where $\{g_{j,1}^{(n/2)}\}$, $\{g_{j,2}^{(n/2)}\}$ are the Discrete Fourier Transforms of $n/2$ interlaced points. Each of the $n/2$ transforms requires only $n^2/4$ multiplications (as they stand) so to construct the set $\{g_j^{(n)}\}$ requires only $n^2/2+n$ multiplications, a significant saving over n^2, if n is large.

However, if n is a multiple of 4, we need not stop here since the Discrete Fourier Transform of the two data sets $\{f_{2k+\ell}\}, \ell = 0,1$ can be split up again for another significant computational saving. This further splitting requires that we calculate transforms of the four sets of points $\{f_{4k+\ell}\}, \ell = 0,1,2,3$.

If $n = 2^p$, the best thing to do is to use this simplification recursively, and find the two point transform of all pairs of points which are spaced 2^{p-1} points apart. From these we construct the transform of all sets of four points spaced 2^{p-2} points apart, and continue to build up from there until the desired transform is found.

Suppose the number of multiplications required for this procedure to produce an n point transform is $K(n)$. From our recursive formula we realize that

$$K(2n) = 2K(n) + 2n$$

multiplications are required for $2n$ points. The solution of this equation is

$$K(n) = n \log_2 n = p \cdot 2^p,$$

where $K(2) = 2$. Certainly, this is a significant saving over the naive calculation which required n^2 multiplications.

It is not a difficult matter to write a computer program that calculates discrete Fourier transforms in this fast way. However, most computers currently have available Fast Fourier Transform software that perform these computations in this efficient way, or if such a program is not readily available, the code given in the book by Press, et al., will work quite well.

Other versions of the discrete Fourier transform may be appropriate in certain situations. For example, with real data, one may prefer a real valued transform (the discrete Fourier transform is complex valued). The discrete Sine transform is defined for the data $\{f_k\}_{k=1}^{n-1}$ by

$$g_j = \sum_{k=1}^{n-1} f_k \sin \frac{\pi kj}{n}$$

with inverse transform

$$f_k = \frac{2}{n} \sum_{k=1}^{n-1} g_j \sin \frac{\pi kj}{n}.$$

Of course, the transform and inverse transform are the same operation except for the scale factor $\frac{2}{n}$.

Once this transform is defined, it is not hard to devise a fast Sine transform algorithm which is more efficient than using the direct calculation. Computer programs to implement this fast Sine transform are also readily available.

One further observation needs to be made here. This transform is actually a matrix multiplication by a matrix A whose entries are $a_{kj} = \sin \pi kj/n$. The columns of the matrix are mutually orthogonal and the norm of each column is $\sqrt{n/2}$. The usefulness of this transform will be enhanced if we can find matrices which are diagonalized by this transform. Indeed, just as Fourier Transforms are useful in the study of ordinary and partial differential equations, the discrete Fourier transforms, and especially the fast algorithms, are quite useful in the study of the discretized versions of differential equations, which we will see in chapters 4 and 8.

2.2.5 Walsh Functions and Walsh Transforms

WALSH FUNCTIONS are a complete set of orthonormal piecewise constant functions defined on the interval $x \in [0,1)$, which are quite useful in the processing of digital information.

There are two equivalent definitions of the Walsh functions. The first is inductive and gives a good understanding of how they are generated and why they are orthogonal and complete. The goal is to build a set of orthogonal, piecewise constant functions. We first show that, if we do this correctly, the set of functions will be complete.

Suppose $f(x)$ is a continuous function. For some fixed n, we divide the interval [0,1] into 2^n equal subintervals, and define

$$q_n(x) = f\left(\frac{j-1/2}{2^n}\right) \quad j = 1,2,\ldots,2^n$$

for $j-1 \le 2^n x \le j$, that is, on each subinterval, $q_n(x)$ is the same as f at the midpoint of the interval. It follows that $|f(x) - q_n(x)| < \omega\left(f, \frac{1}{2^n}\right)$ which is smaller than ϵ for n large enough and that

$$\int_0^1 (f(x) - q_n(x))^2 \, dx < \epsilon^2.$$

Since continuous functions are dense in L^2, the piecewise constant functions are also dense in L^2.

To generate an orthonormal set of piecewise constant functions we take $\mathrm{Wal}(0,x) = 1$; for x in [0,1] and

$$\mathrm{Wal}(1,x) = \begin{cases} 1\,, & 0 \le x < 1/2 \\ -1\,, & 1/2 \le x < 1\,. \end{cases}$$

Using these two functions as starters, we proceed inductively, defining

$$\mathrm{Wal}(2n,x) = \begin{cases} \mathrm{Wal}(n,2x) & 0 \le x < 1/2 \\ (-1)^n \mathrm{Wal}(n,2x-1)\,, & 1/2 \le x < 1 \end{cases}$$

and

$$\mathrm{Wal}(2n+1,x) = \begin{cases} \mathrm{Wal}(n,2x) & 0 \le x < 1/2 \\ (-1)^{n+1} \mathrm{Wal}(n,2x-1)\,, & 1/2 \le x < 1. \end{cases}$$

In words, we take a known Walsh function $\mathrm{Wal}(n,x)$ defined on [0,1), contract it down by a change of scale to the interval $[0,1/2)$ and then reuse the scaled down function on the interval $[1/2,1)$ with multiplier ± 1 to generate two new Walsh functions. In this way $\mathrm{Wal}(1,x)$ is used to generate $\mathrm{Wal}(2,x)$, and $\mathrm{Wal}(3,x)$, and these two in turn generate $\mathrm{Wal}(k,x)$, $k = 4,5,6,7$ in the next step. Notice that the 2^n Walsh functions $\mathrm{Wal}(k,x)$, $k = 0,1,2,\ldots,2^n-1$ are constant on intervals of length no less than $1/2^n$.

A second definition of Walsh functions is a bit more direct and more useful for computer programming, although less insightful. For any $x \in [0,1)$ let $x = \sum_{j=1}^{\infty} a_j 2^{-j}$, $a_j = 0$ or 1 be the BINARY EXPANSION of x. Define $a_k(x)$ to be the k^{th} coefficient of this expansion. For example, to determine a_1 we divide the interval $[0,1)$ into two subintervals $[0,1/2)$ and $[1/2,1)$. If $x < 1/2$, we take $a_1 = 0$ whereas if $x \ge 1/2$, we take $a_1 = 1$. Proceeding inductively, we suppose $a_1, a_2, \ldots, a_k$ are known. Let

$$x_k = \sum_{j=1}^{k} a_j 2^{-j}.$$

If $x \ge x_k + \frac{1}{2^{k+1}}$ we take $a_{k+1} = 1$ whereas if $x < x_k + \frac{1}{2^{k+1}}$ we take $a_{k+1} = 0$. In this way the a_k are uniquely specified. For example, in this representation, $x = 3/8$ has $a_1 = 0$, $a_2 = 1$, $a_3 = 1$, and $a_k = 0$ for $k \ge 4$.

With x given in its binary expansion we define

$$\mathrm{Wal}(2^k - 1, x) = (-1)^{a_k(x)}.$$

The Walsh functions $\mathrm{Wal}(2^k-1,x)$ flip alternately between ± 1 at the points $x_j = j/2^k$, $j = 1,2,\ldots,2^k-1$. To determine the remaining Walsh functions, we define the "exclusive or" binary operation $\oplus$ (also called add without carry) using the table

$\oplus$	0	1
0	0	1
1	1	0.

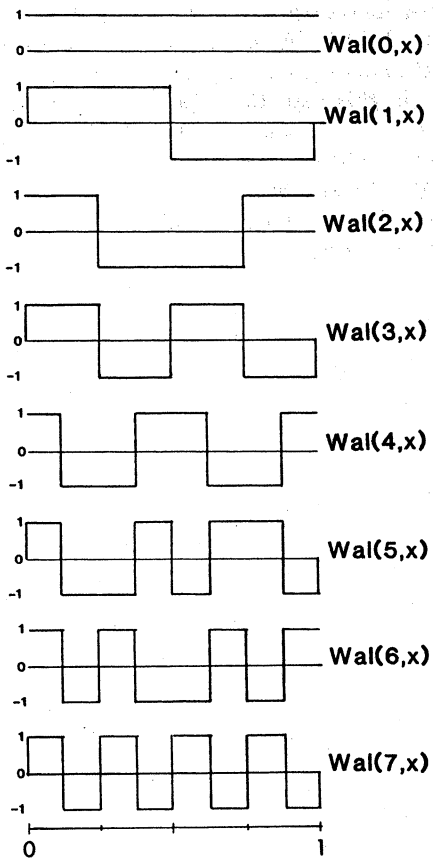

FIGURE 2.2 The first eight Walsh functions.

For integers i,j, we define $i \oplus j$ as the operation $\oplus$ applied to the binary representations of i and j. Since $\oplus$ is its own additive inverse, $i \oplus i = 0$ and $(i \oplus j) \oplus j = i$. The definition of the remaining Walsh functions follows from

$$\mathrm{Wal}(i \oplus j, x) = \mathrm{Wal}(i,x) \cdot \mathrm{Wal}(j,x).$$

To make some sense of this definition, we note that we can uniquely decompose any integer n into a $\oplus$ sum of the integers $2^k - 1$, as

$$n = \sum_{j=0}^{N} b_j 2^j = \sum_{j=1}^{N} \oplus\, (b_{j-1} \oplus b_j)\,(2^j - 1) \oplus b_N\,(2^{N+1} - 1)\,.$$

For example, $4 = 3 \oplus 7$, and $10 = 1 \oplus 3 \oplus 7 \oplus 15$. The way to view this decomposition operationally is via simple shifts. For example, to decompose

the integer 13, write its binary representation 1101 (since 8+4+1=13) and $\oplus$ add this to its shift 0110 (shift the entries 1101 one position to the right to get 0110) as $1101 \oplus 0110 = 1011$. The resulting zeros and ones are coefficients of integers $2^j - 1$. In this case $13 = 15 \oplus 3 \oplus 1$. Another way to view this decomposition uses the fact that $\oplus$ is its own additive inverse. As a result, $13 \oplus 15 = 1101 \oplus 1111 = 0010$ so $13 \oplus 15 \oplus 3 = 0010 \oplus 11 = 1$ and $13 \oplus 15 \oplus 3 \oplus 1 = 0$ so that $13 = 15 \oplus 3 \oplus 1$ as found before.

The proof of this identity follows easily by induction on N. Suppose the binary representation of a number is $b_{N+1}b_Nb_{N-1}\cdots b_1b_0$ and suppose that,

$$b_Nb_{N-1}\cdots b_1b_0 = \sum_{j=1}^{N} \oplus (b_{j-1} \oplus b_j)(2^j - 1) \oplus b_N(2^{N+1} - 1).$$

It is certainly true that $b_0 = b_0(2^1 - 1)$. We use that the binary representation of $2^j - 1$ is the string of j ones, $11\ldots11$ and then

$$\begin{aligned} b_{N+1}b_Nb_{N-1}\cdots b_1b_0 &= b_Nb_{N-1}\cdots b_1b_0 \oplus b_{N+1}00\cdots00 \\ &= b_Nb_{N-1}\cdots b_1b_0 \oplus b_{N+1}((2^{N+2} - 1) \oplus (2^{N+1} - 1)) \\ &= \sum_{j=1}^{N+1} \oplus (b_{j-1} \oplus b_j)(2^j - 1) \oplus b_{N+1}(2^{N+2} - 1). \end{aligned}$$

Now that n is decomposed as a linear combination of integers $2^j - 1$ and $\mathrm{Wal}(2^n - 1, x) = (-1)^{a_j(x)}$, it follows that

$$\mathrm{Wal}(n, x) = (-1)^{\left(\sum_{j=1}^{N} a_j(b_j + b_{j-1}) + a_{N+1}b_N\right)}$$

or

$$\mathrm{Wal}(n, x) = (-1)^{\left(\sum_{j=1}^{N} (a_j + a_{j+1})b_j + a_1b_0\right)}$$

where $x = \sum_{j=1}^{\infty} a_j 2^{-j}$. Notice that replacing $\oplus$ by $+$ in these formulas is not a misprint. Why?

To verify that this definition is equivalent to our first inductive definition, we note that if $n = \sum_{j=0}^{N} b_j 2^j$, then $2n = \sum_{j=0}^{N+1} b'_j 2^{j-1}$ where $b'_j = b_{j-1}$, $b'_0 = 0$ and $2n + 1 = \sum_{j=0}^{N+1} b'_j 2^{j-1}$ where $b'_j = b_{j-1}$ with $b'_0 = 1$. Thus for $m = 2n$ or $2n + 1$

$$\begin{aligned} \mathrm{Wal}(m, x) &= (-1)^{\left(\sum_{j=1}^{N+1} (a_j + a_{j+1})b'_j + a_1b'_0\right)} \\ &= (-1)^{\left(\sum_{j=1}^{N} (a'_j + a'_{j+1})b_j + a'_1b_0 + a_1(b_0 + b'_0)\right)} \\ &= \mathrm{Wal}(n, x')(-1)^{a_1(b_0 + b'_0)} \end{aligned}$$

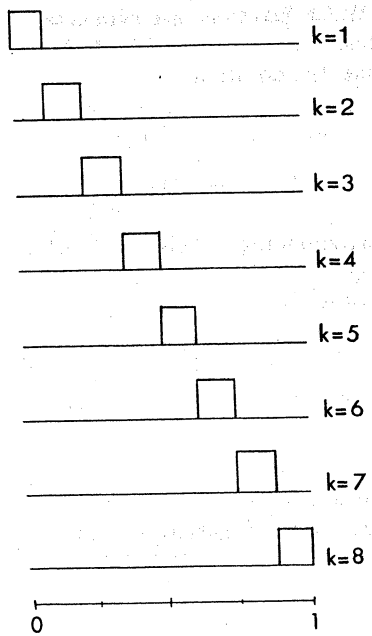

FIGURE 2.3 The "natural basis" of piecewise constant functions.

where $a'_j = a_{j+1}$, $x' = (2x) \bmod 1$. When $0 \le x < 1/2$, $a_1 = 0$ and when $1/2 \le x < 1$, $(-1)^{b_0} = (-1)^n$, so the two definitions agree.

It is clear (by induction) that for $n \ge 1$, $\int_0^1 \mathrm{Wal}(n, x)dx = 0$, whereas $\int_0^1 \mathrm{Wal}(0, x)dx = 1$. To show orthogonality of the set, note that

$$\int_0^1 \mathrm{Wal}(i, x)\mathrm{Wal}(j, x)dx = \int_0^1 \mathrm{Wal}(i \oplus j, x)dx = \delta_{ij},$$

since $i \oplus i = 0$. For $j \le 2^n - 1$, the Walsh functions $\mathrm{Wal}(j, x)$ are constant on intervals of length no less than $1/2^n$. Since there are 2^n such functions and they are orthogonal, hence linearly independent, they span the same set as the "natural" basis functions

$$g_k(x) = \begin{cases} 1 & k - 1 \le 2^n x \le k \\ & \qquad\qquad k = 1, 2, \ldots, 2^n \\ 0 & \text{elsewhere}. \end{cases}$$

It follows that the Walsh functions are complete. The natural basis functions are shown for $n = 3$ in Figure 2.3.

Even numbered Walsh functions are symmetric and odd numbered ones are antisymmetric about the point $x = 1/2$. In light of the analogy with the trigonometric functions, the notation

$$\begin{aligned} \mathrm{cal}(k,x) &= \mathrm{Wal}(2k,x) \\ \mathrm{sal}(k,x) &= \mathrm{Wal}(2k-1,x) \end{aligned}$$

has been adopted. Corresponding to this notation, the identities

$$\begin{aligned} \mathrm{cal}(i,x)\mathrm{cal}(j,x) &= \mathrm{cal}(i \oplus j, x) \\ \mathrm{cal}(i,x)\mathrm{sal}(j,x) &= \mathrm{sal}\left((i \oplus (j-1)) + 1, x\right) \\ \mathrm{sal}(i,x)\mathrm{sal}(j,x) &= \mathrm{cal}\left((i-1) \oplus (j-1), x\right) \end{aligned}$$

can be shown to hold.

Since the Walsh functions are an orthonormal, complete, set of functions in $L^2[0,1]$, we can represent any function in $L^2[0,1]$ as

$$f(x) = \sum_{n=1}^{\infty} a_n \mathrm{Wal}(n,x)$$

where

$$a_n = \int_0^1 f(t)\mathrm{Wal}(n,t)dt$$

which is the WALSH TRANSFORM PAIR for $f(x)$. Note that this integral is easy to evaluate since $\mathrm{Wal}(n,x)$ is ± 1. If the function $f(x)$ is not continuous but consists rather of evenly spaced (digitized) sample points f_i, the DISCRETE WALSH TRANSFORM pair is defined as

$$\begin{aligned} f_i &= \sum_{n=0}^{N-1} \alpha_n \mathrm{Wal}(n,x_i) \\ & \qquad i = 0,1,2,\ldots,N-1 \\ \alpha_i &= \frac{1}{N}\sum_{n=0}^{N-1} f_n \mathrm{Wal}(i,x_n)\ ,\ x_i = \frac{i+1/2}{N} \end{aligned}$$

for $N = 2^p$, some power of 2. One can show that, except for the normalizing factor $\frac{1}{N}$, the forward and inverse Walsh transform are the same operation (see problem 2.2.19).

Numerical routines that perform the discrete Walsh transforms are extremely fast compared even to the Fast Fourier transform. If $N = 2^p$ for some

p, notice that

$$\begin{aligned} \alpha_i &= \frac{1}{N}\sum_{k=0}^{N-1} f_k \mathrm{Wal}(i,x_k) \\ &= \frac{1}{2M}\sum_{k=0}^{M-1} \left(f_{2k}\mathrm{Wal}\left(i,x_{2k}\right) + f_{2k+1}\mathrm{Wal}\left(i,x_{2k+1}\right)\right) \end{aligned}$$

where $M = N/2$. We now use that $\mathrm{Wal}(i,x_k) = \mathrm{Wal}(k,x_i)$ (see problem 2.2.19) to write that

$$\alpha_i = \frac{1}{2M}\sum_{k=0}^{M-1} \left(f_{2k}\mathrm{Wal}(2k,x_i) + f_{2k+1}\mathrm{Wal}(2k+1,x_i)\right)$$

and finally, we use the inductive definition of the Walsh functions and find

$$\begin{aligned} \alpha_i &= \frac{1}{M}\sum_{k=0}^{M-1} \left(\frac{f_{2k}+f_{2k+1}}{2}\right)\mathrm{Wal}(k,\hat{x}_i) \\ & \qquad\qquad 0 \le i \le M-1 \\ \alpha_{i+M} &= \frac{1}{M}\sum_{k=0}^{M-1} (-1)^k \left(\frac{f_{2k}-f_{2k+1}}{2}\right)\mathrm{Wal}(k,\hat{x}_i) \end{aligned}$$

where $\hat{x}_i = \frac{i+1/2}{M}$. The obvious observation to make is that the first M coefficients α_i, $i = 1,\ldots,M-1$ are the Walsh transform of the average of consecutive data points and the coefficients α_i, $i = M, M+1,\ldots,2M-1$ are the Walsh transform of half the difference of consecutive data points.

This formula applied recursively enables us to evaluate the Walsh transform without ever evaluating a Walsh function, but merely by transforming the data by adding or subtracting successive pairs of data points. The following FORTRAN code calculates the Walsh transform of 2^p data points by exploiting this formula to the fullest extent. In the code, the division by two is not done until the very end, when it becomes division by 2^p. This step saves time and gives an operation count of $p2^p$ adds or subtracts, and 2^p divisions. This is indeed faster than the fastest Fast Fourier Transform code.

```
      SUBROUTINE FWT(F,G,N)
      DIMENSION F(1),G(1)
C
C     F...INPUT VECTOR CONTAINING N DATA POINTS
C
```

```
C       G...WORK VECTOR WITH LENGTH AT LEAST N
C
C       N...LENGTH OF INPUT VECTOR, ASSUMED TO BE  POWER OF
C           IF N IS NOT A POWER OF 2, THE SUBROUTINE USES T
C           LARGEST POWER OF TWO LESS THAN N
C
C       EXCEPT FOR THE LOOP DO 60 I=..., THIS CODE CAN BE
C       INTEGERIZED. THAT IS, ALL VARIABLES CAN BE USED AS
C       INTEGERS
C
C       CALCULATE P WHERE N.GE.2**P
        IP=0
        ITP=1
   10   ITP=2*ITP
        IF(ITP.LE.N)THEN
        IP=IP+1
        GO TO 10
        ENDIF
        ITP=ITP/2
        NN=ITP
        WRITE(6,*)NN,IP
C
C       FIND THE WALSH TRANSFORM OF THE VECTOR F
        DO 50 I=1,IP
        JS=NN/ITP
        ITP=ITP/2
        DO 50 J=1,JS
        JST=1+2*(J-1)*ITP
        CALL COMBIN(F(JST),G(JST),ITP)
   50   CONTINUE
        DN=FLOAT(NN)
        DO 60 I=1,NN
        F(I)=F(I)/NN
   60   CONTINUE
        RETURN
        END
C
C       THIS ROUTINE ADDS AND SUBTRACTS CONSECUTIVE PAIRS
C       OF ELEMENTS OF F AND PUTS THEM BACK INTO THE
C       APPROPRIATE POSITION IN F.
        SUBROUTINE COMBIN(F,G,NS)
        DIMENSION F(1),G(1)
        A=-1.
        DO 100 I=1,NS
```



```
        G(I)=F(2*I-1)-F(2*I))
        A=-A
        G(I+NS)=A*(F(2*I-1)-F(2*I))
  100   CONTINUE
        IST=2*NS
        DO 200 I=1,IST
        F(I)=G(I)
  200   CONTINUE
        RETURN
        END
```

2.2.6 Finite Elements

As we know, mutually orthogonal basis functions are useful because they give least squares approximations of a function f using the Fourier coefficients $f = \sum_{n=1}^{\infty} \alpha_k \phi_k(x)$, $\alpha_k = \langle f, \phi_k \rangle$. FINITE ELEMENTS are functions which are not mutually orthogonal but nonetheless give useful approximations to functions in a Hilbert space.

Suppose the interval [0,1] is subdivided into N subintervals of length $h = 1/N$ and $x_j = j/N$, $j = 0, 1, 2, \ldots, N$ are the endpoints of the intervals.

Definition

A FINITE ELEMENT SPACE $S^h(k, r)$ is the set of all functions $\phi(x)$ defined on [0,1] satisfying

1. $\phi(x)$ is a polynomial of degree less than or equal to k, on the interval $[x_j, x_{j+1}]$,
2. $\phi(x)$ is r times continuously differentiable on [0,1].

If $r = 0$, the functions ϕ are continuous but not differentiable and if $\phi(x)$ is allowed to be discontinuous we set $r = -1$.

The finite element space $S^h(k, r)$ is a finite dimensional vector space of dimension $N(k - r) + r + 1$ and as a result has a finite set of basis functions. The simplest example is the set of piecewise constant functions

$$\phi_j(x) = \begin{cases} 1 & x_{j-1} \le x \le x_j \\ 0 & \text{otherwise} \end{cases}$$

denoted $S^h(0, -1)$. This is the set of functions from which we generated the Walsh functions in the previous section.

The next reasonable choice of basis functions are the piecewise linear