-level tree starting from d samples takes $O(dL) = O(d \log d)$ operations per random vector, for a total of $O(Nd \log d)$ operations. After we do this for all the random vectors X_n in the ensemble X, we can produce the binary tree of *variances* by using Equation 11.2: at index p of block f at level s, for example, it contains

$$\sigma_{sf}^2(X)(p) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=0}^{N-1} \left[\lambda_{sf}^{(n)}(p)\right]^2 - \left[\frac{1}{N} \sum_{n=0}^{N-1} \lambda_{sf}^{(n)}(p)\right]^2 . \tag{11.14}$$

This is the variance of the wavelet packet coefficient $\lambda_{sf}(p)$ defined by the filters H, G. Forming this tree takes an additional $O(d \log d)$ operations.

The tree of variances may now be searched for the graph basis which maximizes transform coding gain. We use the $\log \ell^2$ information cost function:

$$\mathcal{H}(V_{jn}) \stackrel{\text{def}}{=} \sum_{k=0}^{d/2^j - 1} \log \sigma_{jn}(X)(k). \tag{11.15}$$

Notice that each block is examined twice during the best basis search: once when it is a parent and once when it is a child. This means that the search requires as many comparison operations as there are blocks in the tree, which is $O(d)$. Computing $\mathcal{H}$ costs no more than a fixed number of arithmetic operations per coefficient in the tree, which is $O(d \log d)$. Thus the total cost of the search is $O(d \log d)$.

Let U be the best basis in the variance tree; call it the *joint best basis* for the ensemble X, in the wavelet packet library. We can also denote by U the $d \times d$ orthogonal matrix which corresponds to the orthogonal basis. Abusing notation, we write $\{U_i \in \mathbf{R}^d : i = 1, \ldots, d\}$ for the rows of U. We may suppose that these rows are numbered so that $\sigma(UX)$ is in decreasing order; this can be done by sorting all the d coefficients in all the blocks $V \in U$ into decreasing order, which can be done in $O(d \log d)$ operations.

If we fix $\epsilon > 0$ and let d' be the smallest integer such that

$$\sum_{n=1}^{d'} \sigma(UX)(n) \geq (1-\epsilon)\mathrm{Var}(X),$$

then the projection of X onto the row span of $U' = \{U_1 \ldots U_{d'}\}$ contains $1-\epsilon$ of the total variance of the ensemble X. Call this projected ensemble X'. The d' row vectors of U' are already a good basis for the ensemble X', but they may be further decorrelated by Karhunen–Loève factor analysis. The row vectors of U' are just $U'_i = U_i$ for $1 \leq i \leq d'$, and the autocovariance matrix for this new collection is given by

$$M'_{ij} = \frac{1}{N}\sum_{n=1}^{N} U'_i \tilde{X}'_n U'_j \tilde{X}'_n.$$

Here $\tilde{X}'_n = X'_n - E(X')$ is a vector in $\mathbf{R}^{d'}$, and $E(\tilde{X}') = 0$. ? Thus M' is a $d' \times d'$ matrix and can be diagonalized in $O(d'^3)$ operations. Let K' be the matrix of eigenvectors of M'. Then K'^* changes from the joint best basis coordinates (calculated from the standard coordinates by U') into coordinates with respect to these decorrelated eigenvectors. We may thus call the composition K'^*U' the *approximate Karhunen–Loève transform* with relative variance error ϵ.

Complexity

The approximate Karhunen–Loève or joint best basis algorithm is fast because we expect that even for small ϵ we will obtain $d' \ll d$. To count operations in the worst case, we make the following assumptions:

Assumptions for computing complexity

1. There are N random vectors;
2. They belong to a d-dimensional parameter space $\mathbf{R}^d$;
3. The autocovariance matrix has full rank, so $N \geq d$.

There are five parts to the algorithm to build the tree of variances and search it for the joint best basis, which *trains* the algorithm to choose a particular decomposition:

Finding the approximate Karhunen–Loève basis

- Expanding N vectors $\{X_n \in \mathbf{R}^d : n = 1, 2, \ldots, N\}$ into wavelet packet coefficients: $O(Nd \log d)$;
- Summing squares into the variance tree: $O(d \log d)$;
- Searching the variance tree for a best basis: $O(d + d \log d)$;
- Sorting the best basis vectors into decreasing order: $O(d \log d)$;
- Diagonalizing the autocovariance matrix of the top d' best basis vectors: $O(d'^3)$.

Adding these up, we see that the total complexity of constructing the approximate Karhunen–Loève transform K'^*U', is $O(Nd \log d + d'^3)$. This compares favorably with the complexity $O(Nd^2 + d^3)$ of the full Karhunen–Loève expansion, since we expect $d' \ll d$.

Depending on circumstances, the last step $U' \mapsto K'^*U'$ may not be necessary, since a large reduction in the number of parameters is already achieved by transforming into the orthonormal basis determined by U'. This reduces the complexity to $O(Nd \log d)$, with the penalty being less decorrelation of the factors.

After training to learn the joint best basis, the algorithm *classifies* new vectors by expanding them in the chosen basis and separating them by their principal components:

The approximate Karhunen–Loève transform of one vector

- Computing the wavelet packet coefficients of one vector: $O(d \log d)$.
- Applying the $d' \times d'$ matrix K'^*: $O(d'^2)$.

Since $d' \ll d$, this estimate compares favorably with the complexity of applying the full Karhunen–Loève transform to a vector, which is $O(d^2)$. Further savings are possible, notably because only a small fraction $d'' \ll d'$ of the Karhunen–Loève singular vectors are needed to capture almost all of the original ensemble variance. Hence we can take K'' to be just the first d'' of the columns of K', and then the total complexity of applying $K''^{*}U'$ is bounded by $O(d \log d + d''d')$.

If we expect to update the Karhunen–Loève basis, then we might also expect to update the average vector and the average value of each coordinate in the library of bases, as well as the variance of the ensemble. But since we keep a sum-of-squares tree and a sum-of-coefficients or means tree rather than a variance tree, each additional random vector just contributes its wavelet packet coordinates into the means tree and the squares of its coordinates into the sum-of-squares tree. The variance tree is updated at the end using the correct new means. This results in the following update complexity:

Updating the approximate Karhunen–Loève basis

- Expanding one vector into wavelet packet coefficients: $O(d \log d)$.
- Adding the coefficients into the means tree: $O(d \log d)$.
- Adding the squared coefficients into the squares tree: $O(d \log d)$.
- Forming the variance tree and computing the new information costs: $O(d \log d)$.
- Searching the variance tree for the joint best basis: $O(d + d \log d)$.

So one new vector costs $O(d \log d)$, and updating the basis with $N > 1$ new vectors costs $O(Nd \log d)$.

11.2.2 Classification in large data sets

The Karhunen–Loève transform can be used to reparametrize a problem so as to extract prominent features with the fewest measurements. When the number of measurements is huge, the fast approximate algorithm must be used at least as a "front end" to reduce the complexity of the SVD portion of finding the Karhunen Loève basis.

We list a few examples to give some indication of the size of a problem that can be treated by the approximate method on typical table top computing equipment.

The rogues' gallery problem

The "rogues' gallery" problem is to identify a face from among a collection of faces. This problem was first suggested to me by Lawrence Sirovich, who also provided the data used in this experiment. The random vectors were several thousand digitized 128×128 pixel, eight-bit gray-scale pictures of Brown University students, so $d =$

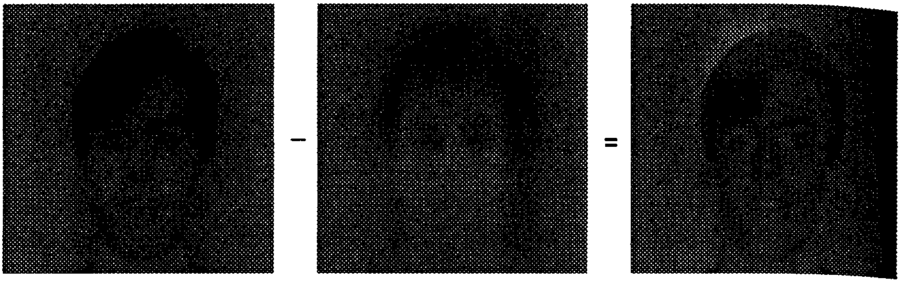


Figure 11.13: Face, minus the average face, yields a caricature.

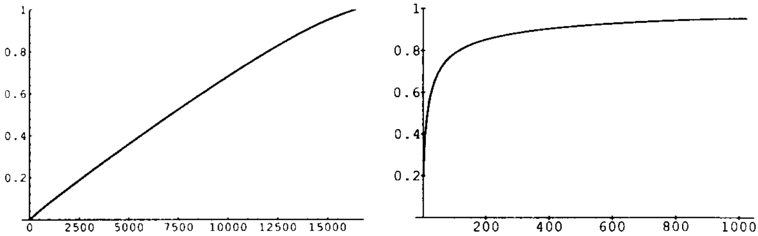


Figure 11.14: Accumulation of variance in the original basis and the joint best basis.

$128^2 = 16,384$. These were initially normalized with the pupils impaled on two fixed points near the center of the image. In [100, 62], a supercomputer was used to compute the Karhunen–Loève transform either of the complete set of pixels or else of an oval subset centered about the eyes. In the following, we will follow Sirovich's methodology and nomenclature, only we will replace the Karhunen–Loève transform with the lower complexity approximate algorithm.

For the experiment described below, we start with a more limited data set containing 143 pictures. Since the ensemble was fixed, we could subtract the average vector at the outset. Thus we transformed the data to floating point numbers, computed average values for the pixels, and then subtracted the average from each pixel to obtain "caricatures," or deviations from the average. Figure 11.13 is one of these caricatures.

The left graph in Figure 11.14 shows how the variance accumulates pixel by pixel, with the pixels sorted into decreasing order of variance.

Each caricature was treated as a picture and expanded into two-dimensional wavelet packets. The squares of the amplitudes were summed into a tree of variances, which was then searched for the joint best basis. In the joint best basis,

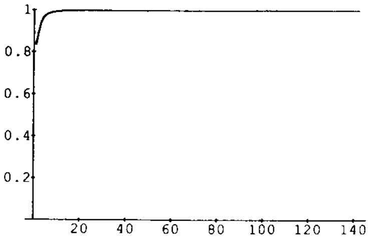


Figure 11.15: Accumulation of variance in the approximate Karhunen–Loève basis.

400 coordinates (of 16,384) contained more than 90 percent of the variance of the ensemble.

The right graph in Figure 11.14 shows the accumulation of total variance on the first d' coordinates in the joint best basis, sorted in decreasing order, as a fraction of the total variance of the ensemble, for $1 \leq d' \leq 1000$. Using 1000 parameters captures more than 95 percent of the ensemble variance, but requires somewhat more computer power than is readily available on a desk top. A 400 parameter system, on the other hand, can be analyzed on a typical workstation in minutes so we choose $d' = 400''$.

The top 400 coordinates were recomputed for each caricature and their autocovariance matrix over the ensemble was diagonalized using the LINPACK [42] singular value decomposition routine.

Figure 11.15 shows the accumulation of total variance on the first d'' coordinates in the approximate Karhunen–Loève basis, sorted in decreasing order, as a fraction of the total variance of the 400 joint best basis coefficients, for $1 \leq d'' \leq 143$. The Karhunen–Loève post-processing for this small ensemble concentrates 98 percent of the retained variance from the top 400 joint best basis parameters into ten coefficients.

The fingerprint classification problem

Virtually the same method as the one applied to faces can be applied to fingerprints for identification purposes. The United States FBI uses eight bits per pixel to define the shade of gray and stores 500 pixels per inch, which works out to about 700,000 pixels and 0.7 megabytes per finger to store fingerprints in electronic form. This means that $d \approx 10^6$, so we are forced to use the fast approximate algorithm if we wish to compute the Karhunen–Loève expansion of a fingerprint.

There is no apparent relation between the parameters chosen by the Karhunen–Loève transform and the traditional parameters (location of "minutiæ" points in a fingerprint) which are used by police inspectors to describe fingerprints. Thus the additional classifying values would have to be stored alongside the more traditional values. But since the Karhunen–Loève parameters occupy only a few hundred bytes, which is a negligible amount of space compared to the million bytes of raw data, this subsidiary classification can be added to the fingerprint data base at a negligible cost.

Rank reduction for complex classifiers

The principle behind the fast approximate Karhunen–Loève transform is to employ a relatively low complexity $O(d^2 \log d)$ "front end" to reduce the rank of the subsequent high complexity $O(d^3)$ algorithm from d to $d' \ll d$.

If some measurements on an ensemble of random vectors are to be processed further by some other complex algorithm, we may similarly gain a significant speed advantage by preprocessing the data to reduce the number of parameters. A typical example would be processing for statistical classification from a large set of measurements. Classes, or regions in the measurement space $\mathbf{R}^d$, may have complicated boundaries which must be approximated by high-order polynomial hypersurfaces. Deciding at high orders whether a point lies in a particular region becomes very expensive when the dimension d grows large, so a reduction in the number of parameters will gain speed even if does not by itself simplify the geometry of the regions.

High complexity classifiers are used in speech recognition and machine vision systems. Adding a front end to reduce their workload is like adding a hearing aid or glasses so they can better focus on the most varying features. In some cases, the speed is desirable because we wish to perform the classification in "real time" or least fast enough to keep up with the inflowing data. Some examples are the following:

Ranks of various feature detection problems

- Mechanical failure detection from strain gauge data: $d \approx 10^2$;
- Target recognition from high-resolution radar range profiles: $d \approx 10^2$;
- Detection of irregular heartbeats from acoustic samples: $d \approx 10^2$;
- Phoneme detection: $d \approx 10^3$;
- Optical character recognition: $d \approx 10^3$;
- Detection of machine tool breakage from acoustic samples: $d \approx 10^3$.

11.2.3 Jacobians of complicated maps

Suppose that $T : \mathbf{R}^p \to \mathbf{R}^d$ is some smooth vector field with $p \ll d$. We may think of making plenty (d) of measurements of Tx for a variable x with only a few (p) degrees of freedom. This situation models one course of action to determine what a complicated map T is doing.

Approximating the tangent space in principal components

Recall that the *Jacobian* of T at a point $x \in \mathbf{R}^p$ is the $d \times p$ matrix $J = J_T[x]$ which gives the best linear approximation to T in a neighborhood of x. The coefficients of J are the various partial derivatives of T.

$$J_T[x](i,j) \overset{\text{def}}{=} \lim_{r \to 0} \left\langle e_i, \frac{T(x + re_j) - T(x)}{r} \right\rangle . \tag{11.16}$$

Here $1 \le i \le d$, $1 \le j \le p$, and e_i is the i^{th} standard basis vector. However, the numerical computation of this Jacobian poses some difficulties because the difference quotient is ill-conditioned. Furthermore, the Jacobian might itself be an ill-conditioned matrix, but this difference quotient procedure offers no way of estimating the condition number of J. We will address these difficulties by replacing the difference quotient formula with an approximation based on the Karhunen–Loève transform for the positive matrix JJ^*. The error will lie solely in the approximation, since the Karhunen–Loève transform is orthogonal and thus perfectly conditioned. We will estimate the *condition number* of J from the singular value decomposition of J^*J. Then

$$\mathit{cond}(J) = \sqrt{\mathit{cond}(J^*J)} \approx \sqrt{\mu_1/\mu_p}, \tag{11.17}$$

where μ_1 and μ_p are respectively the first and n^{th} singular values of our estimate for J^*J.

Suppose first that T is a linear map, so that $T = J$ is its own Jacobian. Fix $x \in \mathbf{R}^p$, and consider the ball $B_r = B_r(x) \overset{\text{def}}{=} \{y \in \mathbf{R}^p : \|y - x\| \le r\}$ of radius $r > 0$, centered at x. We can consider the image $JB_r = \{Jy : y \in B_r(x)\} \subset \mathbf{R}^d$ of this ball under the transformation J to be an ensemble of random vectors. This will have expectation $E(JB_r) = JE(B_r) = Jx$, and we can compute the autocovariance matrix of the zero-mean ensemble $\widetilde{JB_r} \overset{\text{def}}{=} JB_r - Jx$ as follows:

$$\begin{aligned} E(\widetilde{JB_r} \otimes \widetilde{JB_r}) &= E_{y \in B_r(x)}(J\tilde{y}\,[J\tilde{y}]^*) \\ &= JE_{y \in B_r(x)}(\tilde{y}\tilde{y}^*)J^* = r^2 JJ^*. \end{aligned}$$

Here $\tilde{y} = y - x$ and the last equality holds since $E_{y \in B_r(x)}(\tilde{y}\tilde{y}^*) = r^2 I_d$ is just a constant times the $d \times d$ identity matrix and thus commutes out from between J and J^*. Thus $r^{-2}E(\widetilde{JB_r} \otimes \widetilde{JB_r}) = JJ^*$.

Proposition 11.3 *For every matrix J,*

$$\textit{Rank } JJ^* = \textit{Rank } J = \textit{Rank } J^* = \textit{Rank } J^*J.$$

Proof: To prove the first equality, notice that Range $JJ^* \subset$ Range J implies that Rank $J \geq$ Rank JJ^*. Now suppose that Range $JJ^* \neq$ Range J. There is some $y \neq 0$ with $y \in$ Range J and $\langle y, JJ^*z\rangle = \langle J^*y, J^*z\rangle = 0$ for all z. Putting $z = y$ we see that $J^*y = 0$. But by its definition, $y = Jx$ for some x, so we have $\|y\|^2 = \langle y, Jx\rangle = \langle J^*y, x\rangle = 0$, a contradiction. The third equality follows from the same argument if we substitute J^* for J. For the middle equality, note that Rank $AB \leq \min\{\text{Rank } A, \text{Rank } B\}$, so that the first equality gives Rank $J \leq$ Rank J^* while the third gives Rank $J \geq$ Rank J^*. □

Suppose J is an $d \times p$ matrix with $d \geq p$. If J has maximal rank p, then J^*J also has rank p. Now, the condition number of J is

$$\sup\{\frac{\|Jx\|}{\|x\|} : x \neq 0\} \Big/ \inf\{\frac{\|Jy\|}{\|y\|} : y \neq 0\}. \tag{11.18}$$

If J^*J has n nonzero singular values $\mu_1 \geq \cdots \geq \mu_p > 0$, counting multiplicities, then the supremum is $\sqrt{\mu_1}$ and the infimum is $\sqrt{\mu_p}$. To see this, let $z_1, \ldots, z_p$ be the orthonormal basis of $\mathbf{R}^p$ consisting of unit singular vectors for J^*J, which is guaranteed to exist because J^*J is a Hermitean matrix. Writing $x = \sum_i a_i z_i$ we have

$$\frac{\|Jx\|^2}{\|x\|^2} = \frac{\langle Jx, Jx\rangle}{\|x\|^2} = \frac{\langle J^*Jx, x\rangle}{\|x\|^2} = \frac{\sum_i a_i^2 \mu_i}{\sum_i a_i^2}. \tag{11.19}$$

This average is maximized when just a_1 is nonzero and minimized when just a_p is nonzero; it then equals μ_1 and μ_p, respectively. Thus we can compute the condition number of J using the formula in Equation 11.17.

Now suppose that T is any smooth vector field from $\mathbf{R}^p$ to $\mathbf{R}^d$, and x is some point in $\mathbf{R}^p$. We compute $z_r = E(TB_r)$, where $TB_r = TB_r(x) = \{Ty : \|y-x\| \leq r\}$; this is the expected value of Ty for y in the ball $B_r(x)$ of radius r centered at x. This average gives a second-order approximation to Tx:

Proposition 11.4 $\|z_r - Tx\| = O(r^2)$ *as* $r \to 0$.

Proof: We can use Taylor's theorem to write $T(x+y) = Tx + Jy + O(\|y\|^2)$ for $\|y\| \leq r$. But $E(Jy) = JE(y) = 0$ because we evaluate the expectation over $y \in B_r(0)$. Thus $E(TB_r) = Tx + O(r^2)$. □

We now define a positive matrix

$$A = A_r = E([TB_r - z_r] \otimes [TB_r - z_r]), \tag{11.20}$$

where the expectation is taken over the ball of radius r. Our main theorem is the following:

Theorem 11.5 $\lim_{r\to 0} \frac{1}{r^2} A_r = JJ^*.$

Proof: Using Proposition 11.4 we write $z_r = Tx + O(r^2)$. We then use Taylor's theorem to get the following estimate:

$$\begin{aligned}[T(x+y) - z] \otimes [T(x+y) - z] &= [Jy + O(\|y\|^2)] \otimes [Jy + O(\|y\|^2)] \\ &= (Jy)(Jy)^* + O(\|y\|^3).\end{aligned}$$

Taking the expectation of both sides over $y \in B_r(0)$ gives $A_r = r^2 JJ^* + O(r^3)$, yielding the desired result. The error estimate is $\|JJ^* - \frac{1}{r^2}A_r\| = O(r)$. □

Now suppose that $x \in \mathbf{R}^p$ is a point for which the Jacobian $J_T[x]$ has full rank p. Full rank is an open condition, *i.e.*, is it possessed by all matrices sufficiently close to J as well, so we have the following reassuring fact:

Corollary 11.6 *For all sufficiently small $r > 0$, Rank $A_r \geq$ Rank $J = p$.* □

Then we can approximate the map T in a neighborhood of Tx using the singular vectors for A_r:

Corollary 11.7 *If $\{z_1, \ldots, z_p\} \subset \mathbf{R}^d$ is a set of unit orthogonal singular vectors for A_r, then there are p linear functions $c_1, \ldots, c_p$ on $\mathbf{R}^p$ such that $T(x + y) = z + \sum_{i=1}^{p} c_i(y) z_i + O(r\|y\|^2)$.* □

We are not really concerned with the rank of A_r being too small, since A_r is a $d \times d$ matrix and $d \gg p$ is the interesting situation. Rather, we worry that choosing a too large value for r will result in Rank A_r being too large, so that we will not be able to identify the top few singular vectors which approximately span Range J. The problem is that if T has nonvanishing higher order derivatives, then the range of A_r will be higher-dimensional than the tangent space to T at Tx, which is the range of J. Schematically, this is depicted in Figure 11.16. The range of A_r is drawn as the two unit singular vectors z_1, z_2 multiplied by their respective singular values μ_1, μ_2. Notice that $\mu_2 \ll \mu_1$, illustrating what happens with smooth T: the variation μ_2 of TB_r in the nontangential direction z_2 is much smaller than the variation μ_1 in the tangent or z_1 direction. In practice, we will always have Rank $A_r = d$ because

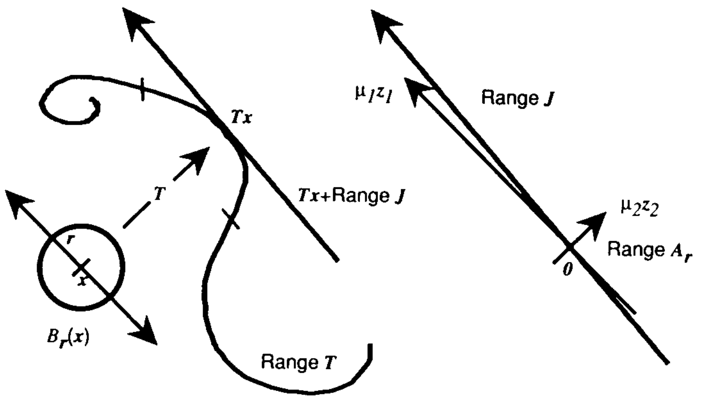


Figure 11.16: Tangent space (Range J) of T and its approximation (Range A_r).

to a finite precision machine every approximate matrix looks like it has full rank. However, if we arrange the singular values of A_r (with multiplicity) in decreasing order $\mu_1 \geq \cdots \mu_p \geq \cdots \geq 0$, then for small enough r we expect a steep drop between μ_p and μ_{p+1}. This then provides a method of choosing the largest r for which the singular vectors of A_r provide an accurate parametrization of T near x. Namely, we let r increase until $\sqrt{\mu_{p+1}/\mu_p}$ reaches some preset threshold of precision ϵ_μ. Then the nontangential components will contribute an error which is $1/\epsilon_\mu$ times smaller than the tangential components.

The functions $c_1, \ldots, c_p$ in Corollary 11.7 correspond to partial derivatives, but they are computed by orthogonal projection. We define matrix coefficients c_{ij} using the elementary basis vectors e_j of $\mathbf{R}^p$ as follows:

$$C_{ij} \stackrel{\text{def}}{=} \frac{1}{r} \left\langle z_i, T(x + re_j) - z \right\rangle . \tag{11.21}$$

Then we extend this to the superposition $y = \sum_{j=1}^{p} a_j e_j$ by taking linear combinations as follows:

$$c_i(y) \stackrel{\text{def}}{=} \sum_{j=1}^{p} C_{ij} a_j . \tag{11.22}$$

Comparing Equation 11.21 with Equation 11.16, we see that the $d \times p$ matrix J has been replaced with the $p \times p$ matrix C, the limit has been reduced to a single evaluation using the largest acceptable r, the initial point Tx is now an average z, and the standard basis $\{e_j : j = 1, \ldots, d\}$ in the range space has been replaced with a new orthonormal basis which is locally adapted to T.

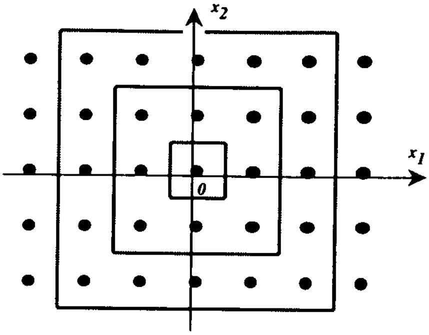


Figure 11.17: Cubes of various radii centered at zero.

Notice that the columns of the $p \times p$ matrix $C = (C_{ij})$ are given by the top p coordinates of the Karhunen–Loève transform of the secant vectors

$$\frac{1}{r}\left[T(x + re_1) - z\right], \frac{1}{r}\left[T(x + re_2) - z\right], \ldots, \frac{1}{r}\left[T(x + re_p) - z\right],$$

since the unit singular vectors $z_1, \ldots, z_p$ of A_r are the Karhunen–Loève eigenvectors for the ensemble TB_r. These secant vectors are approximations to the directional derivatives of T in the directions $e_1, e_2, \ldots, e_p$, and the Karhunen–Loève transform projects them onto the principal orthogonal components along Range T.

Fast approximate Jacobians

We can use the fast approximate Karhunen–Loève algorithm to compute the approximate Jacobian. Suppose that the domain of T includes a cube centered at the origin, namely $B_r = B_r(0) \overset{\text{def}}{=} \{x \in \mathbf{R}^p : |x_1| \leq r, \ldots, |x_p| \leq r\}$. Suppose we lay down a uniform grid of points of the form $x_i = k$ where $k = 0, \pm 1, \pm 2, \ldots$ and $i = 1, \ldots, p$. The intersection of the cube with this grid, which we will also call B_r, contains $(2r + 1)^p$ points in all. We now compute Tx at all points $x \in B_r$, and call the resulting set TB_r. This will be our ensemble of "random" vectors. Each $x \in B_r$ produces a vector $Tx \in \mathbf{R}^d$ which requires d numbers to store, so TB_r will contain $|B_r|d = (2r + 1)^p \times d$ floating-point numbers.

The approximation to T at zero using B_r will be computed by the wavelet packet best basis algorithm above. The mean vector $z = E(TB_r)$ is computed in all wavelet packet bases at once in the means tree. We form the variance tree from the squares tree and the means tree, and search it for the joint best basis of TB_r. We may assume that this basis is sorted into decreasing order of variance.

Now we take d' of the most varying terms out of the d in the joint best basis to retain $1-\epsilon$ of the total variance. We then form the $d' \times d'$ autocovariance matrix for these d' joint best basis vectors and diagonalize it, finding the singular vectors $z_1, \ldots, z_{d'}$ and corresponding singular values $\mu_1 \geq \ldots \geq \mu_{d'}$. We put the singular vectors into the columns of a matrix K', and get the *approximate Karhunen-Loève transform* K'^*, a $d' \times d'$ matrix which gives the approximate principal components when applied to the top d' joint best basis coordinates.

We now test whether the cube B_r is too large for the singular vectors to approximate the tangent space well. The rank of the Jacobian is at most p, so we must have $\epsilon_\mu(r) = \sqrt{\mu_{p+1}/\mu_p} \ll 1$. This gives the first parameter of the algorithm: we know that $\epsilon_\mu(r) \to 0$ as $r \to 0$, so if it exceeds a preset threshold we need to reduce r. However, if $\epsilon_\mu(r)$ meets our requirements, then we can discard all but the first p columns of K' to get the $d' \times p$ matrix K''. The range of K'' serves as the approximate tangent space at $T0$, and K''^* computes coordinates in this space from the top d' joint best basis coordinates.

Finally, we form the approximate Jacobian into this approximate tangent space by using Equation 11.21 with an approximate principal factor for z_i. One by one, we take the secant vectors $\frac{1}{r}\left[T(x+re_j)-z\right]$ for $j = 1, 2, \ldots, p$, which live in $\mathbf{R}^d$, and we find their joint best basis expansions. We then extract from each of those expansions the previously chosen d' coordinates which have most of the variance over the ensemble and apply K''^* to the vectors of these coordinates. That gives a list of p vectors in $\mathbf{R}^p$ which approximate the coefficients of the partial derivatives $\partial_1 T, \ldots, \partial_p T$ in the approximate tangent space basis.

The approximate Jacobian data consists of the following:

Data needed to approximate the Jacobian

- The joint best basis description down to d' coordinates (d' numbers);
- The p vectors of the approximate tangent space expressed as combinations of the first d' joint best basis vectors (pd' numbers);
- The $p \times p$ matrix C of partial derivatives expressed as combinations of the approximate tangent vectors.

Computing these quantities will cost us $O(|B_r| \times [d'^3 + d^2 \log d])$ arithmetic operations, where we expect $d' \ll d$.

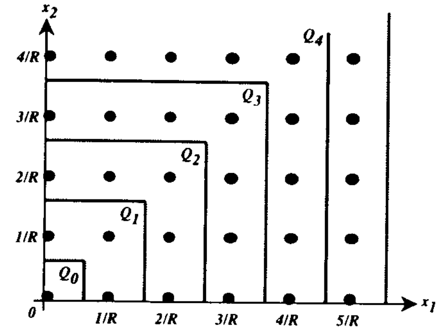


Figure 11.18: Patches of gridpoints at various distances from zero.

Efficient storage of complicated maps

Suppose for this application that the domain of T is the unit cube in $Q \subset \mathbf{R}^p$ defined by $Q = \{x \in \mathbf{R}^p : 0 \leq x_1 \leq 1, \ldots, 0 \leq x_p \leq 1\}$, and suppose we lay down a uniform grid of points of the form $x_i = r/R$ where $r = 0, 1, \ldots, R$ and $i = 1, \ldots, p$. We will call this grid G; it has mesh $1/R$ and contains $|G| = (R+1)^p$ points in all. We now compute Tx at all points $x \in G$, and call the resulting set TG. This is an enormous data set: each $x \in G$ produces a vector $Tx \in \mathbf{R}^d$ which requires d numbers to store, so TG will contain $|G|d = (R+1)^p \times d$ floating-point numbers.

We now use approximate Jacobians to reduce the size of the data set. We will do this by building up patches in the domain where T is well-approximated by its approximate Jacobian. With the fast update algorithm, we can segment the domain of T into patches on which we are sure that the approximation remains within a preset error. Suppose we start at $0 \in G$, at one corner of the cube Q. For each $r = 0, 1, 2, \ldots$ we define the set $Q_r = Q_r(0) \stackrel{\text{def}}{=} \{x \in G : 0 \leq x_i \leq r/R \text{ for } i = 1, \ldots, p\}$. We also define the set $P_r = P_r(0) \stackrel{\text{def}}{=} Q_r(0) \setminus Q_{r-1}(0)$, the partial cubical shell at radius r/R. Then $Q_{r+1} = Q_r \cup P_{r+1}$. This arrangement is depicted for the two-dimensional case $p = 2$ by the points enclosed in lightly colored boxes in Figure 11.18. Note that Q_r contains $|Q_r| = (r+1)^p$ points, while P_r contains $|P_r| = |Q_r| - |Q_{r-1}| = (r+1)^p - r^p \approx pr^{p-1}$ points.

The segmentation algorithm works by first initializing $r = 1$, and then iterating through an algorithm that enlarges a patch on which T is linearly approximated by its approximate Jacobian. We stop enlarging the patch Q_r when the variance of TQ_r along the approximate tangent vectors to T stops being much larger than the variance along approximate normal vectors. In following, we assume that the approximate Karhunen–Loève basis for TQ_{r-1} has already been determined, using the algorithm in Section 11.2.1 above. The update algorithm is also defined in that section.

Segmentation into regions of good linear approximation

- Compute the wavelet packet means and sums-of-squares trees for the extra vectors TP_r;
- Update the joint best basis for TQ_{r-1} by adding in the data from TP_r to get the joint best basis for TQ_r;
- Compute and store the approximate Karhunen–Loève basis K'_r for TQ_r and the singular values $\mu_1 \geq \ldots \geq \mu_{d'} \geq 0$;
- If $\epsilon_\mu(r) = \sqrt{\mu_{p+1}/\mu_p}$ is too large, then:
  - Compute and store $z = ETQ_{r-1}$ for the center;
  - Compute and store the approximate tangent vectors K''_{r-1} for the patch TQ_{r-1};
  - Compute and store the approximate Jacobian using K''_{r-1} and Equation 11.21;
  - Reset $k = 1$;
  - Move to the next free point in G;
- Else if $\epsilon_\mu(r)$ is still small enough, then increment r by one;
- Repeat.

This algorithm will eat away at the domain G, producing a covering of patches Q of various sizes, each with its center point $z_Q = ETQ$, its local approximate tangent space K''_Q, and its approximate Jacobian C_Q. These quantities will require d', pd', and p^2 real numbers to store, respectively. If there are a total of N patches, then the total amount of data to store is $N(d'+pd'+p^2) = O(Npd')$ numbers since $p \leq d'$. If p is small, $N \ll |G|$ and $d' \ll d$, then this compares favorably with the storage requirements for TG, namely $O(d|G|)$.

The complexity of computing these quantities on all of the patches can be estimated from the complexity of finding the approximate Jacobian for the single patch containing all of G, since this is the worst case. But from the previous section, we see that this is $O(|G| \times [d'^3 + d^2 \log d])$ arithmetic operations.

Applying this approximation of T to a vector $x \in \mathbf{R}^p$ involves first finding the patch Q with $x \in Q$. Suppose that x_Q is the center grid point of Q. Then in the first d' joint best basis coordinates,

$$\widetilde{Tx} = z_Q + K''_Q C_Q(x - x_Q). \tag{11.23}$$

We finally superpose the d' joint best basis vectors to get the coordinates of the point $Tx \in \mathbf{R}^d$ from $\widetilde{Tx}$. The complexity of computing Tx this way is $O(p + p^2 + pd' + d' + d \log d)$ which under our assumptions is bounded by $O(d \log d)$.

Precomputation for inverse problems

The final application is to use the local approximate Jacobians to invert the map $x \mapsto Tx$, which we suppose has already been computed at all points x on a finite grid G.

One classical way is to use linear interpolation: given $y \in \mathbf{R}^d$, we find the nearest points $Tx_k \in \mathbf{R}^d$ computed from grid points $x_k \in G$ and write $y = \sum_k a_k Tx_k$. Then the linear approximation to $T^{-1}y$ is just $\sum_k a_k x_k$. This is exactly correct for linear maps T and has at least $O(h)$ accuracy for a differentiable map T on a grid with mesh h. However, it requires that we store the precomputed values $\{Tx : x \in G\}$ and that we search the whole list for the points close to y. The last step, in particular, requires computing $|G|$ distances for a grid G.

If we have invested the effort to compute the approximate Jacobian representation of T, then the inverse can be approximated from that data instead. Let N be the number of patches in the cover of G, and suppose that $N \ll |G|$. We also suppose for the sake of simplicity that we keep the same number d' of joint best basis components in each patch, although of course they may be different components in different patches. Finally, we suppose that we have already computed the inverses C_Q^{-1} of the approximate Jacobians on each patch Q, which requires a one-time investment of $O(Np^3)$. Then the necessary computations and their complexities for computing $T^{-1}y$ at a single $y \in \mathbf{R}^d$ are the following:

Approximate inverse via approximate Jacobian

- Find the complete wavelet packet expansion of y, which simultaneously computes all joint best basis expansions $\tilde{y}$: $O(d \log d)$;
- Compute the distances from $\tilde{y}$ to the means z_Q for each patch Q, and let Q henceforth be the patch with nearest mean: $O(Nd')$;
- Compute the approximate inverse,

$$T^{-1}y \approx x_Q + C_Q^{-1} K''^{*}_{Q}(\tilde{y} - z_Q),$$

for the nearest patch Q: $O(d' + pd' + p^2 + p) = O(pd')$.

11.3 Nonstandard matrix multiplication

By decomposing a matrix into its two-dimensional best basis, we reduce the number of nonnegligible coefficients and thus reduce the computational complexity of matrix application.

11.3.1 Two-dimensional best basis sparsification

Write $\mathcal{W}(\mathbf{R})$ for the collection of one-dimensional wavelet packets. Let ψ_{sfp} be a representative wavelet packet of frequency f at scale s and position p. Conjugate quadrature filters may be chosen so that $\mathcal{W}(\mathbf{R})$ is dense in many common function spaces. With minimal hypotheses, $\mathcal{W}(\mathbf{R})$ will be dense in $L^2(\mathbf{R})$. Using the Haar filters $\{1/\sqrt{2}, 1/\sqrt{2}\}$ and $\{1/\sqrt{2}, -1/\sqrt{2}\}$, for example, produces $\mathcal{W}(\mathbf{R})$ which is dense in $L^p(\mathbf{R})$ for $1 < p < \infty$. Longer filters can generate smoother wave packets, so we can also produce dense subsets of Sobolev spaces, etc.

Ordering wave packets

Wavelet packets ψ_{sfp} can be totally ordered. We say that $\psi < \psi'$ if $(s, f, p) < (s', f', p')$. The triplets are compared lexicographically, counting the scale parameters s, s' as most significant.

Write $\psi_X = \psi_{s_X f_X p_X}$, etc., and observe that tensor products of wavelet packets inherit this total order. We can say that $\psi_X \otimes \psi_Y < \psi'_X \otimes \psi'_Y$ if $\psi_X < \psi'_X$ or else if $\psi_X = \psi'_X$ but $\psi_Y < \psi'_Y$. This is just lexicographical comparison of the lists $(s_X, f_X, p_X, s_Y, f_Y, p_Y)$ and $(s'_X, f'_X, p'_X, s'_Y, f'_Y, p'_Y)$ from left to right.

The *adjoint order* $<^*$ just exchanges X and Y indices: $\psi_X \otimes \psi_Y <^* \psi'_X \otimes \psi'_Y$ if and only if $\psi_Y \otimes \psi_X <^* \psi'_Y \otimes \psi'_X$. It is also a total order.

Projections

Let $\mathcal{W}^1$ denote the space of bounded sequences indexed by the three wave packet indices s, f, p. With the ordering above, we obtain a natural isomorphism between ℓ^∞ and $\mathcal{W}^1$. There is also a natural injection $J^1 : L^2(\mathbf{R}) \hookrightarrow \mathcal{W}^1$ given by $J^1 x = \{\lambda_{sf}(p)\}$ for $x \in L^2(\mathbf{R})$, with $\lambda_{sf} = \langle x, \psi^{<}_{sfp}\rangle$ being the sequence of backwards inner products with functions in $\mathcal{W}(\mathbf{R})$. If B is a basis subset, then the composition J^1_B of J^1 with projection onto the subsequences indexed by B is also injective. J^1_B is an isomorphism of $L^2(\mathbf{R})$ onto $l^2(B)$, which is defined to be the square-summable sequences of $\mathcal{W}^1$ whose indices belong to B.

The inverse is a map $R^1 : \mathcal{W}^1 \to L^2(\mathbf{R})$ defined by

$$R^1\lambda(t) = \sum_{(s,f,p)\in\mathbf{Z}^3} \bar{\lambda}_{sf}(p)\psi'^{\lessgtr}_{sfp}(t). \qquad (11.24)$$

This map is defined and bounded on the closed subspace of $\mathcal{W}^1$ isomorphic to l^2 under the natural isomorphism mentioned above. In particular, R^1 is defined and bounded on the range of J^1_B for every basis subset B. The related restriction $R^1_B : \mathcal{W}^1 \to L^2(\mathbf{R})$ defined by $R^1_B\lambda(t) = \sum_{(s,f,p)\in B} \bar{\lambda}_{sfp}\psi'^{\lessgtr}_{sfp}(t)$ is a left inverse for J^1 and J^1_B. In addition, $J^1R^1_B$ is a projection of $\mathcal{W}^1$. Likewise, if $\sum_i \alpha_i = 1$ and $R^1_{B_i}$ is one of the above maps for each i, then $J^1\sum_i \alpha_i R^1_{B_i}$ is also a projection of $\mathcal{W}^1$. It is an orthogonal projection on any finite subset of $\mathcal{W}^1$.

Similarly, writing $\mathcal{W}^2$ for $\mathcal{W}^1 \times \mathcal{W}^1$, the ordering of tensor products gives a natural isomorphism between ℓ^∞ and $\mathcal{W}^2$. Objects in the space $L^2(\mathbf{R}^2)$, *i.e.*, the Hilbert-Schmidt operators, inject into this sequence space $\mathcal{W}^2$ in the obvious way, namely $M \mapsto \langle M, \psi^{<}_{s_X f_X p_X} \otimes \psi^{<}_{s_Y f_Y p_Y}\rangle$. Call this injection J^2. If B is a basis subset of $\mathcal{W}^2$, then the composition J^2_B of J^2 with projection onto subsequences indexed by B is also injective. J^2_B is an isomorphism of $L^2(\mathbf{R}^2)$ onto $\ell^2(B)$, the square summable sequences of $\mathcal{W}^2$ whose indices belong to B.

The map $R^2 : \mathcal{W}^2 \to L^2(\mathbf{R}^2)$ given by $R^2c(x,y) = \sum \bar{c}_{XY}\psi'^{\lessgtr}_X(x)\psi'^{\lessgtr}_Y(y)$, is bounded on that subset of $\mathcal{W}^2$ naturally isomorphic to ℓ^2. In particular, it is bounded on the range of J^2_B for every basis subset B.

We may also define the restrictions R^2_B of R^2 to subsequences indexed by B, defined by $R^2_Bc(x,y) = \sum_{(\psi_X,\psi_Y)\in B} \bar{c}_{XY}\psi'^{\lessgtr}_X(x)\psi'^{\lessgtr}_Y(y)$. There is one for each basis subset B of $\mathcal{W}^2$. Then R^2_B is a left inverse of J^2 and J^2_B, and $J^2R^2_B$ is a projection of $\mathcal{W}^2$. As before, if $\sum_i \alpha_i = 1$ and B_i is a basis subset for each i, then $J^2\sum_i R^2_{B_i}$ is also a projection of $\mathcal{W}^2$. It is an orthogonal projection on any finite subset of $\mathcal{W}^2$.

11.3.2 Applying operators to vectors

For definiteness, let X and Y be two named copies of $\mathbf{R}$. Let $x \in L^2(X)$ be a vector, whose coordinates with respect to wave packets form the sequence $J^1x = \{\langle x, \psi^{\lessgtr}_X\rangle : \psi_X \in \mathcal{W}(X)\}$.

Let $M : L^2(X) \to L^2(Y)$ be a Hilbert-Schmidt operator. Its matrix coefficients with respect to the complete set of tensor products of wave packets form the sequence $J^2M = \{\langle M, \psi^{\lessgtr}_X \otimes \psi^{\lessgtr}_Y\rangle : \psi_X \in \mathcal{W}(X), \psi_Y \in \mathcal{W}(Y)\}$. We obtain the identity

$$\langle Mv, \psi^{\lessgtr}_Y\rangle = \sum_{\psi_X\in\mathcal{W}(X)} \langle M, \psi^{\lessgtr}_X \otimes \psi^{\lessgtr}_Y\rangle\, \langle x, \psi^{\lessgtr}_X\rangle. \qquad (11.25)$$

This identity generalizes to a linear action of $\mathcal{W}^2$ on $\mathcal{W}^1$ defined by

$$c(x)_{sfp} = \sum_{(s'f'p')} \lambda_{sfps'f'p'} v_{s'f'p'}. \tag{11.26}$$

Now, images of operators form a proper submanifold of $\mathcal{W}^2$. Likewise, images of vectors form a submanifold $\mathcal{W}^1$. We can lift the action of M on x to these larger spaces via the commutative diagram

$$\begin{tikzcd}
\mathcal{W}^1 \arrow[r, "J_B^2 M"] \arrow[d, phantom, ""] & \mathcal{W}^1 \arrow[d, "R^1"] \\
L^2(\mathbf{R}) \arrow[u, "J^1"] \arrow[r, "M"] \arrow[ur, phantom, "\bigcirc"] & L^2(\mathbf{R}).
\end{tikzcd} \tag{11.27}$$

The significance of this lift is that by a suitable choice of B we can reduce the complexity of the map $J_B^2 M$, and therefore the complexity of the operator application.

Example

We illustrate the above ideas by considering nonstandard multiplication by square matrices of order 16. Using wavelet packets of isotropic dilations, we obtain the *best isotropic basis* action depicted in the Figure 11.19. The domain vector is first injected into a 16 log 16-dimensional space by expansion into the complete domain wavelet packet analysis tree at bottom. It then has its components multiplied by the best basis coefficients in the matrix square at center, as indicated by the arrows. The products are summed into a range wavelet packet synthesis tree at right. The range tree is then projected onto the 16-dimensional range space by the adjoint of the wavelet packet expansion. Notice that input blocks have the same size as the output blocks, so that this method reveals only how energy moves within scales. However, the best isotropic basis expansion of the matrix makes explicit how frequencies and positions are mixed. Of course it is possible to move energy between scales, but the action is not immediately readable from the nonstandard matrix coefficients.

Figure 11.20 depicts the best basis expansion for the matrix $m_{ij} = \sin \frac{\pi ij}{1024}$, with $0 \leq i, j < 1024$. All coefficients smaller in magnitude than one percent of the largest have been set to zero; the survivors are depicted as black dots in the location they would occupy within the tree. No information about the chosen basis is available in that picture; we have suppressed drawing the outlines of the subspace boxes in order to avoid clutter.

If we allow all tensor products of wavelet packets in the decomposition of the matrix, we get the *best tensor basis* nonstandard representation. Finding the best

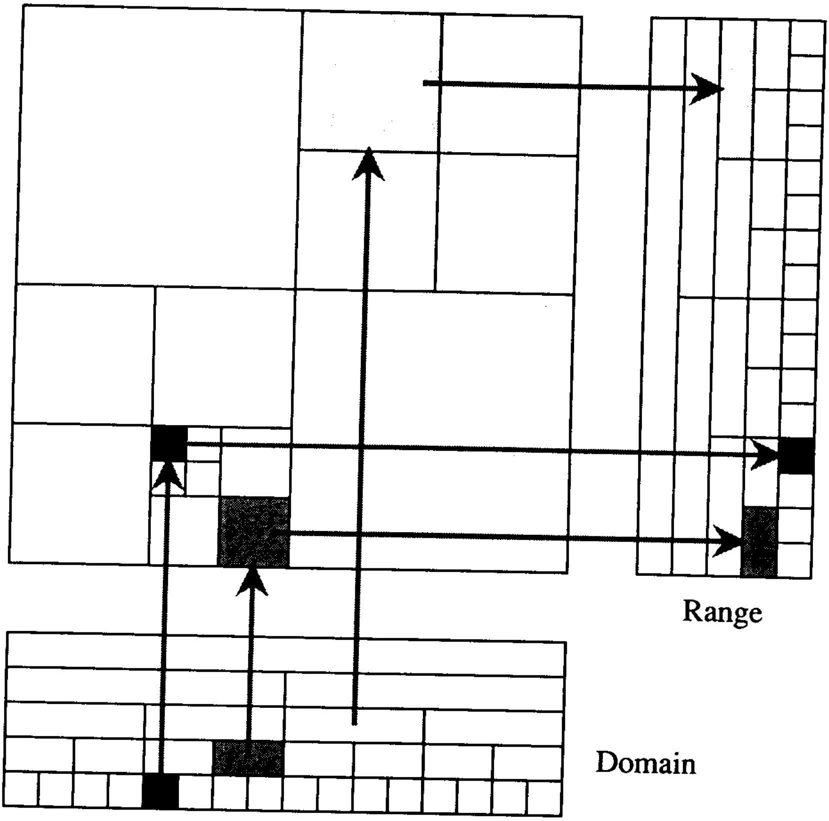


Figure 11.19: Nonstandard 16 × 16 matrix multiplication in isotropic best basis.

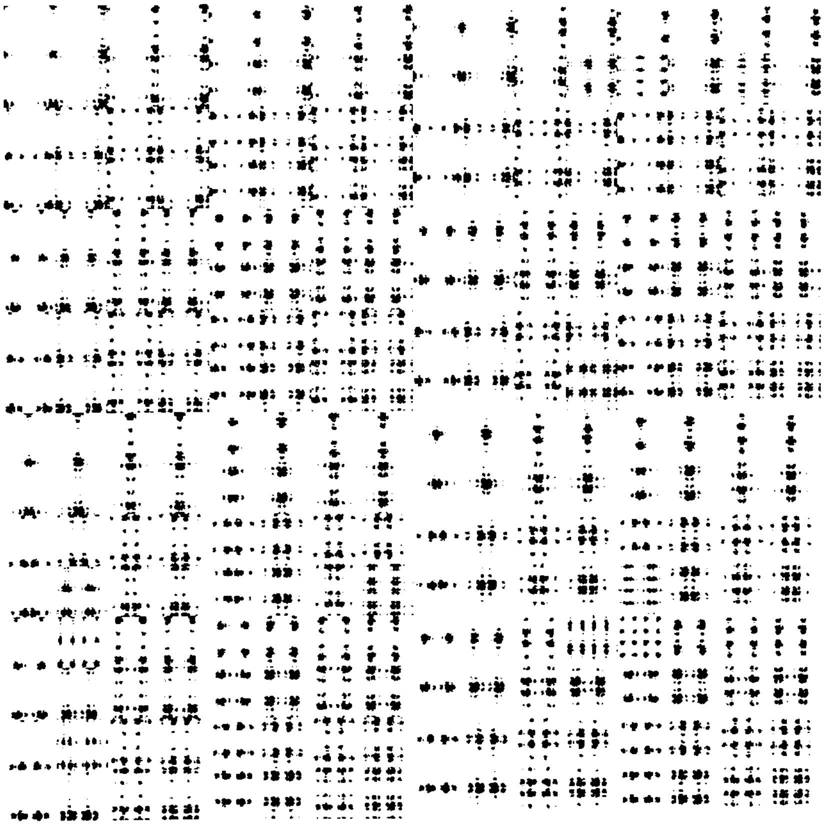


Figure 11.20: A 1024 × 1024 matrix in its isotropic best basis, analyzed with C30 filters.

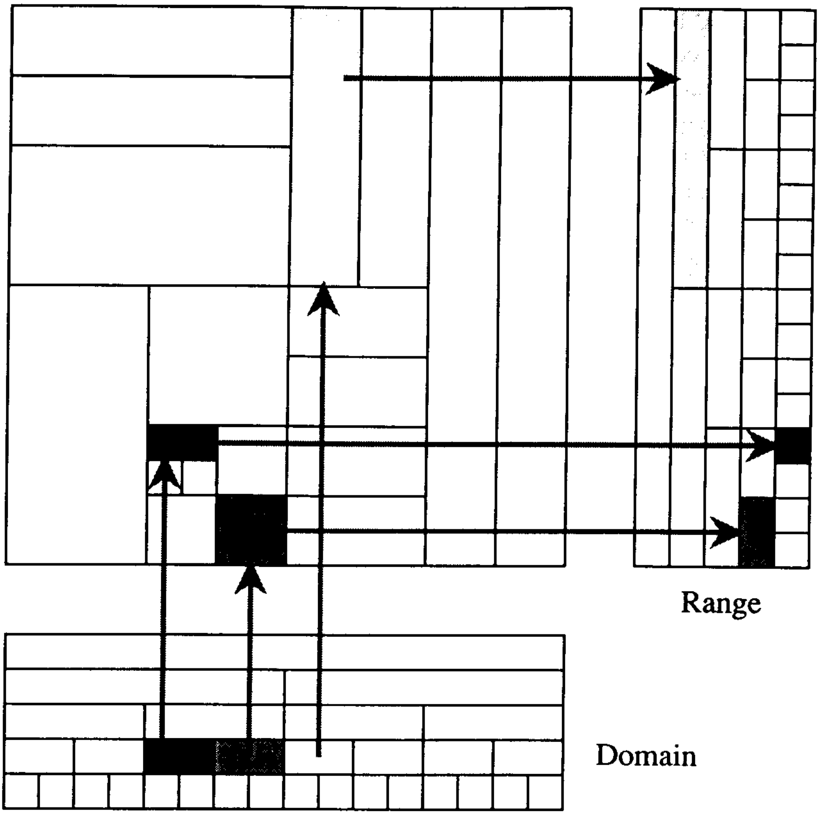


Figure 11.21: Nonstandard 16 × 16 matrix multiplication: anisotropic best basis.

basis for the matrix becomes harder, but the multiplication algorithm is practically the same. Again we must develop the complete wavelet packet analysis of the input to get the domain tree. Blocks in the nonstandard matrix send blocks in the domain tree to possibly larger or smaller blocks in the range tree, where they are synthesized into the output. This is depicted in Figure 11.21, for a 16 × 16 matrix. Such an expansion explicitly shows how energy moves from one scale to another, as well as how position and frequency are mixed.

Rather than choose a basis set among the two-dimensional wavelet packets, we may also consider matrix multiplication to be a list of inner products. The complexity of the operation may be reduced by expressing each row of the matrix in its individual best basis, expanding the domain vector into the complete tree of wavelet packet coefficients, then evaluating the inner products row by row, up to any desired accuracy. Schematically, this accelerated inner product may be depicted as in the left-hand side of Figure 11.22. Notice how the wavelet components of the input are gathered and summed to yield the output values.

A typical candidate for such a *best row basis* expansion is the 1024-point sine transform matrix, depicted in Figure 11.23. In this schematic, enough coefficients were retained to preserve 99 percent of the Hilbert–Schmidt norm of each row, and the surviving coefficients are plotted as black dots in the locations they would occupy. Approximately seven percent of the coefficients survived. No information is provided about the chosen bases in that picture.

We may also expand the columns in their own individual best-bases and consider matrix multiplication to be the superposition of a weighted sum of these wavelet packet expansions. This *best column basis* algorithm is depicted schematically in Figure 11.24. The input value is sprayed into the output wavelet packet synthesis tree, then the output is reassembled from those components.

Operation count

Suppose that M is a nonsparse operator of rank r. Ordinary multiplication of a vector by M takes at least $O(r^2)$ operations, with the minimum achievable only by representing M as a matrix with respect to the bases of its r-dimensional domain and range.

On the other hand, the injection J^2 will require $O(r^2[\log r]^2)$ operations, and each of J^1 and R^1 require $O(r \log r)$ operations. For a fixed basis subset B of $\mathcal{W}^2$, the application of $J^2_B M$ to $J^1 v$ requires at most $\#|J^2_B M|$ operations, where $\#|U|$ denotes the number of nonzero coefficients in U. We may choose our wavelet packet library so that $\#|J^2_B M| = O(r^2)$. Thus the multiplication method described above costs an initial investment of $O(r^2[\log r]^2)$, plus at most an additional $O(r^2)$ per right-hand side. Thus the method has asymptotic complexity $O(r^2)$ per vector in its exact form, as expected for multiplication by a matrix of order r.

We can obtain lower complexity if we take into account the finite accuracy of our calculation. Given a fixed matrix of coefficients C, write C_δ for the same matrix with all coefficients set to zero whose absolute values are less than δ. By the continuity of the Hilbert-Schmidt norm, for every $\epsilon > 0$ there is a $\delta > 0$ such that $\|C - C_\delta\|_{HS} < \epsilon$. Given M and ϵ as well as a library of wavelet packets, we can choose a basis subset $B \subset \mathcal{W}^2$ so as to minimize $\#|(J^2_B M)_\delta|$. The choice algorithm has complexity $O(r^2[\log r]^2)$, as shown above. For a certain class of operators, there

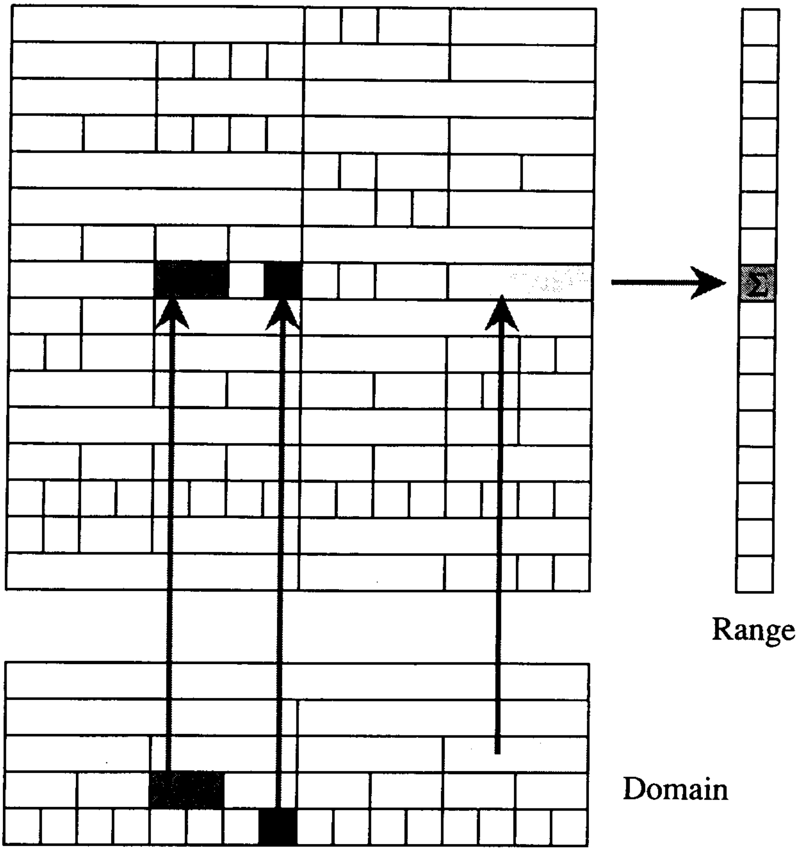


Figure 11.22: Nonstandard 16 × 16 matrix multiplication in best row basis.

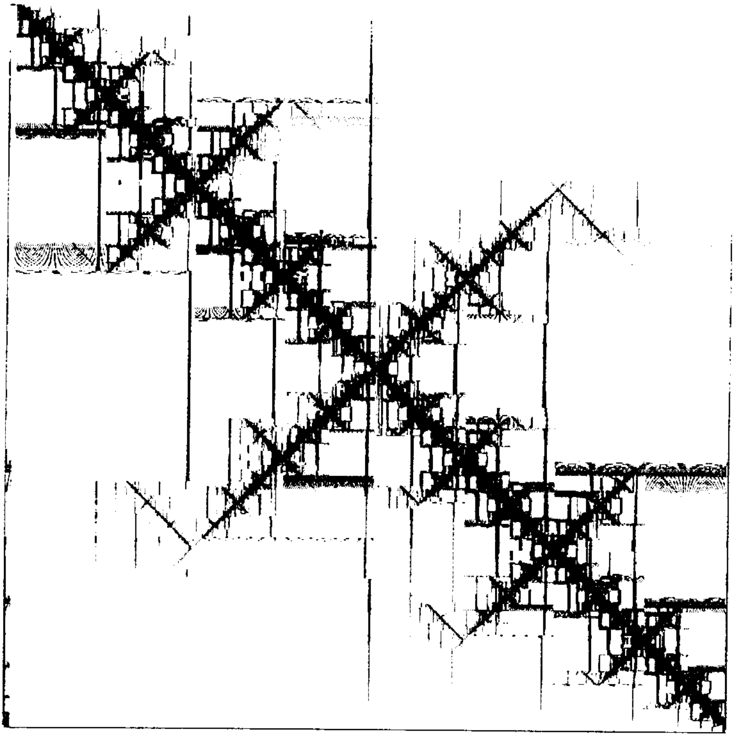


Figure 11.23: A 1024 × 1024 matrix in its best row basis, analyzed with C30 filters.

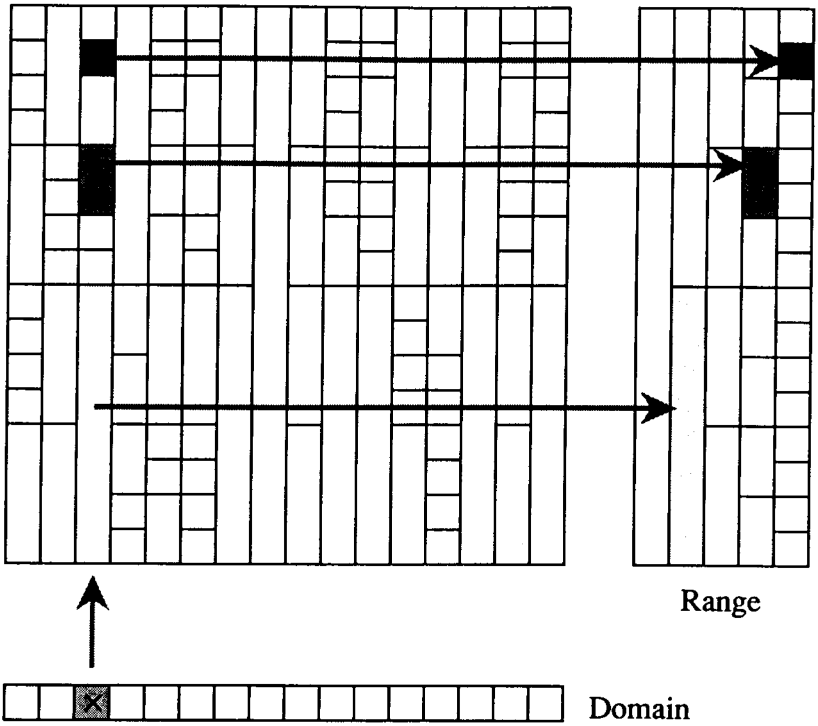


Figure 11.24: Nonstandard 16 × 16 matrix multiplication: best column basis.

is a library of wavelet packets such that for every fixed $\delta > 0$ we have

$$\#|(J_B^2 M)_\delta| = O(r \log r), \tag{11.28}$$

with the constant depending, of course, on δ. Call this class with property 11.28 the *sparsifiable* Hilbert-Schmidt operators $\mathcal{S}$. By the estimate above, finite-precision multiplication by sparsifiable rank-r operators has asymptotic complexity $O(r \log r)$.

The best row basis algorithm is also asymptotically $O(r \log r)$ for sparsifiable matrices, since it requires $O(r \log r)$ operations to find the complete tree of wavelet packet coefficients for the domain vector, then $O(\log r)$ multiply-adds to evaluate each of the r inner products. Finding the best basis for each of the r rows of the matrix requires an initial investment of $O(r^2 \log r)$ operations.

11.3.3 Composing operators

Let X, Y, Z be three named copies of $\mathbf{R}$. Suppose that $M : L^2(X) \to L^2(Y)$ and $N : L^2(Y) \to L^2(Z)$ are Hilbert-Schmidt operators. We have the identity

$$\langle NM, \psi_X^< \otimes \psi_Z^< \rangle = \sum_{\psi_Y^< \in \mathcal{W}(Y)} \langle N, \psi_Y^< \otimes \psi_Z^< \rangle \, \langle M, \psi_X^< \otimes \psi_Y^< \rangle.$$

This generalizes to an action of $\mathcal{W}^2$ on $\mathcal{W}^2$, which is defined by the formula

$$\lambda(d)_{sfps'f'p'} = \sum_{s''f''p''} \mu_{sfps''f''p''} \, \lambda_{s''f''p''s'f'p'},$$

where λ and μ are sequences in $\mathcal{W}^2$. Using J^2, we can lift multiplication by N to an action on these larger spaces via the commutative diagram

$$\begin{tikzcd}
\mathcal{W}^2 \arrow[r, "J_B^2 N"] & \mathcal{W}^2 \arrow[d, "R^2"] \\
L^2(\mathbf{R}^2) \arrow[u, "J^2"] \arrow[r, "N"] \arrow[ur, phantom, "\bigcirc"] & L^2(\mathbf{R}^2).
\end{tikzcd} \tag{11.29}$$

Again, by a suitable choice of B the complexity of the operation may be reduced to below that of ordinary operator composition.

Operation count

Suppose that M and N are rank-r operators. Standard multiplication of N and M has complexity $O(r^3)$. The complexity of injecting N and M into $\mathcal{W}^2$ is

$O(r^2[\log r]^2)$. The action of $J_B^2 N$ on $J^2 M$ has complexity $O(\sum_{sfp} \#|J_B^2 N_{YZ} : (s_Y, f_Y, p_Y) = (s, f, p)| \#|J^2 M_{XY} : (s_Y, f_Y, p_Y) = (s, f, p)|)$. The second factor is a constant $r \log r$, while the first when summed over all sfp is exactly $\#|J_B^2 N|$. Thus the complexity of the nonstandard multiplication algorithm, including the conjugation into the basis set B, is $O(\#|J_B^2 N|\, r \log r)$. Since the first factor is r^2 in general, the complexity of the exact algorithm is $O(r^3 \log r)$ for generic matrices, reflecting the extra cost of conjugating into the basis set B.

For the approximate algorithm, the complexity is $O(\#|(J_B^2 N)_\delta|\, r \log r)$. For the sparsifiable matrices, this can be reduced by a suitable choice of B to a complexity of $O(r^2[\log r]^2)$ for the complete algorithm. Since choosing B and evaluating J_B^2 each have this complexity, it is not possible to do any better by this method.

11.4 Speech signal segmentation

Here we apply the adapted local trigonometric transform to decompose digitized speech signals into orthogonal elementary waveforms in such a way that the choice of widths for the waveforms yields a segmentation of the signal. This algorithm leads to a local time-frequency representation which can also be used for compression and recognition. We present some experimental results of signal compression and automatic *voiced-unvoiced segmentation*, prepared by Eva Wesfreid [110]. The compressed signal has been simplified but still appears to be useful for detecting fundamental frequencies and characterizing *formants.*

We begin with a clean, digitized speech signal. The signal is decomposed by a complete adapted local trigonometric analysis, into cosines or sines multiplied by smooth cutoff functions. It is possible to compute several local cosine transforms all at once, recursively subdividing segments of the signal into halves. The basis functions on each subinterval are the orthogonal direct sum of the basis functions on its left and right halves, and this orthogonality propagates up through the multiple levels of the binary "family tree" in Figure 4.17. We then apply the best basis method obtained by entropy minimization [30]. This algorithm produces an adapted orthogonal elementary waveform decomposition, which is a local spectral representation for the speech signal. Roughly speaking, we get a windowed cosine transform of the signal, with the window size well-adapted to the spectrum it contains. A superposition of these functions may be depicted by a sequence of adjacent envelopes or windows, with vertical lines drawn between the nominal window boundaries. This is done in Figure 4.15.

The associated time partition, or choice of windows, appears to be useful for segmentation into voiced and unvoiced portions, which can be recognized by the