Chapter 8

Progressive Data Access for Regular Grids

John Clyne

Computational Information Systems Laboratory, National Center for Atmospheric Research

This chapter presents three progressive refinement methods for data sampled on a regular grid. Two of the methods are based on multiresolution: the grid may be coarsened or refined as needed by dyadic factors. The third is based on the energy compaction properties of the discrete wavelet transform, which enables the sparse representation of signals. In all cases, the objective is to afford the end-user the ability to make trade-offs between fidelity and speed, in response to the available computing resources, when visualizing or analyzing large data.

8.1 Introduction

Fueled by decades of exponential increases in microprocessor performance, computational scientists in a diverse set of disciplines have enjoyed unprecedented supercomputing capabilities. However, with the increase in computing power, comes more sophisticated and realistic computing models and an invariable increase in resolution of the discrete grids used to solve the equations of state. A direct result of the increase in grid resolution is the profuse amount of stored data. Unfortunately, the ability to generate data has not been matched by our ability to consume it. Whereas microprocessor floating point performance, combined with communication interconnect bandwidth and latencies, largely determines the scale by which numerical simulations are run, it is often primary storage capacities and I/O bandwidths that constrain access to data during visualization and analysis. These latter technologies, and I/O bandwidths, in particular, have not kept pace with the performance advancements of CPUs, GPUs, or high performance communication fabrics. For many visualization and analysis workflows, there is a bottleneck caused by the rate at which data is delivered to the computational components of the analysis pipeline.

This chapter discusses methods used to reduce the volume of data that must be processed in order to support a meaningful analysis. Ideally, the user should be able to trade-off data fidelity for increased interactivity. In many applications, aggressive data reduction may have a negligible impact, or no impact whatsoever, on the resulting analyses. The specific target workflow is highly interactive, quick-look exploratory visualization enabled by coarsened approximations of the original data, followed by a less interactive validation of results using the refined or original data, as needed. This model and these methods, which are referred to as *progressive data access* (PDA), are similar to those employed by the ubiquitous GoogleEarth$^{\text{TM}}$—coarsened imagery is transmitted and displayed when the viewpoint is far away, and continuously refined as the user zooms in on a region of interest.

The PDA approaches in this chapter are all intended to minimize the volume of data accessed from secondary storage, whether the storage is a locally attached disk, or a remote, network-attached service. Additionally, the following properties of the data model are also deemed important: (1) the ability to quickly access coarsened approximations of the full data domain; (2) the ability to quickly extract subsets of the full domain at a higher quality; (3) lossless reconstruction of the original data; and (4) minimal storage overhead.

8.2 Preliminaries

A trivial approach to supporting PDA for regular grids is the generalization of 2D texture MIP mapping to higher dimensions. A MIP Map, also referred to in the literature as an *image pyramid*, is a precalculated hierarchy of approximations created by successively coarsening a 2D image, typically reducing the sampling along each dimension by a factor of 2, with each pass. Without loss of generality, if assuming a 3D grid of dimension N^3, a series of approximations is created of the dimensions $(\frac{N}{2})^3$, $(\frac{N}{4})^3$, $(\frac{N}{8})^3$, and so on. An unfortunate consequence of this simplistic tack, though, is the additional storage required by each approximation. For a d-dimensional hypercube, the fractional storage increase is given by:

$$\sum_{j=1}^{J} \frac{1}{2^{dj}},$$

where J is the number of approximations in the hierarchy—an ostensibly small multiplier, but one that may be unacceptable when N is large and multiple variables and timesteps are involved.

In the following sections, there are more sophisticated methods than MIP mapping to support PDA. The overall goal is to improve upon MIP mapping by eliminating, or at least reducing, the additional storage required. The notion of multiple resolutions—based on a hierarchical decomposition of the grid sampling space—plays a role in all of the methods presented. The convention adopted throughout this chapter is that the level in a multiresolution hierarchy is given by j, with j_{min} referring to the coarsest member of the hierarchy, and j_{max} referring to the finest member. The total number of members is $j_{max} - j_{min} + 1$. In general $j_{min} = 0$, but not always. A distinction is always made when j_{min} is not zero.

8.3 Z-order curves

A space-filling curve is one that passes through all points of a 2-dimensional square, or more generally, all points of a d-dimensional hypercube. In the context of a regular grid of dimension N^d, where $N = 2^n$, a space-filling curve visits all vertices of the mesh exactly once, providing a mapping between 1D and higher dimensional space. Here we will focus our attention on the 2D and 3D cases. The Morton space-filling curve, also referred to as the *Z-order* curve due to its shape (see Figure 8.1), visits all vertices in the order of a depth-first traversal of a quadtree (or octree in 3D) [8]. Figure 8.1(d) shows the complete

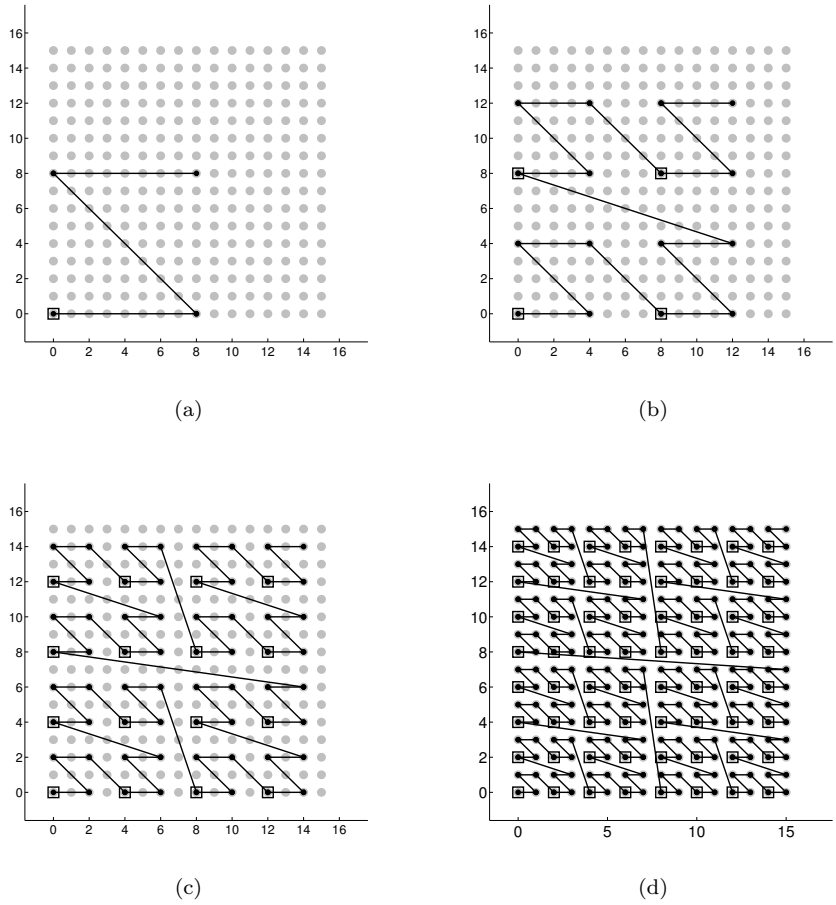


FIGURE 8.1: The Z-order, or Morton, space-filling curve maps multi-dimensional data into one dimension while preserving the locality of neighboring points.

Z-order curve traversal, starting at the grid point with coordinates $(0, 0)$, of a 16×16 2D mesh. When a third dimension is added, plane-adjacent pairs of Z shapes are connected by a diagonal.

Informally, the property of Z-order curves that makes them appealing is that, in contrast to a conventional row or column major order traversal of grid vertices, the Z-order curve preserves the locality of data points; points whose integer offsets are near each other along the Z-order curve are geometrically close to each other, as well. Moreover, the grid points that are members of a particular level of the quadtree (octree) hierarchy are identified in a straightforward manner.

If, instead of using a unit stride, one chooses a suitable non-unit value to traverse the Z-order curve, $s = 2^{d(j_{max}-j)}$, then, all points can be visited at any level j of the tree, while still preserving the locality of points. The maximum number of levels in the hierarchy defined by the Z-order curve is given by $log_2(N) + 1$, thus, $0 \leq j \leq j_{max} = log_2(N)$. The coarsened grid corresponding to $j = 0$, in the Z-order curve decomposition, contains a single point located at the origin of the original grid. Figures 8.1(a)-(d) show the resulting curves when $s = 64$, 16, 4, and 1, respectively. Not shown is the coarsest level: $j = 0$.

8.3.1 Constructing the Curve

The construction of the 1D Z-order curve from the the d-dimensional hypercube is accomplished by interleaving the bits of the binary representation of each grid point index $(i_0, \ldots i_{d-1})$ to form an integer offset, z, along the curve. If each dimension index, i_k is expressed in binary with n bits as $b_k^0 b_k^1 \ldots b_k^{n-1}$, then the 1D reference, z, containing nd bits, is constructed by interleaving the bits of each d-dimensional index starting with the slowest changing dimension. For example, in the 2D case with grid point addresses given by (i_0, i_1), i_1 varying slowest, z would be given by the string of $2n$ bits: $b_1^0 b_0^0 \ldots b_1^{n-1} b_0^{n-1}$. The curve offsets, z, are then sorted from smallest to largest to provide the depth-first traversal order. This establishes a mapping from a point's d-dimensional address, $(i_0, \ldots i_{d-1})$, to a point's offset along the Z-order curve, z.

8.3.2 Progressive Access

As noted, changing the Z-order curve stride, s, can restrict the visitation of points in the tree up to a level j. More formally, a hierarchy of ordered sets of grid point Z-order curve offsets can be constructed, such that:

$$S_0 \subset S_1 \subset \ldots \subset S_{j_{max}}$$

where

$$S_j = \{z_j | z_j = si, i \in \mathbb{Z}, 0 \leq si < 2^{dj_{jmax}}\},$$

$s = 2^{d(j_{max}-j)}$, and the cardinality $|S_j| = 2^{dj}$.

A progressive access storage layout can be affected simply by ordering the data on disk from S_0 to $S_{j_{max}}$, with the order of elements within each S_j given by z. Any approximation level in the hierarchy can then be accessed, reading only those points contained in S_j. An added benefit of preserving the Z-ordering within each S_j is the locality of points. The data are effectively partitioned into blocks, which can offer substantially improved I/O performance, over row (column) major ordering on block-based storage when subsets of the data are read.

There is one problem with the scheme just described: data replication. Each set S_j is a subset of S_{j+1}. To avoid data replication a set differences is constructed, such that:

$$S'_j = S_j - S_{j-1}.$$

S'_j is then stored, instead of S_j. Reconstructing S_j from S'_j requires reading all S'_i, where $0 \leq i \leq j$, and forming the union $S_j = S'_0 \cup \ldots \cup S'_j$. In each of the Figures 8.1(a)-(d), a circle enclosed inside a square is used to denote a grid point added through the union of the sets S'_j and S_{j-1}.

There are a few additional properties of the Z-order curve that are helpful to support the construction and access of S'_j.

The level in the hierarchy that a point with Z-order curve index z belongs to is given by:

$$j = \begin{cases} 0 & \text{if } z == 0, \\ j_{max} - \lfloor \frac{n}{d} \rfloor & \text{if } z > 0, \end{cases} \tag{8.1}$$

where 2^n is the largest power-of-two factor of z.

The cardinality of S'_j is

$$|S'_j| = \begin{cases} 1 & \text{if } j == 0, \\ 2^{dj} - 2^{d(j-1)} & \text{if } j > 0. \end{cases} \tag{8.2}$$

Finally, the z value from a point in the set S'_j can be reconstructed as:

$$z = (l + \lfloor \frac{l}{2^d - 1} \rfloor + 1) 2^{d(j_{max} - j)}, \tag{8.3}$$

where l is the integer offset of the point within the set (recall set elements are ordered by z).

When reordering data on disk from row (column) major order to Z-order curve, Eq. 8.1 allows the user to walk the Z-order curve and determine what point belongs to which set of S'_j When reconstructing row (column) major order from Z-order curve data stored on disk, Eq. 8.3 provides a point's z offset based on its position in S'_j.

There are a few other items worth noting. No floating point calculations are performed on the data in the scheme described above and the storage of the data is simply reordered in a manner that lends itself to progressive access

and subregion extraction. Hence, the reconstructed data at the highest level of the hierarchy are bit-for-bit identical to the original data. In general, this will not be true of other progressive access models. A less desirable aspect of the approach is that data coarsening is performed through simple subsampling. The quality of the coarsened approximations will not be as high as when achieved by other methods.

8.4 Wavelets

In the preceding section, the Z-order curve offers a convenient way to create and access a multiresolution hierarchy composed of the nodes of a gridded hypercube. While the method allows for bit-for-bit identical reconstruction of the original grid and does not impose any additional storage overhead, the restriction to the grids with regular symmetry and power-of-two dimensions, or the crudeness of coarsened approximations created by sub-sampling, may prove limiting. In this section, both of these shortcomings are addressed, but they give up exact, bit-for-bit reconstruction.

Consider the following set of samples from a 1D signal:

$$c_3[n] = [1\ 3\ 3\ 5\ 9\ 7\ 7\ 5]$$

The eight samples can be approximated with pair-wise, unweighted averaging to produce signal at half the resolution:

$$c_2[\frac{n}{2}] = [2\ 4\ 8\ 6].$$

The samples in $c_2[\frac{n}{2}]$ are constructed by

$$c_{j-1}[i] = \frac{1}{2}(c_j[2i+1] + c_j[2i]). \tag{8.4}$$

Now consider the signal constructed with

$$d_{j-1}[i] = \frac{1}{2}(c_j[2i+1] - c_j[2i]). \tag{8.5}$$

So, for the $d_2[\frac{n}{2}]$ example:

$$d_2[\frac{n}{2}] = [1\ 1\ -1\ -1].$$

The signal $d_2[\frac{n}{2}]$, when combined with $c_2[\frac{n}{2}]$, provides a way to reconstruct the original $c_3[n]$:

$$\begin{aligned} c_j[2i] &= c_{j-1}[i] - d_{j-1}[i], \\ c_j[2i+1] &= c_{j-1}[i] + d_{j-1}[i]. \end{aligned} \tag{8.6}$$

The samples in c_j are referred to as *approximation* coefficients, and those in d_j, which represent the differences or the error introduced by averaging two neighboring data values, are referred to as *detail* coefficients. The calculations given by Eqs. (8.4, 8.5) can be recursively applied to c_j until a single approximation coefficient is left—the average of all the samples in our original signal—and, in this example, seven detail coefficients:

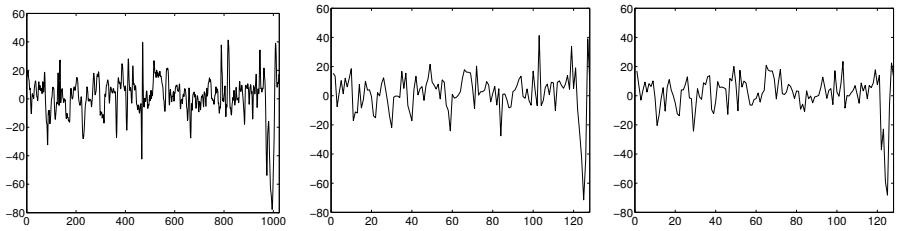
$$c_0, d_{0\ldots2} = [5\ 2\ 1\ -1\ 1\ 1\ -1\ -1]$$

Thus, a linear transform has been applied to the original signal that allows a reconstruction of an approximation with 1, 2, 4, or 8 samples, but without any storage overhead. In the context of data read from disk, the coarsest approximation, c_0, can be retrieved by reading a single sample, the next approximation level, c_1, by reading one additional coefficient, d_0, and applying Eq. (8.6), and the next, by reading two more coefficients, and so on. Multiple dimensions can be easily generalized by averaging neighboring points along each dimension to generate approximation coefficients, and generate detail coefficients by taking the difference between the approximation coefficients and each of the points used to find its average.

Mathematically, this is the one-dimensional, unnormalized, Haar wavelet transform. The Haar wavelet transform, and other wavelet transforms that are discussed later, constructs a multiresolution representation of a signal. In one dimension, each multiresolution level j contains, on average, half the samples of the $j+1$ level approximation, with coarser level samples created by averaging finer level samples. Note, while the Haar transform exhibits what is known in the digital signal processing world as the property of *perfect reconstruction*, due to limited floating point precision the signal reconstructed from the c_j and d_j coefficients will not, as was the case with the Z-order curve, be an exact copy of the original signal. Moreover, even if the input signal consists entirely of the integer data the division by two in Eq. (8.4) will result in floating point coefficients. However, integer-to-integer wavelet transforms do exist [2], but they are not discussed here.

Figure **??** shows a qualitative comparison of a 1D sampling of the x-component of vorticity obtained from a 1024^3 simulation by Taylor Green turbulence [7]. The original signal is seen in **??**(a). The signal generated by nearest neighbor sampling (analogous to Z-order curve approximation) at $1/8^{th}$ resolution is shown in **??**(b), and at the same coarsened resolution, approximated with the Haar wavelet (**??**c).

Eqs. (8.4)-(8.6) provide everything that is needed to support a PDA scheme based purely on a multiresolution hierarchy, not suffering from the limitations of the Z-order curve, but giving up bit-for-bit identical reconstruction. One drawback of both the Z-order curve and the wavelet based multiresolution approach is that they fail to take advantage of any coherence in the data. Better approximations of a function can be achieved for a given number of coefficients, by further exploiting some of the properties of wavelets.



(a) (b) (c)

FIGURE 8.2: PDA based on multiresolution. The original signal containing 1024 samples (a), reduced to $1/8^{th}$ resolution using: subsampling, analogous to the Z-order curve (b); and the Haar wavelet approximation coefficients (c).

8.4.1 Linear Decomposition

Often it is advantageous to represent a signal or function f as a superposition:

$$f(t) = \sum_k a_k u_k(t) \qquad k \in \mathbb{Z}, a \in \mathbb{R}, \tag{8.7}$$

where a's are *expansion coefficients*, and $u_k(t)$'s are a set of functions of time, t, that form a basis for L^2—the space of square, integrable (finite energy) functions. If, for example, u_k's are complex exponentials, then the expansion is a Fourier series of f. More generally, if the basis functions, $u_k(t)$, are orthogonal, that is if

$$\langle u_k(t), u_l(t) \rangle = \int u_k(t) u_l(t) dt = 0 \quad l \neq k \tag{8.8}$$

then the coefficients a_k can be easily calculated by the inner product:

$$a_k = \langle f(t), u_k(t) \rangle = \int f(t) u_k(t) dt. \tag{8.9}$$

The wavelet expansions of signals are also superpositions of basis functions but they are represented as a two-parameter expression:

$$f(t) = \sum_k c_{j_{min},k} \phi_{j,k}(t) + \sum_j \sum_k d_{j,k} \psi_{j,k}(t), \tag{8.10}$$

where $c_{j,k}$ and $d_{j,k}$ are again real-value coefficients. $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ are the *scaling* and *wavelet* basis functions, respectively. The parameterization of time (or space) is provided by k, while frequency or scale, not coincidentally, is parameterized by j.

Several properties of wavelets and the above wavelet expansion are applicable to all wavelets discussed in this section:

1. The wavelet expansion computes efficiently. For discrete $f(t)$, the upper

bound on the parameter j is $log_2(N)$, where N is the number of samples in f. Similarly, for many wavelets both $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ have *compact support* and are thus zero-valued outside of a small interval of t. Thus the expansion is computed with only a handful of multiplications and additions, making its complexity $O(N)$. The same is true for the wavelet transform used to compute $c_{j,k}$ and $d_{j,k}$. Compare this to $O(Nlog_2(N))$ complexity for the Fourier expansion.

2. Unlike the Fourier transform, the wavelet transform localizes in time (space) the frequency information of a signal. Most of the information or energy of a signal is represented by a small number of expansion coefficients (a property exploited later for progressive data access).

3. Finally, Eq. (8.10) provides a multiresolution decomposition of f. The first term of the right-hand side of the equation describes a coarsened, low-resolution approximation of f, while the second term, for increasing j, provides finer and finer details—that represent the missing information not contained in the first term. Another view is that the first term contains the low frequency components of f, while the second term provides the high frequency components of f. Hence, the labels are applied: *approximation* and *detail* to the coefficients, c_j and d_j, respectively.

8.4.2 Scaling and Wavelet Functions

Up to this point, scaling and wavelet functions have been discussed without formally defining them. Unlike the Fourier basis functions, the complex exponentials, the wavelet and scaling functions are an infinitely large family of functions. Here, the discussion will strictly pertain to normalized, orthogonal, or *orthonormal*, families of wavelets as defined by the relations:

$$\left.\begin{aligned} \langle \phi_{j,k}(t), \phi_{j,l}(t) \rangle &= \delta(k,l) \\ \langle \psi_{j,k}(t), \psi_{j,l}(t) \rangle &= \delta(k,l) \\ \langle \phi_{j,k}(t), \psi_{j,l}(t) \rangle &= 0 \end{aligned}\right\} \quad \text{for all } j,k, \text{ and } l, \tag{8.11}$$

where $\delta(k,l)$ is 1 for $k = l$, otherwise $\delta(k,l)$ is 0.

The multiresolution properties of the wavelet expansion arise from the scaling function and the ability to recursively construct this basis function from a canonical version of itself. That is

$$\phi(t) = \sum_n h_0[k]\sqrt{2}\phi(2t - n) \quad n, k \in \mathbb{Z}. \tag{8.12}$$

Eq. (8.12) is a *dilation* equation; it shows how to construct the scaling function, $\phi(t)$, from the superposition of scaled, translated, and dilated copies of itself. The translation $(t - n)$ shifts $\phi(t)$ to the right. The multiplier $2t$ narrows (dilates) $\phi(t)$, and $\sqrt{2}$ scales the function. In other words, $\phi(t)$ is

expressed as a linear expansion, similar in form to Eq. (8.7), but by using itself as a basis! By convention, the expansion coefficients are relabeled from a_k to $h_0[k]$.

For the Haar basis function discussed in the beginning of 8.4, the canonical form of $\phi(t)$ is:

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & otherwise \end{cases} \tag{8.13}$$

Thus, the Haar scaling function is a box of unit height over the interval $[0, 1)$. The expansion coefficients, $h_0[k]$, for the Haar scaling function are $h_0[0] = h_0[1] = 1/\sqrt{2}$.

Using shifted, scaled and dilated versions of the scaling function $\phi(t)$ alone, we can construct a multiresolution hierarchy of $f(t)$. However, much like the first attempt with the Z-order curve, this leads to more replications. For these and other reasons that are beyond the scope of this text, the wavelet expansion includes a second basis function, $\psi(t)$, that provides a means to capture the information that is missing from the approximation based on $\phi(t)$.

Similar to the scaling functions, the wavelets are also defined by the canonical scaling function:

$$\psi(t) = \sum_n h_1[k]\sqrt{2}\phi(2t - n). \tag{8.14}$$

Here, h_1 are a different set of expansion coefficients from h_0. Without proof, the orthogonal wavelets h_0 and h_1 are related to each other by

$$h_1[n] = (-1)^n h_0(N - 1 - n), \tag{8.15}$$

where N is the support size of h_0.

Thus, for the Haar wavelet, the non-zero coefficients in Eq. (8.14) are $h_1[0] = 1/\sqrt{2}$ and $h_1[1] = -1/\sqrt{2}$. The canonical Haar wavelet function is therefore:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & 1/2 \leq t < 1 \\ 0 & otherwise \end{cases} \tag{8.16}$$

A multitude of wavelet families exist, each with a variety of parameterizations. A reasonable question to ask is: If Haar wavelets possess all of the multiresolution properties required, why consider other wavelets? Again, an in-depth treatment is beyond the scope of this text, and the remarks will be limited to those properties related only to the goals of this discussion.

The tendency of wavelet transform expansion coefficients is to quickly become small in magnitude [4]. Many signals can be accurately approximated by only a small number of coefficients (see 8.4.4). The Haar wavelets, constructed from box functions, are the most efficient in representing signals that

are piecewise-constant; spans of the signal that are constant-valued between dyadic points (multiples of powers of two) can be represented by approximation coefficients alone, and without any need for detail coefficients. While smoother signals, in fact, any signal in $L2$, can be represented by the Haar basis, the number of non-zero detail coefficients will, in general, be large. To represent signals sparsely, a basis is required whose properties, in some way, match those of the signal. One measure of the wavelet function's ability to match a signal is given by the number of vanishing moments, p, where,

$$\int x^k \psi(t) dt = 0 \quad \text{for } 0 \leq k < p, \tag{8.17}$$

which shows that ψ is orthogonal to any polynomial of degree $p-1$. If f resembles a Taylor polynomial of degree $k < p$ over some interval then the detail coefficients of the wavelet transform of f will be small at fine scales 2^j. For this reason much research has gone into the design of wavelets that have support of minimum size for a given number of vanishing moments, p. Not surprisingly, within a family of wavelets, increasing vanishing moments comes at the cost of wider support.

8.4.3 Wavelets and Filter Banks

Now that the wavelets and wavelet transforms have been reviewed, this subsection explores the application of progressive data access from a digital filter bank point of view. The discussion of this point of view is necessary because the digital filter bank matches the discrete, gridded data better than the function expansion point of view. Now, instead of a continuous function $f(t)$, consider a discretely sampled, finite function, $x[n]$, $0 \leq n < N$, where N is the total number of samples. To compute the wavelet expansion coefficients of Eq. (8.10) there is no need to actually evaluate the scaling or wavelet function. The normalized scaling and wavelet expansion coefficients of Eqs. (8.12) and (8.14)—h_0 and h_1—are the only coefficients needed to compute $c_{j,k}$ and $d_{j,k}$. Without proof, the following relations are known as the Discrete Wavelet Transform (DWT):

$$c_{j,k} = \sum_n \tilde{h}_0[2k-n] c_{j+1}[n], \tag{8.18}$$

$$d_{j,k} = \sum_n \tilde{h}_1[2k-n] c_{j+1}[n], \tag{8.19}$$

where $\tilde{h}_0(n) = h_0(-n)$ and $\tilde{h}_1(n) = h_1(-n)$. These equations show that the approximation and detail coefficients for a level j are computed via a simple two channel filter bank, recursively applied in a process known as *analysis* (see Figure **??**a). Analysis is equivalent to convolving the input signal, c_{j+1}, with time-reversed copies of the filter coefficients, $h_0[-n]$ and $h_1[-n]$, and then downsampling the filtered signal by discarding every odd term. The analysis

filter implemented by h_0 is a low-pass filter, and the one implemented by h_1 is a high-pass filter. Thus the filters split the signal into its low frequency and high frequency components. The cascade of outputs formed by each iteration, or pass, of the filter pairs divides the frequency spectrum into a logarithmic sequence of bands. By downsampling the results of each convolution, the total number of non-zero coefficients is held constant through each pass.

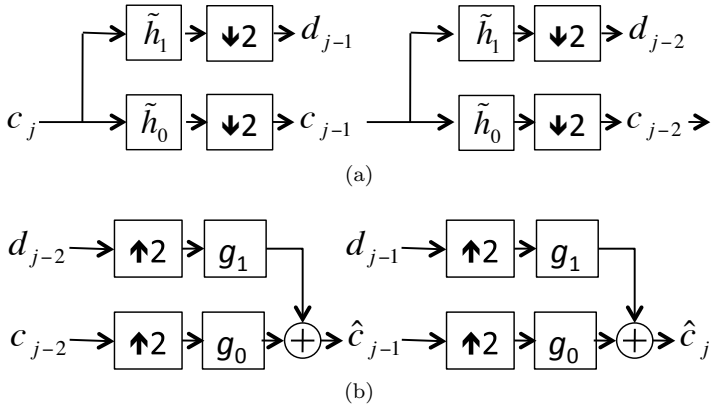


FIGURE 8.3: The *analysis* of the input signal, c_j, is performed by a cascade of convolutions, followed by downsampling steps in (a). Reconstruction of c_j is performed by a cascade of upsampling, followed by convolution steps in a process known as *synthesis* in (b).

A reconstructed signal, $\hat{c}_{j,k}$, is obtained with the Inverse Discrete Wavelet Transform (IDWT) in a process known as *synthesis*, where,

$$\hat{c}_{j,k} = \sum_n g_0[k-2n]c_{j-1}[n] + g_1[k-2n]d_{j-1}[n]. \tag{8.20}$$

Analogous to Eqs. (8.18) and (8.19) the IDWT is equivalent to upsampling c_j and d_j by the introduction of zeros in every other sample and summing the convolutions of c_j and d_j with g_0 and g_1, respectively, as illustrated in Figure **??**(b). To ensure that $\hat{c}_{j,k} = c_{j,k}$, the inverse transform must exhibit the property of *perfect reconstruction*. The reconstruction filters, g_0 and g_1, must be chosen accordingly. But how does one know that g_0 and g_1 even exist? Conveniently, for orthogonal filters the reconstruction (synthesis) filters do exist and are related to the analysis filters by:

$$\begin{aligned} g_0[n] &= h_0[n] = \tilde{h}_0[N-n], \\ g_1[n] &= h_1[n] = \tilde{h}_1[N-n]. \end{aligned} \tag{8.21}$$

An obvious question that arises is this: Given a signal $x[k]$, how are the initial $c_{j_{max},k}$ computed? Fortunately, if the samples of $x[k]$ are computed above the Nyquist rate, it can be assumed that $c_{j_{max},k} \approx x[k]$.

8.4.4 Compression

Now, all the machinery is in place to provide a multiresolution representation of a 1D sampled signal. The DWT transforms $x[n]$ into a wavelet space sequence of coefficients, $c_{j,k}$ and $d_{j,k}$. From the wavelet space representation there is a coarsened approximation of $x[n]$ given by $c_{j=0}$ that can progressively be refined by adding a resolution by way of Eq. (8.20).

The real power of the wavelet transform comes from the aforementioned ability of wavelets to concentrate energy into a small number of coefficients. Assume a discrete signal, $x[n]$, expressed as the expansion

$$x[n] = \sum_{k=0}^{K-1} a_k u_k[n],$$

for some set of compactly supported basis functions u_k. The goal is to find an approximation of $x[n]$, such that,

$$\tilde{x}[n] = \sum_{k=0}^{\tilde{K}-1} \tilde{a}_k u_k[n],$$

where $\tilde{K} < K$. Moreover, for a given $\tilde{K}$, it is desired that the approximation $\tilde{x}[n]$ to be the best possible approximation for $x[n]$ by some error metric. One possibility is $||x[n] - \tilde{x}[n]||_p < \epsilon$ for some norm, p. For an orthonormal basis, for $p = 2$,

$$||x[n] - \tilde{x}[n]||_2^2 = \sum_{k=\tilde{K}}^{K-1} (a_{\pi(k)})^2, \tag{8.22}$$

where $\pi(k)$ is a permutation of $0 \ldots K-1$ such that

$$|a_{\pi(0)}| \geq \ldots \geq |a_{\pi(K-1)}|,$$

and $\tilde{x}$ uses the coefficients corresponding to the first $\tilde{K}-1$ elements of $\pi(k)$. Formally:

$$\tilde{a}_k = \begin{cases} a_k & \text{if } \pi(k) < \tilde{K}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the L^2 error of $\tilde{x}[n]$ is given by the sum of the square of the coefficients $a[k]$ that are not included in the expansion of $\tilde{x}[n]$. To minimize the L^2 error for a given $\tilde{K}$, a_k is sorted by decreasing the magnitude and only the $\tilde{K}-1$ largest coefficients are included in the approximation. Alternatively, if a particular ϵ is wanted, Eq. (8.22) allows the calculation of how many and which coefficients can be discarded to ensure $||x[n] - \tilde{x}[n]||_p < \epsilon$. Note, for any $a[k] = 0$ its exclusion introduces no error in the approximation.

Eq. (8.22) sheds some light on the choice of wavelet. As discussed earlier

(see Eq. 8.17), a wavelet possessing a higher number of vanishing moments will permit the compaction of more energy (information) into fewer expansion coefficients. Figure 8.4 qualitatively compares the compression of a signal by a factor of 8, by using wavelets with different numbers of vanishing moments. For ease of comparison, Figures 8.4(a) and 8.4(b) reproduce Figures (**??**(a) and **??**(b), respectively. Results using the Haar Figure 8.4(c) and Daubechies D4 Figure 8.4(d), wavelet are shown with 1 and 2 vanishing moments, respectively. Note the "blockiness" resulting from the box shape of the Haar wavelet. Also note that while the multiresolution approximation of Figure 8.4(b) may appear visually more appealing than Figure 8.4(c), many extreme values in the original signal are lost, which is the result of the cascade of applications of the low-pass scaling filter.

Finally, unlike the pure multiresolution approaches based on Z-order curve and wavelets, an arbitrary number of approximations are now produced. In the extreme case, the approximation is refined one coefficient at a time.

8.4.5 Boundary Handling

Finite length signals present a couple of challenges that have been ignored until this point. Consider the application of the DWT to a signal $x[n]$ with a low-pass filter h of length $L = 4$. By Eq. (8.18) the computation of $c[0]$ is given by

$$c[0] = \tilde{h}[3]x[-3] + \tilde{h}[2]x[-2] + \tilde{h}[1]x[-1] + \tilde{h}[0]x[0].$$

But, for a finite signal, the samples $x[-3]$, $x[-2]$, and $x[-1]$ do not exist! One simple solution is to extend $x[n]$ so that it is defined for $n < 0$ and $n \geq N$. There are a number of choices for the extension values, such as zero padding or repeating the boundary samples, each with its own set of trade-offs. In the general case, however, if the input signal, x, is extended, creating a new input signal, x_e of length $N_e > N$, the resulting output after filtering and downsampling no longer has exactly N non-zero and non-redundant approximations and detail coefficients. Moreover, perfect reconstruction of x_e requires all of the N_e coefficients. Due to downsampling in the case of the DWT, each iteration of the filter bank requires an extension of the incoming approximation coefficients. For a 1D signal, this overhead may not be significant, but later, when moving on to 2D and 3D grids, the additional coefficients can consume a substantial amount of space.

A filter bank that is *nonexpansive* is more preferable, as it preserves the number of input coefficients on output, and does not introduce discontinuities at the boundaries. The solution is the employment of symmetric filters combined with a symmetric signal extension. Only the case of signals of length $N = 2^n$, and odd length filters where the symmetry is about the center boundary coefficients, is considered here. For an extensive discussion on symmetric filters of even length, which introduce considerably more complexity, or handling $N \neq 2^n$, see [12, 5]. For clarity of exposition, assume that the filter is

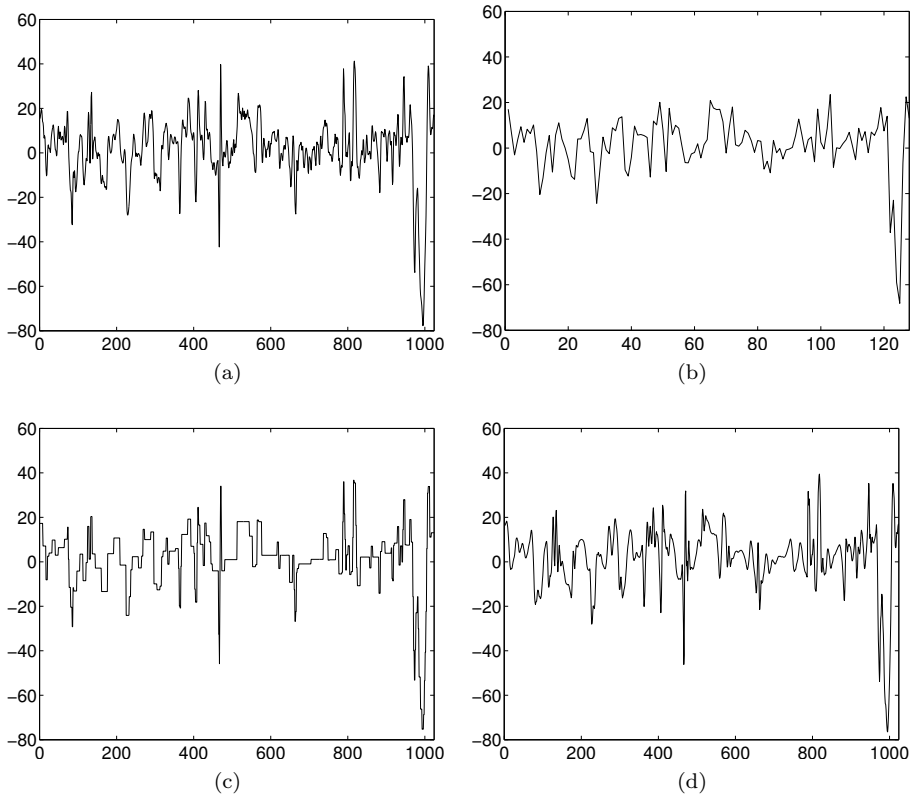


FIGURE 8.4: A test signal (a) with 1024 samples and a multiresolution approximation, (b), at $1/8^{th}$ resolution, reproduced from Figure **??**. The test signal reconstructed using $1/8^{th}$ of the expansion coefficients with the largest magnitude from the Haar, (c), and Daubechies 4-tap wavelet, (d), respectively.

whole-sample symmetric about $h[0]$. That is, $h[n] = h[-n]$, and $h[0]$ is not repeated. For example, if $L = 3$, the center point is $h[0]$, and $h[-1] = h[1]$. Moreover, the input signal must be made a whole-sample symmetric about the first ($n = 0$) and last ($n = N - 1$) samples. The motivation for this symmetry is straightforward: the output samples for $n < 0$ and $n \geq N$ are redundant and need not be explicitly stored. Consider the calculation in a general linear transform of the expansion coefficient $a[-1]$, by the convolution of x with $h[-n]$ for whole-sample symmetric x, and h, with $L = 3$:

$$a[-1] = h[0]x[-2] + h[1]x[-1] + h[2]x[0].$$

Due to symmetry, $h[n] = h[-n]$, and $x[n] = x[-n]$. Therefore:

$$a[1] = a[-1] = h[0]x[2] + h[1]x[1] + h[2]x[0],$$

and $a[-1]$ is redundant and need not be explicitly stored!

Things become a little more complicated with the DWT, which operates as a dual channel convolution filter, followed by downsampling. Here, centering the filter must be done carefully for each channel, such that symmetry is preserved after downsampling, and the total number of coefficients output by the two channels, equals the number of input coefficients. For even N, each channel outputs exactly $N/2$ samples.

Symmetric filters combined with a symmetric signal extension provide a straightforward mechanism for dealing with finite length signals and the DWT. Unfortunately, with the exception of the Haar wavelet, there are no orthogonal wavelets with compact support possessing both the property of symmetry and perfect reconstruction. The solution to this dilemma is the relaxation of the orthogonality requirement and the introduction of *biorthogonal* wavelets. For biorthogonal wavelets, the properties of Eq. (8.11) no longer hold. Different *analysis* scaling and wavelet basis functions, $\tilde{\phi}(t)$ and $\tilde{\psi}(t)$, and *synthesis* scaling and wavelet basis, $\phi(t)$ and $\psi(t)$, must be introduced. Similarly, new analysis, $\tilde{h_0}$ and $\tilde{h_1}$, and synthesis filters, g_0 and g_1, will appear.

From a filter bank perspective, h_1 no longer relates to h_0 by a simple expression. However, the following cross relationship between synthesis and analysis filters hold:

$$\tilde{h}_0[n] = (-1)^n g_1(N - 1 - n), \quad g_0[n] = (-1)^n \tilde{h}_1(N - 1 - n), \tag{8.23}$$

where N is the support size of the filter.

With the analysis filters no longer related to each other by Eq. (8.15), the support of these respective filters need not be the same. The support sizes of $\tilde{h}_0$ and $\tilde{h}_1$ are then denoted as L_0 and L_1, respectively, leading to new analysis equations:

$$c_{j,k} = \sum_{n=-(L_0-1)/2}^{(L_0-1)/2} \tilde{h}_0[n] c_{j+1}[2k - n] \tag{8.24}$$

$$d_{j,k} = \sum_{n=-(L_1-1)/2}^{(L_1-1)/2} \tilde{h}_1[n] c_{j+1}[2k-n+1] \tag{8.25}$$

Note that Eq. (8.24) processes the even samples of c_{j+1}, while Eq. (8.25) processes the odd samples—a necessity for nonexpansive output [12].

Because of the shift in the inputs to the analysis equation, a new synthesis equation is also necessary, which must shift the $d[n]$ coefficients:

$$\hat{c}_{j,k} = \sum_n g_0[k-2n] c_{j-1}[n] + g_1[k-2n-1] d_{j-1}[n] \tag{8.26}$$

As already noted, the correct behavior of Eqs. (8.24) and (8.25) is predicated on treating the input signal, $c_{j+1}[n]$, as exhibiting whole-sample symmetry about the left and right boundary. Perfect reconstruction from Eq. (8.26) requires a mixture of whole-sample and *half-sample* symmetry, where the point of symmetry lies halfway between two samples. Signals with the left boundary, half-sample symmetry are symmetric about the non-integer point $-\frac{1}{2}$, while right boundary, half-sample symmetric signals are symmetric about the point $N - \frac{1}{2}$—for the left boundary, $x[-1] = x[0]$, $x[-2] = x[1]$ and so on. For Eq. (8.26), the left and right boundaries of $c[n]$ and $d[n]$ must be made whole-sample symmetric, respectively, while the right and left boundaries of $c[n]$ and $d[n]$ must be half-sample symmetric, respectively. Lastly, for both the symmetric analysis filters, $\tilde{h}_0[n]$ and $\tilde{h}_1$[n], and the symmetric synthesis filters, g_0[n] and g_1[n], the filters have a zero value for $n < (L-1)/2$ and $n > (L-1)/2$, where L is the support size of the filter.

Although symmetry is gained and the number of inputs and output are preserved, by giving up orthogonality, other important properties are lost. Most significantly, Eq. (8.22) no longer holds, and the L^2 error between a signal x and its approximation $\tilde{x}$ can no longer be determined by the coefficients (not included in the construction of $\tilde{x}$). Nevertheless, the L^2 error, introduced in the reconstruction after discarding coefficients, is still minimized by discarding the coefficients of smallest magnitude.

Figure 8.5 shows plots of the Cohen-Daubechies-Feauveau (CDF) 9/7 biorthogonal synthesis and the analysis functions. Note the symmetry in all of the functions. Table 8.1 provides the filter coefficients for the CDF 5/3 and 9/7 normalized, biorthogonal wavelets. These filters are the foundation for the JPEG2000 image compression standard. The reader is cautioned that the naming of the CDF family of biorthogonal wavelets is inconsistent in the literature. Here, the naming convention based on the support size in the low-pass analysis and synthesis filters, respectively (e.g., CDF 9/7), is adopted. Other authors use a naming scheme based on the number of analysis and synthesis filter vanishing moments, respectively (e.g., bior4.4).

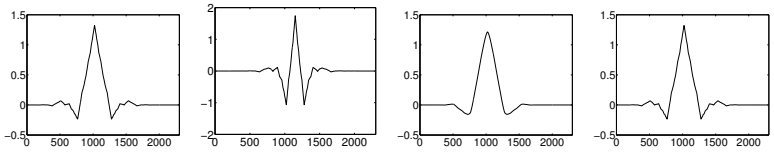


FIGURE 8.5: The CDF 9/7 biorthogonal wavelet and scaling functions: analysis scaling (a) and wavelet functions (b), and synthesis scaling (c) and wavelet functions (d).

TABLE 8.1: Biorthogonal wavelet filter coefficients for the CDF 5/3 (top) and 9/7 (bottom) wavelets

p	**n**	$\mathbf{h_0[n]}$	$\mathbf{g_0[n]}$
2	0	1.06066017177982	0.70710678118655
	$-1, 1$	0.35355339059327	0.35355339059327
	$-2, 2$	-0.17677669529663	0
4	0	0.852698679008894	0.788485616405583
	$-1, 1$	0.377402855612831	0.418092273221617
	$-2, 2$	-0.110624404418437	-0.0406894176091641
	$-3, 3$	-0.023849465019557	-0.0645388826286971
	$-4, 4$	0.037828455507264	0.0

8.4.6 Multiple Dimensions

Extending the 1D wavelet filter bank to multiple dimensions is straightforward. The 1D analysis filter is simply applied along each dimension is illustrated in the 2D example in Figure 8.6. Thus, for a $M \times N$ grid, a single pass of the 2D DWT yields, on average $\frac{MN}{4}$ approximation coefficients and $\frac{3MN}{4}$ detail coefficients. For 3D data, seven times as many detail coefficients as approximation coefficients are generated for each iteration of the DWT.

8.4.7 Implementation Considerations

There are two possible approaches used to construct a PDA model, based on the forward and inverse DWT. A multiresolution hierarchy can be constructed, just like with the Z-order space-filling curve, and by exposing the scale parameter, j, in Eq. (8.10) the grid may be coarsened or refined by factors of 2^d. This approach is called *frequency truncation*. Each iteration of the analysis filters uses the normalized coefficients, provided in Table 8.1, which scales the amplitude of the approximation coefficients by $\sqrt{2.0}$. If the approximation coefficients are used as an approximation of the original signal, this scaling should be undone by multiplication by $2^{-1/j}$, where j is the number of iterations of the analysis filter. Alternatively, the wavelet expansion represen-

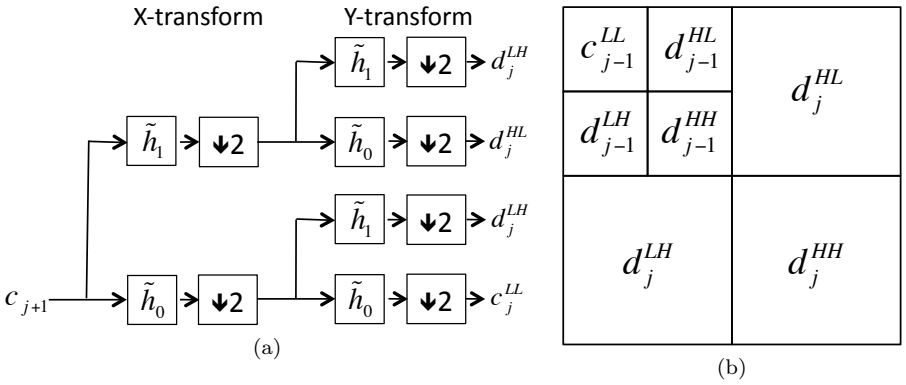


FIGURE 8.6: On the left, (a) shows a single pass of the 2D DWT resulting in one set of approximation coefficients, and three sets of detail coefficients. The right side of the figure (b) shows the resulting decomposition after two passes of the DWT. As each coefficient is the result of two filtering steps, one along each dimension, the superscripts in (a) and (b) indicate highpass, H, or lowpass, L, filtering.

tation's power can be exploited to concentrate most of a signal's information content into a small number of expansion coefficients. This approach is called *coefficient prioritization*. With either method, a number of practical implementation issues arise.

8.4.7.1 Blocking

The multi-dimensional DWT, discussed in 8.4.6, can be applied directly to a d-dimensional grid. However, for high-resolution grids, there are a number of reasons for considering decomposing the grid into a collection of smaller blocks and applying the DWT to each individual block. Computational efficiency is one consideration. Significantly improved performance may be achieved on cache-based microprocessors by operating on a collection of smaller blocks that better accommodate the memory hierarchy rather than one large, single volume. The second reason, and perhaps the more compelling one, is efficient region extraction. For many visualization workflows—such as drawing a cutting plane through a 3D volume or zooming in and volume rendering a small subregion of a larger volume—only a subset of the data is required. By decomposing the volume into blocks, it is possible to access (from storage) only those blocks that intersect the region of interest. The merits of data blocking with regard to I/O are discussed in *Hierarchical and Geometrical Methods in Scientific Visualization* [9].

The best choice of block size is somewhat application dependent. Given the dyadic nature of the wavelet, and the complexities of boundary handling

discussed in 8.4.5, power-of-two dimensions are sensible. As each pass of the DWT reduces the number of approximation coefficients by half along each dimension, larger block sizes permit more passes of the DWT resulting in a deeper multiresolution hierarchy. A deeper hierarchy offers more refinement levels for a progressive access scheme based on frequency truncation. Similarly, if coefficient prioritization is employed, more coefficients present greater degrees of freedom for compression. However, if the block size is too large, the goal of performing efficient region extraction and computational performance may be defeated. As a reasonable compromise for 3D grids, the suggested block sizes are 64^3 or 128^3.

For data that is not block-aligned, padding of the boundary blocks is required. Care should be taken to use an extension strategy that does not introduce sharp discontinuities. Constant, linear extrapolation, or symmetric extension, generally produces reasonable results.

8.4.7.2 Wavelet Choice

The advantages of symmetric, biorthogonal wavelets have been discussed; and filter coefficients for two wavelets commonly used have been provided in image compression applications in 8.4.5. Other biorthogonal wavelets possessing more vanishing moments and, therefore, greater information compaction capability also exist at the cost of additional filter taps. It is worth noting that while filters with better energy compaction capabilities may yield sparser representations, the additional filter coefficients incur additional computational cost. Also, the wider the filter the greater the number of input coefficients required, which may limit the possible number of cascades of the DWT. For example, with a block of dimension 64^3 after three stages of the DWT, there are 8^3 approximation coefficients—too few for another pass with the nine-tap CDF9/7 filter, but sufficient for one more stage of the narrower, five-tap CDF5/3 filter.

8.4.7.3 Coefficient Addressing

An important attribute of regular grids is the implicit addressing of grid vertices. For example, each vertex in the grid can be addressed by an integer index, (i, j, k) in 3D, whose offsets can be implicitly determined by the serialized storage order. Therefore, only the sampled field value for each grid point needs to actually be stored. The implicit addressing of expansion coefficients is easily preserved during storage with frequency truncation. The coefficients are simply written in the order that they are output from each stage of the filter bank. With coefficient prioritization, however, the coefficients must be ordered by their information content. Their addresses are no longer implicit by their position in the output stream, and must be explicitly preserved. The number of bits required to uniquely address each of the N expansion coefficients is $log_2(N)$, which introduces a sizeable storage overhead.

Consider, however, binning the N expansion coefficients output by the DWT (which will generically refer to as $c[k]$) into a collection of P sets,

$$S_p = \{c_{\pi(i)} | C(p-1) \leq i < C(p), 0 \leq i < N-1\}, \tag{8.27}$$

where $C(0) = 0$, $C(p \geq 1) = \sum_1^p |S_p|$, and, as before $\pi(i)$ is a permutation of $0 \ldots N-1$ such that $|c_{\pi(0)}| \geq \ldots \geq |c_{\pi(N-1)}|$. Then, if within each set S_p, the expansion coefficients are stored in their relative order of output from the filter bank, the addresses of the coefficients need only be stored for sets $S_1 \ldots S_{P-1}$. The addresses of the elements of S_P can be inferred from the addresses of $S_1 \ldots S_{P-1}$. By keeping track of the ordered set of addresses not found in $S_1 \ldots S_{P-1}$, the lowest address not found in $S_1 \ldots S_{P-1}$ is the address of the first coefficient found in S_P and so on. If $|S_P|$ is sufficiently large, the storage savings can be considerable.

8.4.8 A Hybrid Approach

Frequency truncation and coefficient prioritization each have their strengths and weaknesses. Frequency truncation preserves the implicit ordering of grid vertices, thereby imposing minimum storage overhead, but it offers only a very coarse grain control over the the level of detail available; incrementing or decrementing the j scaling parameter by one changes the number of grid points by a factor 2^d. Furthermore, unlike coefficient prioritization, all coefficients are treated equally regardless of their contribution to the signal. On the other hand, though, coefficient prioritization allows for arbitrary control over level of detail, and generally offers better approximations for a given bit budget, but prioritization requires additional storage to keep track of coefficient addresses. Perhaps, a less obvious difference between the two schemes is that, by virtue of possessing fewer grid points, multiresolution approximations (frequency truncation) benefit all the resources of the visualization pipeline including: physical memory, floating point calculations, and I/O. Frequency truncation, on the other hand, only benefits the transmission of the data (e.g., access to secondary storage). Once a signal is reconstructed—albeit even if from fewer expansion coefficients than samples in the original signal—the number of samples in the reconstructed signal is the same as for the original signal. After the reconstruction is finished, there is no further realized benefit of the wavelet expansion.

The merits of both methods may readily be combined with little or no additional effort. By allowing control over both the j parameter in Eq. (8.26), which controls the resolution, and (if the coefficients are binned by their information content) the p parameter, then, Eq. (8.27) controls the expansion coefficients used in reconstruction.

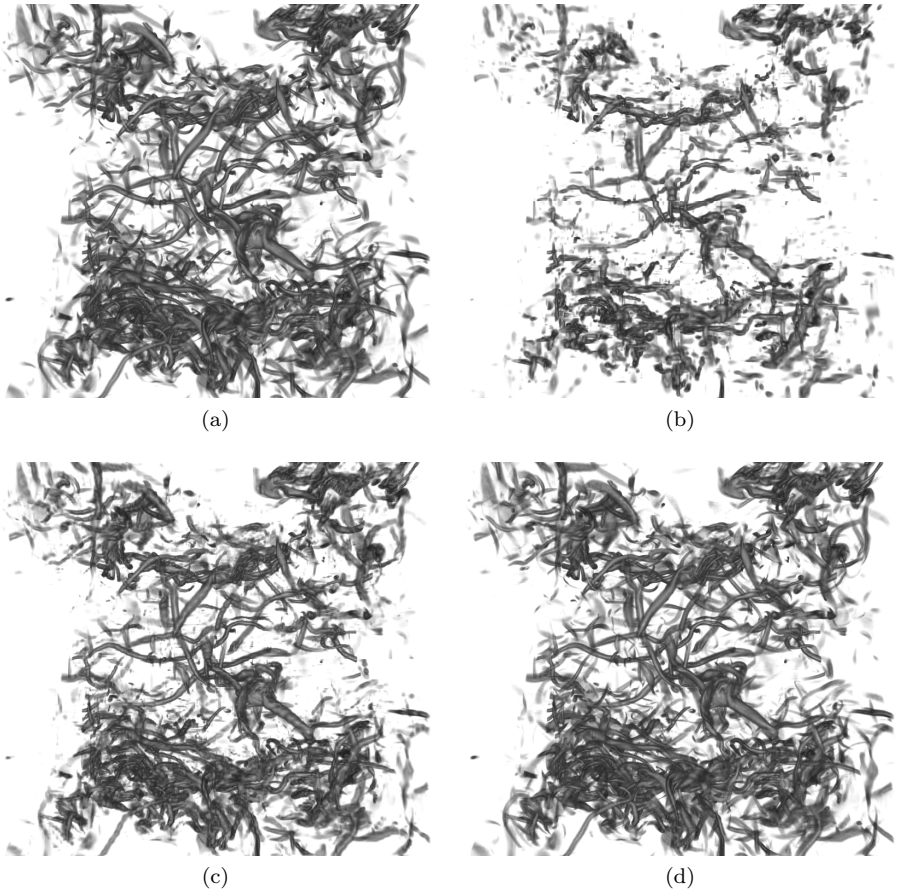
8.4.9 Volume Rendering Example

Figure 8.7 shows at varying approximations a direct volume rendering of an enstrophy field derived from a subregion of a 1024^3 isotropic and homogeneous Taylor Green (TG) incompressible turbulence simulation [7]. The original data is shown in Figure 8.7(a). Approximations based on coefficient prioritization, using reduction factors of $\frac{1}{500}$, $\frac{1}{100}$, and $\frac{1}{10}$, are shown in Figures 8.7(b)-8.7(c), respectively. The data was decomposed into 128^3 blocks, and transformed with the CDF9/7 wavelet.

8.5 Further Reading

The presentation on Z-order curves and their application in progressive data access is based almost entirely on the work of Pascucci and Frank. We refer the reader to their papers for further details on the method, suggested implementation strategies, and a discussion of their results with real data sets and applications [10, 9].

Wavelets are a relatively new field in mathematics with a wealth of applications related to data analysis that go far beyond progressive data access. Here we have only scratched the surface on their theory and their capabilities. For the interested reader, an excellent introduction on wavelets and wavelet transforms may be found in the books by Burrus et al. [1] and Stollnitz, et al. [11]. For a deeper understanding we recommend the authoritative books by Mallat [6], and Strang and Nguyen [12], and the seminal work by Daubechies [3].



(a) (b)

(c) (d)

FIGURE 8.7: Direct volume rendering of an enstrophy field: original data (a), and shown in (b)-(d), respectively, are results after reduction of the number of expansion coefficients used in reconstruction by factors of $\frac{1}{500}$, $\frac{1}{100}$, and $\frac{1}{10}$.

Bibliography

[1] Sidney C. Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, August 1997.

[2] A. R. Calderbank, Ingrid Daubechies, Wim Sweldens, and Boon-Lock Yeo. Wavelet transforms that map integers to integers. *Appl. Comput. Harmon. Anal.*, 5(3):332–369, 1998.

[3] Ingrid Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, June 1992.

[4] D. L. Donoho. Unconditional bases are optimal bases for data compression and for statistical estimation. *Applied and Computational Harmonic Analysis*, 1(1):100–115, 1993.

[5] Christopher M. and Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Applied and Computational Harmonic Analysis*, 3(4):337 – 357, 1996.

[6] Stephane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3 edition, December 2008.

[7] P. D. Mininni, A. Alexakis, and A. Pouquet. Large-scale flow effects, energy transfer, and self-similarity on turbulence. *Phys. Rev. E*, 74:016303, Jul 2006.

[8] G.M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Ontario, Canada, 1966.

[9] V. Pascucci and R. J. Frank. *Hierarchial and Geometrical Methods in Scentific Visualization*, chapter Hierarchical Indexing for Out-of-Core Access to Multi-Resolution Data, pages 225–241. Mathematics and Visualization. Springer-Verlag, Berlin, 2002. UCRL-JC-140581.

[10] Valerio Pascucci. Global static indexing for real-time exploration of very large regular grids. ACM Press, 2001.

[11] Eric J. Stollnitz, Anthony D. Derose, and David H. Salesin. *Wavelets for Computer Graphics (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, January 1996.

[12] Gilbert Strang and T. Nguyen. *Wavelets and filter banks*. Wellesley-Cambridge Press, 1997.