Seminararbeit

Discrete Alternatives to the Sines

by

Andreas Schärtl

Matrikel-Nr.: —

Supervision:
Prof. Oliver Keszöcze

September 29, 2019

This document was produced with the typesetting system LaTeX2e.

This seminar paper started as a work based on Donald Knuth's *The Art of Computer Programming*, in particular on a somewhat recent addition which deals with algorithms and theorems related to "generating all n-tuples" [Knu01].

While originally, this paper was supposed to to start with a gradual introduction to n-tuples and binary encodings, ultimately that proved to be too broad of a topic. Instead, this work now focuses on one tidbit Knuth mentions in his discussion of n-tuples, namely the theory of Rademacher and Walsh functions, which are similar to the well-known sines and can be used in some of the same applications.

Contents

1 Introduction

In science and technology, often periodic processes can be formalized in terms of sine waves

$$y(x) = A \sin(\omega x + \varphi), \tag{1.1}$$

parameterized by amplitude A, frequency ω and phase φ [Pap09, pp. 163] with the following effects on function $y(x)$ as visualized in Figure 1.1.

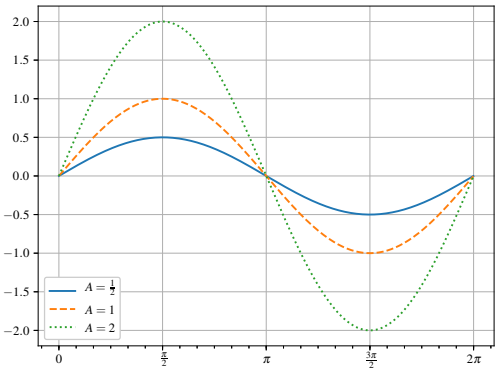
- Amplitude A stretches the wave in the vertical direction.
- Frequency ω on the other hand stretches the wave in the horizontal direction. As such, ω controls the number of oscillations of $y(x)$ in one period.
- Phase φ shifts the sine on the horizontal.

Not only can we control amplitude, frequency and shift of sine waves, with the technique known as *Fourier transformation*, we can also represent any periodic and continuous function as a sum of sine waves, the so called *Fourier series* [Sha95]. Fourier transformation powers many systems taken for granted today, such as digital telecommunication and media compression [Pap09, pp. 163].

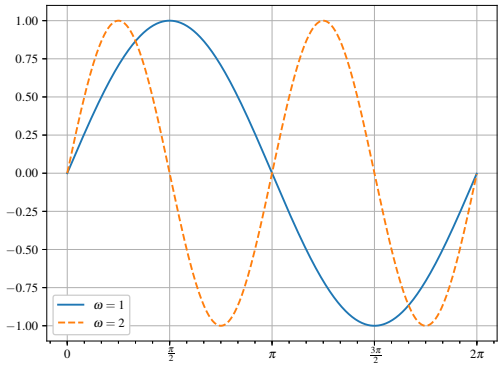
1.1 Discrete Alternatives

The theory of sines and the Fourier transform are often part of an engineering or math education. Less known is that there exist alternatives to the sines and the related Fourier transforms. In particular, there exist *discrete* alternatives to the sines; instead of formulating problems in terms of continuous waves, discrete pulses can be a viable alternative. A potentially better fit for applications on digital computers.

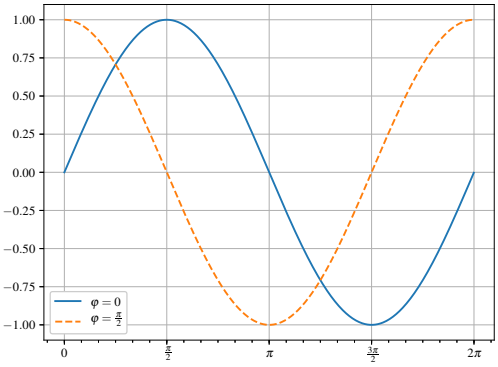
This seminar paper looks at two discrete alternative to the sines, starting with the simple *Rademacher functions*. While they do provide similar flexibility as the sines, Rademacher functions are missing some not immediately apparent features intrinsic to the trigonometric functions. This weakness is not shared by the *Walsh functions*. Despite their digital nature, sums of Walsh functions can be used to approximate other functions in the same way we approximate signals using the Fourier series.



(a) $A\sin(x)$ with $A = \frac{1}{2}, 2, 3$.



(b) $\sin(\omega x)$ with $\omega = 1, 2$.



(c) $\sin(x+\varphi)$ with $\varphi = 0, \frac{\pi}{2}$.

Figure 1.1: Illustrating the three parameters of a sine wave, namely amplitude A, frequency ω and phase φ.

2 A Canonical Approach

Tasked with finding a discrete alternatives for the sines, we might come up with something that looks like square waves S_T,

$$S_T(x) = \begin{cases} 1 & 0 \leq x < T/2 \\ -1 & T/2 \leq x < T \end{cases}, \tag{2.1}$$

defined on some interval $[0, T)$. This definition fulfills our requirement for a digital function when values $\{-1, 1\}$ are encoded as binary zero and one. To be as close as possible to the familiar trigonometric functions, we would set $T = 2\pi$ for a square wave that looks very much like the familiar sine (Figure 2.1.)

It should not come as a surprise that such functions have been discovered and studied before. Indeed, the set of *Rademacher functions* originally described by German mathematician Hans Rademacher in 1922 [Rad22, pp. 130] is in line with our intuition about discrete sines.

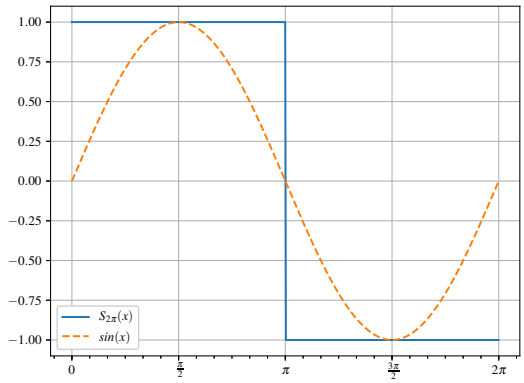


Figure 2.1: $S_{2\pi}$ compared to the regular sine.

Definition. The *Rademacher functions* $r_k : \mathbb{R} \to \{-1, 1\}$ are defined defined as

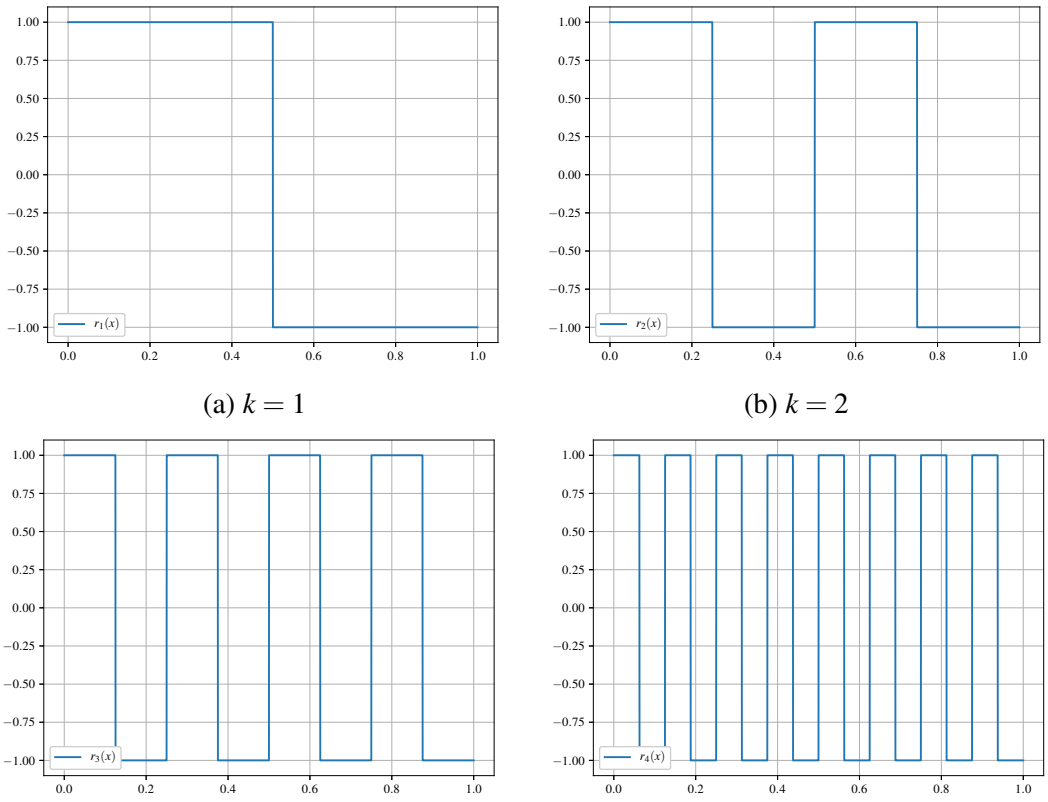
$$r_k(x) = (-1)^{\lfloor 2^k x \rfloor} \tag{2.2}$$

with $k \in \mathbb{N}$.

The Rademacher functions for $k = 1$ (identical to S_1) as well as $k = 2, 3, 4$ are plotted in Figure 2.2. Above definition of r_k is based on Knuth's formulation, rather than the one original proposed by Rademacher himself as Knuth's definition is closer to what a digital computer might end up evaluating [Knu01, p. 8]. Another equivalent definition of the Rademacher functions is

$$r_k(x) = \operatorname{sgn}\left(\sin(2^k \pi x)\right) \tag{2.3}$$

when $\operatorname{sgn}(0) = 1$ [Weib]. Equation 2.3 is useful because it illustrates the relationship between Rademacher functions and their trigonometric cousins. However, it can only be illustrative; discrete alternatives for the trigonometric functions must not involve evaluating the sine.



(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

Figure 2.2: Plotting Rademacher functions on $[0, 1)$.

2.1 Parameters

Similar to how sines can be controlled with parameters A, ω and φ, it is easy to modify r_k with parameters to control amplitude a and phase p. Less obvious is that intrinsic to the Rademacher functions is an equivalent to frequency, namely parameter k.

Let us investigate all these parameters in order. First of all, multiplying r_k with *amplitude* a changes the image of r_k such that

$$r_{k,a}(x) = a\, r_k(x) \ \in \{-a, a\}. \tag{2.4}$$

This is not really the same as amplitude A of the sines which changes the image to

$$A\ \sin(x) \in [-A, A], \tag{2.5}$$

because the sine is a surjective mapping. But for a discrete alternative, Equation 2.4 is as far as we should go, maintaining the digital nature of the Rademacher functions. As for the next parameter, introducing a *shift* p is equality trivial; merely adding p to argument x moves r_k on the horizontal, viz.

$$r_{k,p}(x) = r_k(x+p). \tag{2.6}$$

Intrinsic to each Rademacher function r_k is parameter k which controls the number of oscillations r_k makes in the unit interval $[0,1)$; function r_k includes exactly 2^{k-1} oscillations on $[0,1)$.

Proof. With each increment of k, exponent $\xi(x) := \lfloor 2^k x \rfloor$ grows to twice its previous size. In addition, flooring $2^k x$ acts like a filter that discards everything right of the decimal point. As a result, $\xi(x)$ is a stairway function with its step width controlled by k. With each increment of k, we get twice as many stairs of equal width. When evaluating $r_k(x)$, each time a new step is reached, the sign of $r_k(x)$ switches. This results in 2^{k-1} oscillations on $[0,1)$. □

With parameters a, p and k, the Rademacher functions could be an alternative to the sines. For example, parameterized Rademacher functions are used to modulate messages in digital communication [Moh77]. Their advantage compared to the sine is that they are very easy to compute, as we will see now.

2.2 Evaluating Rademacher Functions

In practical applications of the Rademacher functions, it is necessary to evaluate $r_k(x)$ in a reasonable amount of time. There are multiple ways to compute r_k efficiently on digital hardware, this section introduces two approaches.

Knuth notes that $r_k(x)$ can be computed quite elegantly if argument x is represented as a fixed point number, that is as a binary string with a decimal point, viz.

$$x = (\ldots\, c_2\, c_1\, c_0\, .\, c_{-1}\, c_{-2} \ldots). \tag{2.7}$$

In that case we get

$$r_k(x) = (-1)^{c_{-k}} = \begin{cases} 1 & c_{-k} = 0 \\ -1 & c_{-k} = 1 \end{cases} \tag{2.8}$$

[Knu01, p. 8]

Proof. Multiplying x with 2^k is equivalent to a shift to the left by k bits, even in this decimal representation. We get

$$\rho(x) := 2^k x = (\dots c_{-k} \,.\, c_{-(k+1)} \dots).$$

Taking the floor from ρ means cutting all digits right of the decimal point. As such, exponent $\lfloor \rho \rfloor$ is an even integer in case $c_{-k} = 0$ and odd when $c_{-k} = 1$, resulting in Equation 2.8. □

Fixed width numbers are used when more sophisticated floating point hardware is unavailable, for example on digital signal processors or embedded systems powered by microcontrollers [Kne17, p. 177]. Knuth's unorthodox approach to evaluating r_k might be a surprisingly good fit.

On a system that represents fractional values as some kind of floating point number, like the one described by IEEE 754 [Kah96] [Kne17, pp. 101], evaluating r_k is also fast. Listing 2.1 illustrates this in C-like pseudo code, where function `rademacher(k, x)` computes $r_k(x)$. On modern machines, all instructions required to execute `rademacher` should run in constant time, which should be obvious for incrementing `x`'s exponent and branching over `p`. Converting floating point `x` to integer `p`, effectively flooring `x`, can be considered a constant time operation only if hardware support is available, for example provided by the `fistp` instruction on the ubiquitous x84_64 architecture [Int19, Vol. 2A 3-355].

Regardless whether we are using low-powered embedded hardware with fixed point numbers or more sophisticated computers with floating point units, evaluating r_k is a simple operation that should not be a bottleneck for any application.

2.3 Orthogonality and Completeness

As previously mentioned, the process known as *Fourier transform* allows us to approximate functions on some interval to arbitrary accuracy. This is not possible with the Rademacher functions. Details on such transformation are described in the following section; for now we only wish to investigate why it is not feasible to use the Rademacher functions for this task.

A deep-dive into the underlying theory is out of scope for this work, however we will define two important concepts and try to gain an intuition on what they mean. What we are looking for is a set of functions ϕ_k *orthogonal* and *complete* on some interval (a,b) as sums of such functions ϕ_k can be used to approximate periodic functions to arbitrary accuracy. Let us begin with the idea behind orthogonality.

```
float rademacher(int k, float x)
{
    // Compute x = (2^k) * x. Because 'x' is represented as a floating
    // point number, it is sufficient to increment its exponent by 'k'.
    x.exponent += k;

    // We are only interested in what is left of the decimal point.
    int p = (int) x;

    // Equivalent to computing (-1)^p.
    return (p % 2 == 0) ? 1 : -1;
}
```

Listing 2.1: Evaluating r_k when x is represented as a floating point.

2.3.1 Orthogonality

Just like orthonormal vectors make a base for a vector space, orthogonal sets of functions make a base for function spaces.

Definition. A set of functions ϕ_k is called *orthogonal* in interval (a,b) if

$$\int_a^b \lambda\ \phi_n(x)\ \phi_m(x)\ dx = \begin{cases} \lambda & n = m \\ 0 & n \neq m \end{cases} \tag{2.9}$$

(Lebesgue integral) and orthonormal when $\lambda = 1$ [Bea84, p. 4].

The Rademacher functions make up a set of orthonormal functions on interval $(0,1)$. Indeed, this is why they were interesting to Rademacher in the first place [Rad22]. Because the proof is quick and intuitive, it is included here.

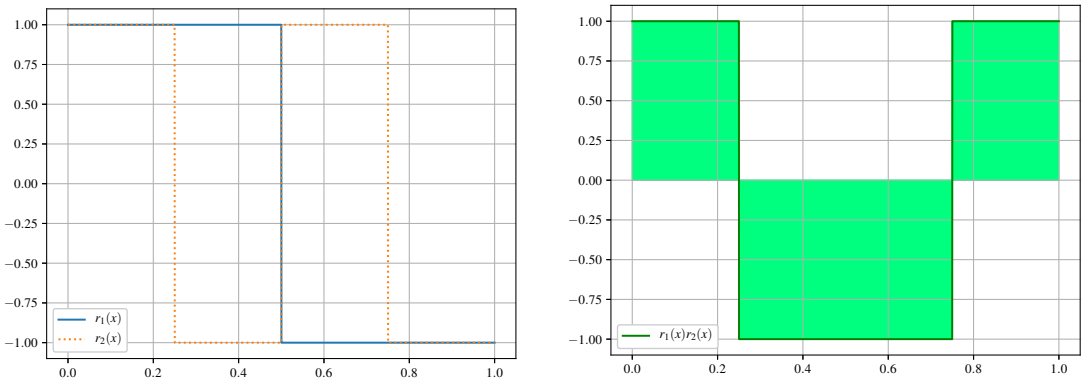
Proof. If $n = m$, we get

$$\int_0^1 r_n(x)\ r_m(x)\ dx = \int_0^1 r_n^2(x)\ dx = \int_0^1 1\ dx = 1. \tag{2.10}$$

If on the other hand $n \neq m$, the product of two Rademacher functions r_n and r_m, $n < m$, is negative just as often as it is positive. We need to realize two things.

- r_k has equally many points x at which $r_k(x) = 1$ as it has points at which $r_k(x) = -1$.
- r_{k+1} has twice as many oscillations as r_k. Each oscillation of r_n is overlaid by some 2^p oscillations from r_m.

The result is that $r_n(x)r_m(x)$ is negative just as often as it is positive, in consequence the integral is zero. The case for r_1 and r_2 is illustrated in Figure 2.3. □



(a) Rademacher functions r_1 and r_2.

(b) Product $r_1(x)r_2(x)$. The integral over this function is 0.

Figure 2.3: The integral over the product of two different Rademacher function is always zero. As an example, here we look at r_1 and r_2.

2.3.2 Completeness

The second property we are after is *completeness*. If we wish to approximate functions f with a series of functions ϕ_k to arbitrary accuracy, we should be able to sum up ϕ_k for $k = 0, 1, 2, \ldots$ in such a way that the error between f and series converges to zero.

Definition. A set of functions ϕ_k orthogonal in (a, b) is called *complete* in that interval if for any piecewise continuous function f, coefficients c_n can be picked such that they minimize the mean-square error E_N,

$$E_N = \int_a^b \left(f(x) - \sum_{n=1}^{N} c_n \, \phi_n(x) \right)^2 dx, \tag{2.11}$$

(Lebesgue integral) that is $E_N \to 0$ as $N \to \infty$ [Weia] [Bea84, p. 4] [BS15, pp. 474].

While functions r_k are orthogonal on the unit, they are not complete [Bea84, p. 8]. Here, the trigonometric functions are more more capable. The set of sines and cosines is orthogonal and complete on $(-\pi, \pi)$, making the Fourier transform possible [Weia]. However, other complete sets of orthogonal functions exist, far beyond the traditional sines. One such set, the one we will look at next, is the set of *Walsh functions*.

Summary

The Rademacher functions match our intuition about a discrete sine. Functions r_k have the same sign as their trigonometric counterparts and are easy to parameterize. Evaluating r_k is fast, regardless whether the argument is represented as a fixed point or floating point number. The downside is that the Rademacher functions are not complete and as such sophisticated applications such as the Fourier transform are not possible with r_k.

3 Walsh Functions

After investigating the rather simple Rademacher functions, we will now look at another set of discrete waves, namely the family of *Walsh functions* originally described by mathematician Joseph Walsh in 1923 [Wal23].

Definition. The Walsh functions $w_k : \mathbb{R} \to \{-1, 1\}$ are defined as

$$w_0(x) = 1 \quad \text{and} \quad w_k(x) = (-1)^{\lfloor 2x \rfloor \lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x). \tag{3.1}$$

for $k \in \mathbb{N}$.

To gain a feel for the Walsh functions, Figure 3.1 plots w_k for $k = 1$ to $k = 6$ together with similar sine waves. These plots graph w_k on the unit and indeed many authors define w_k on $[0, 1)$ rather than $\mathbb{R}$. This is not unlike how some authors define the sine on $(-\pi, \pi)$ or similar. However, setting the domain of w_k to $\mathbb{R}$ introduces no fundamental problems or limitations and above definition reflects this.

3.1 Parameters and Properties

Similar to how we parameterized the Rademacher functions, we can also parameterize the Walsh functions. Amplitude a stretches the function on the vertical while shift p moves it on the horizontal, viz.

$$w_{k,a,p}(x) = a\, w_k(x + p). \tag{3.2}$$

An interesting property of the Walsh functions emerges related to shifts p. w_k is cyclic on the unit interval, that is

$$w_k(x) = w_k(x + 1) = w_k(x - 1). \tag{3.3}$$

Proof. Plugging in argument $x + 1$, we get

$$w_k(x + 1) = (-1)^{\lfloor 2x+2 \rfloor \lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x + 2). \tag{3.4}$$

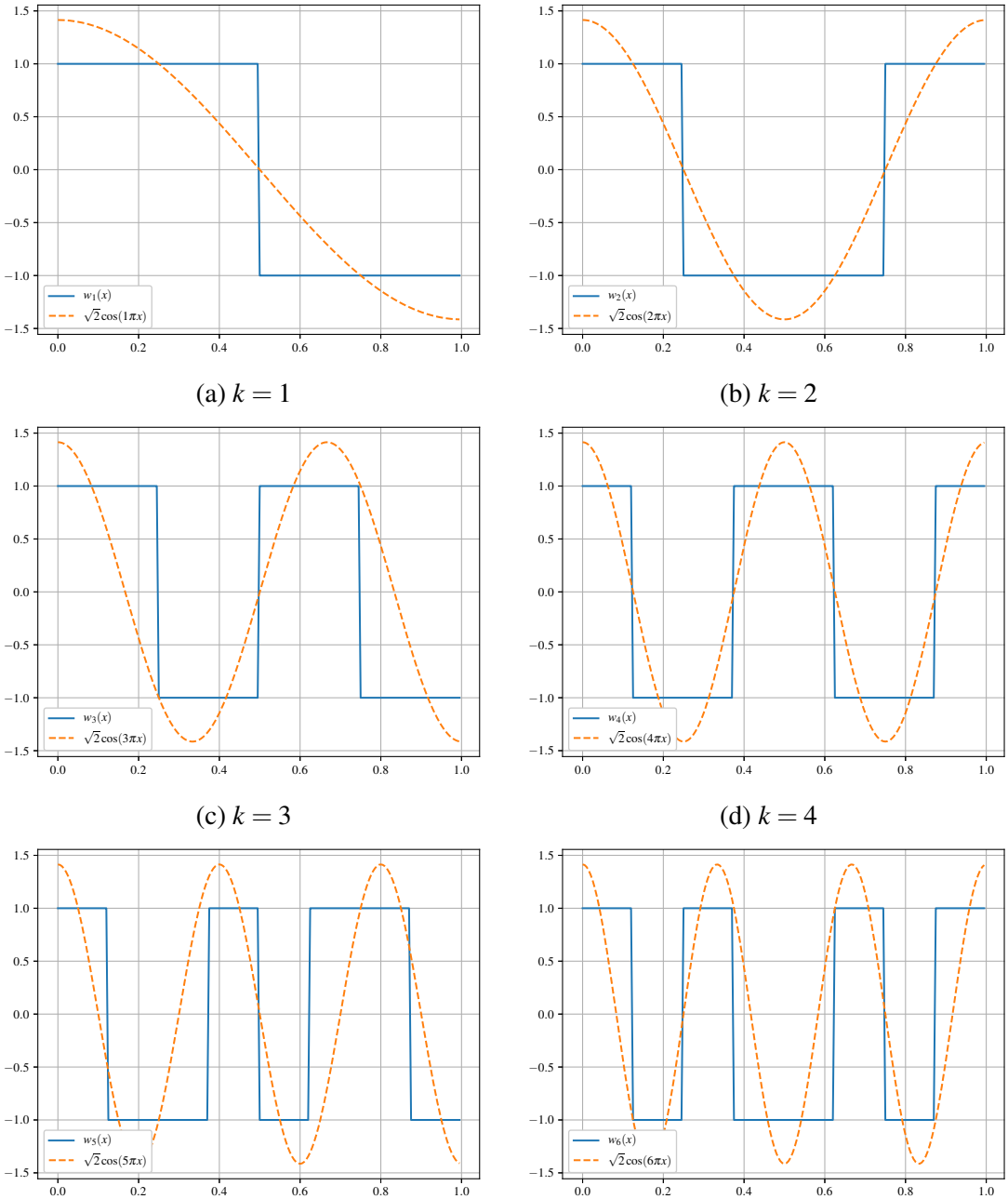
We can drop $+2$ both in exponent and recursive argument as adding 2 to any number does not change its parity.

$$w_k(x + 1) = (-1)^{\lfloor 2x \rfloor \lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x). \tag{3.5}$$

With Equation 3.5 we arrive at the original formulation of the Walsh functions, proving $w_k(x) = w_k(x + 1)$. Finally, with $y := x - 1$, we get

$$w_k(y) = w_k(y + 1) \quad \Rightarrow \quad w_k(x - 1) = w_k(x). \tag{3.6}$$

□



(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

(e) $k = 5$ (f) $k = 6$

Figure 3.1: Comparing the Walsh Functions $w_k(x)$ for $1 \leq k \leq 6$ to a similar sinus wave $\sqrt{2}\cos(k\pi x)$.

Defining amplitude and phase for the Walsh function is trivial and should be so for most functions. What is more involved is understanding the equivalent to frequency ω of the sines. Walsh functions w_k fix the number of sign changes, that is w_k makes exactly k sign changes on $[0,1)$. The plots in figure 3.1 illustrate this property quite well. They also illustrate that unlike simple square waves, the distance between sign changes is not guaranteed to be equal with each step. Because the proof of this property is rather long, it was moved to Section A.1 of the appendix.

Probably inspired by similarities to the frequency of sines, parameter k is referred to as *sequency* k by some authors [Knu01, p. 7]. Sequency k, together with amplitude a and shift p, allow us to parameterize the Walsh functions much like we can parameterize the sines.

3.2 Evaluating Walsh Functions

The recursive nature of w_k might lead to concerns whether Walsh functions can be used in practical applications where $w_k(x)$ needs to be evaluated quickly. This intuition is unfounded; in a naive implementation, each recursive call halves parameter k, resulting in runtime only logarithmic in k.

But Knuth notes that there also is another way. $w_k(x)$ can be evaluated as a product of simpler Rademacher functions, viz.

$$w_k(x) = \prod_{i \geq 0} r_{i+1}(x)^{b_i \oplus b_{i+1}}, \tag{3.7}$$

where $k = (b_{n-1} \ldots b_1\, b_0)$ is the binary representation of k [Knu01, p. 8]. Again, the proof of this equality is quite technical, as such it was moved into the appendix in Section A.2 as well. Trying out some examples, we get

$$w_1(x) = r_1(x) \qquad w_2(x) = r_1(x) r_2(x) \qquad w_3(x) = r_2(x) \tag{3.8}$$

and so on. Note in particular that the product in Equation 3.7 is always finite as k has only a finite number of bits and as such for large i we get product terms

$$r_{i+1}(x)^{0 \oplus 0} = (-1)^0 = 1. \tag{3.9}$$

If we use fixed point arithmetic as discussed in Section 2.2, computing $r_k(x)$ as products of Rademacher functions is particularly attractive. Say we have

$$r_k(x) = r_1(x)\, r_2(x) \cdots r_n(x), \tag{3.10}$$

for some n, we then only need to look at bits

$$c_{-1},\, c_{-2},\, \ldots\, c_{-n} \tag{3.11}$$

of argument x to compute the individual Rademacher functions.

Computing Walsh functions as products of Rademacher functions therefore has a runtime only logarithmic in the number of bits in k which for sufficiently large k is better than the naive recursive algorithm. Equation 3.7 is also useful in that it provides us with ways to show some properties of w_k. One such property is the identity

$$w_k(x)w_{k'}(x) = w_{k\oplus k'}(x), \tag{3.12}$$

a handy theorem similar to the sines' more complicated multiplication rule.

Proof. Say we write arguments k and k' as binary strings

$$k = (\ldots b_2\, b_1\, b_0) \quad \text{and} \quad k' = (\ldots b'_2\, b'_1\, b'_0), \tag{3.13}$$

and $k \oplus k'$ as

$$k \oplus k' = \big(\ldots (b_2 \oplus b'_2)\, (b_1 \oplus b'_1)\, (b_0 \oplus b'_0)\big) = (\ldots c_2\, c_1\, c_0). \tag{3.14}$$

We then get

$$\begin{aligned}
w_k(x)w_{k'}(x) &= \left(\prod_{i\geq 0} r_{i+1}(x)^{b_i \oplus b_{i+1}}\right)\left(\prod_{i\geq 0} r_{i+1}(x)^{b'_i \oplus b'_{i+1}}\right) && (3.15)\\
&= \prod_{i\geq 0} r_{i+1}(x)^{(b_i \oplus b_{i+1}) + (b'_i \oplus b'_{i+1})} && (3.16)\\
&= \prod_{i\geq 0} r_{i+1}(x)^{(b_i \oplus b_{i+1}) \oplus (b'_i \oplus b'_{i+1})} && (3.17)\\
&= \prod_{i\geq 0} r_{i+1}(x)^{(b_i \oplus b'_i) \oplus (b_{i+1} \oplus b'_{i+1})} && (3.18)\\
&= w_{k\oplus k'}(x) && (3.19)
\end{aligned}$$

Equality 3.17 is legal because $r_k^2(x) = 1$ holds for all k and

$$(-1)^{a\oplus b} = (-1)^{a+b} \quad \forall\, a, b \in \{0,1\}. \tag{3.20}$$

□

This proof illustrates how an equality like Equation 3.7 can be useful both in practical applications (when actually evaluating the function), as well as a way of discovering and proving properties.

3.3 Continuous Transform

When describing Walsh functions, most authors begin with an introduction to orthogonality and completeness and then discuss how to construct a series of Walsh functions to approximate other functions. Knuth takes a different route in his work, immediately jumping to how Walsh functions can be used to compute transforms of other functions. We deviated from Knuth here and took the more traditional approach.

As already hinted at multiple times, the sequence of Walsh functions w_k are a complete and orthogonal set of functions on the unit [GES12, p. 4-5, pp. 56]. They can therefore be used to approximate functions on an interval in terms of a *Walsh series*, an equally capable alternative to the traditional Fourier series.

To compare Walsh and Fourier series with each other, we should start with a recap of the Fourier series. A Fourier series $\hat{f}_N$ approximates a function f with period T as a sum of N sines and cosines,

$$f(x) \approx \hat{f}_N(x) = \frac{a_0}{2} + \sum_{k=1}^{N} a_k \cos(2\pi kx) + b_k \sin(2\pi kx), \tag{3.21}$$

for some coefficients a_k and b_k chosen such that the series minimizes the mean-square error (Equation 2.11) compared to the original function f. Computing a_k and b_k is what is referred to as Fourier transform and can be achieved by evaluating the fixed forms

$$a_k = \frac{2}{T}\int_0^T f(x)\cos(2\pi kx)\,dx \quad \text{and} \quad b_k = \frac{2}{T}\int_0^T f(x)\sin(2\pi kx)\,dx \tag{3.22}$$

which do require us to evaluate an integral [Bea84, p. 7, 17] [BS15, p. 474]. As an example, in Figure 3.2 we approximate the polynomial

$$f(x) = (x-5)\,(x-3)\,x \tag{3.23}$$

on $(0,6)$ using a Fourier series with N summands. As expected, with increasing N, we get increasingly better approximations. The same is true when using the Walsh functions instead of sine and cosine. We can approximate functions f with period T in terms of a *Walsh series* $\hat{f}_N$ with N summands,

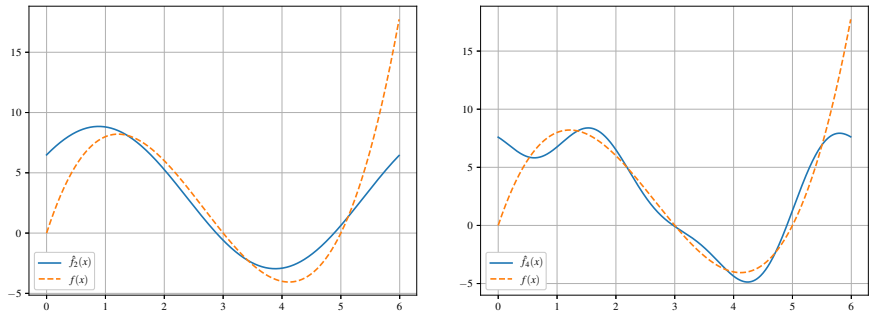
$$f(x) \approx \hat{f}_N(x) = \sum_{k=0}^{N} c_k\, w_k(x), \tag{3.24}$$

and weights

$$c_k = \frac{1}{T}\int_0^T f(x)\, w_k(x)\,dx \tag{3.25}$$

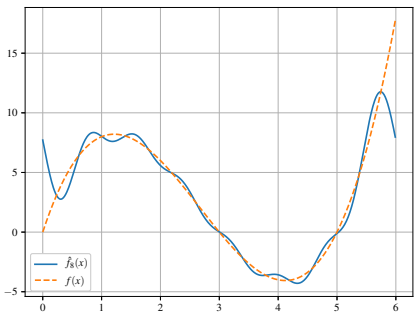
that minimize the mean-square error compared to f [Bea84, pp. 50]. To illustrate the Walsh series, the same polynomial as before is approximated using a Walsh series in Figure 3.3. Approximating digital signals, such as a simple square wave, is also possible and is illustrated in Figures 3.4 and 3.5.

The obvious question that arises is when one should use a Fourier series and when to prefer a Walsh series. Judging from the limited examples provided, we might argue that the Fourier series converges more quickly, though the Walsh functions are easier to evaluate, especially on primitive hardware. Unfortunately, a definite answer to this question could not be found in the literature. For practical applications, it remains a per-case decision which transform to use.

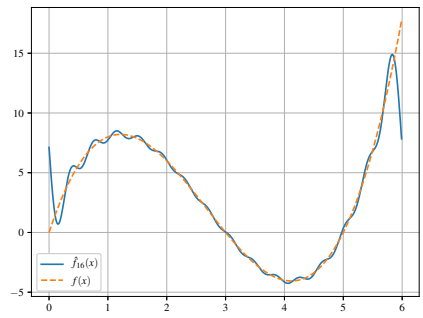


(a) $N = 2$

(b) $N = 4$

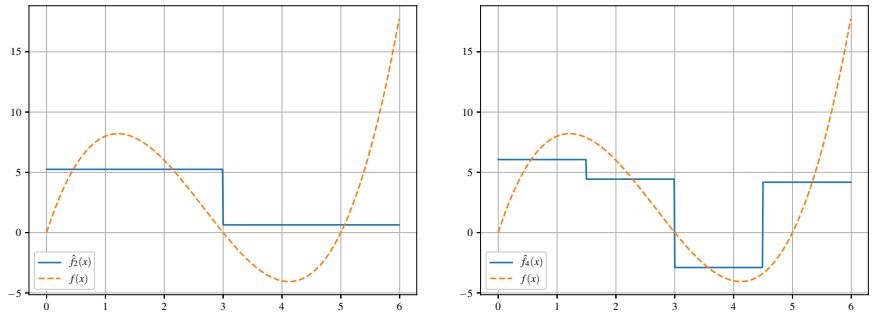


(c) $N = 8$



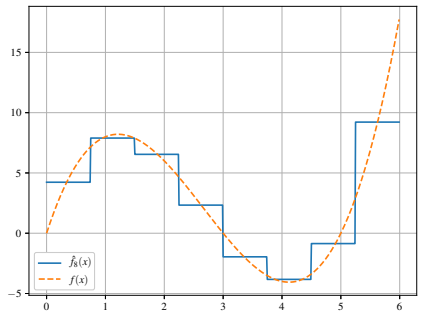
(d) $N = 16$

Figure 3.2: Approximating polynomial $f(x) = (x-5)\,(x-3)\,x$ using a Fourier series $\hat{f}_N$ with N summands.

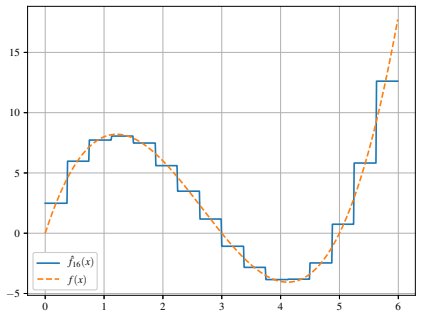


(a) $N = 2$

(b) $N = 4$

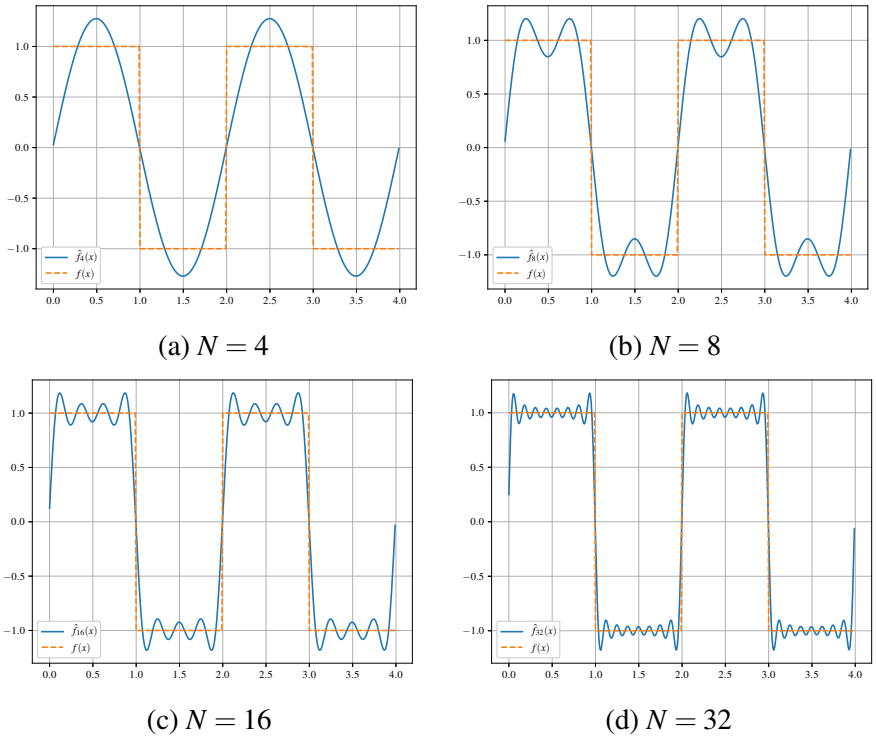


(c) $N = 8$



(d) $N = 16$

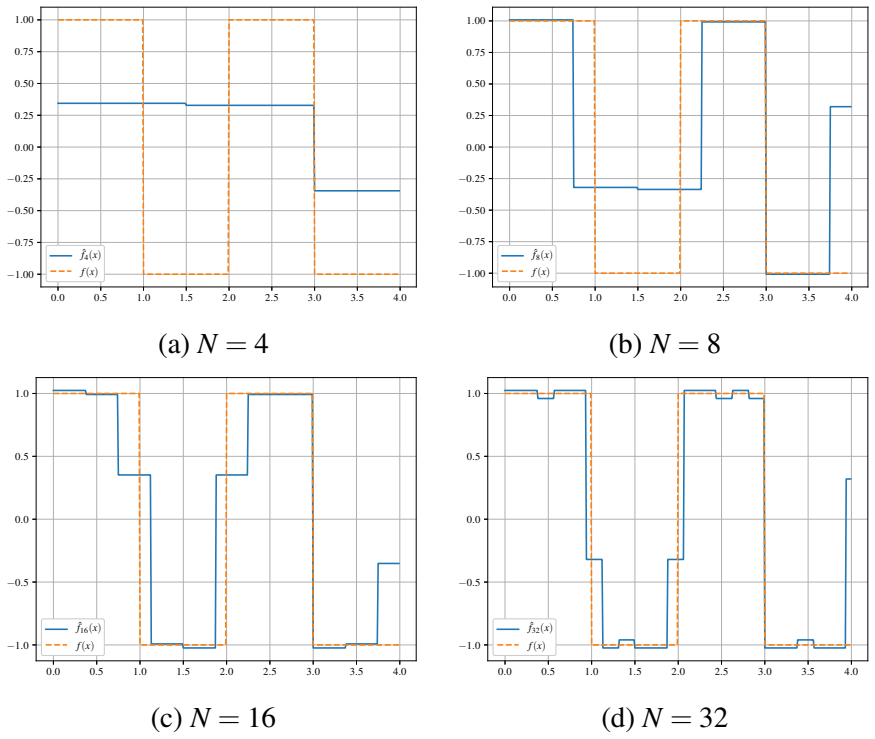
Figure 3.3: Approximating polynomial $f(x) = (x-5)\,(x-3)\,x$ using Walsh series $\hat{f}_N$ with N summands.



(a) $N = 4$ (b) $N = 8$

(c) $N = 16$ (d) $N = 32$

Figure 3.4: Approximating quare wave $f(x) = (-1)^{\lfloor x \rfloor}$ using a Fourier series $\hat{f}_N$ with N summands.



(a) $N = 4$ (b) $N = 8$

(c) $N = 16$ (d) $N = 32$

Figure 3.5: Approximating quare wave $f(x) = (-1)^{\lfloor x \rfloor}$ using Walsh series $\hat{f}_N$ with N summands.

3.4 Discrete Walsh Transform

Often in practical applications, we do not work with abstract functions f, but rather with *samples*, that is measurements taken at fixed intervals. Looking at samples rather than continuous functions, we can use the *discrete Walsh transform*, similar to the well known discrete Fourier transform. This final section introduces a canonical application of the discrete Walsh transform, compression of audio signals.

To get started, the the *discrete Walsh transform* X_n of a single sample x_i out of a total of N samples is given by

$$X_n = \frac{1}{N} \sum_{i=0}^{N-1} x_i \, w_n\left(\frac{i}{N}\right). \tag{3.26}$$

The inverse operation is

$$x_i = \sum_{n=0}^{N-1} X_n \, w_n\left(\frac{i}{N}\right). \tag{3.27}$$

If we store all N transformed points X_n, we can reproduce all N samples x_i to perfect accuracy [Bea84, pp. 49]. An equivalent definition is given by Knuth who notes that the discrete Walsh transform of 2^n samples x_i written in a vector

$$\mathbf{x} = (x_0, \ldots, x_{n-1})^\mathsf{T} \tag{3.28}$$

can be obtained by multiplying $\mathbf{W_n x}$ where $\mathbf{W_n}$ is a $2^n \times 2^n$-matrix with entries

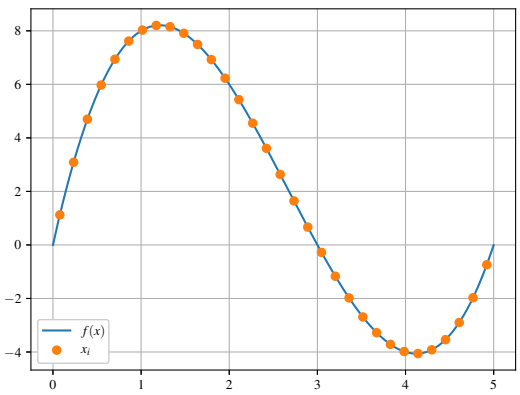
$$w_i\left(\frac{j}{2^n}\right) \tag{3.29}$$

in row i and column j with $i, j \geq 0$ [Knu01, p. 8]. Formulating the Walsh transform as a matrix operation can be quite helpful as highly optimized matrix libraries will perform this operation efficiently.

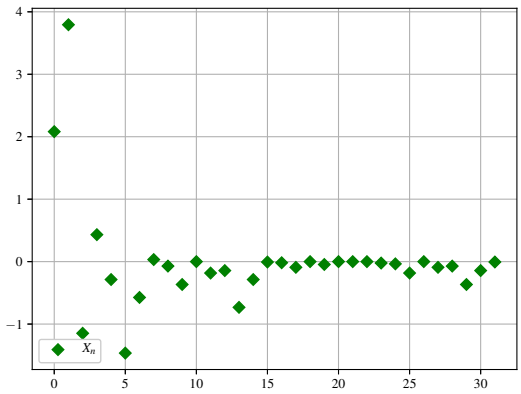
3.4.1 Application in Compression

Now how can we use the discrete Walsh transform to compress audio signals? For one last time, consider the polynomial $f(x) = (x-5)\,(x-3)\,x$, which we will think of as a sound wave. We first sample f at $N = 32$ equally spaced points x_i, illustrated in Figure 3.6a. Applying the discrete Walsh transform, we gain N transformed points X_n. Plotting X_n in Figure 3.6b, we are looking at the *sequency domain* of f. Each point X_n represents the scale of the associated Walsh function w_n. In our example, we see peaks for small n. For larger n, X_n tends to be close to zero.

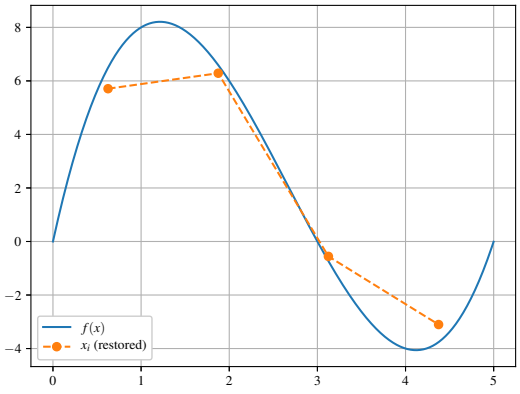
We can exploit this to our advantage and construct a simple compression algorithm for the sound signal. First, apply the discrete Walsh transform to gain N points X_n. Then, only store some $M \leq N$ points. We can approximate the original signal f with just a small amount of points X_n. Figures 3.6c to 3.6f illustrate this approach to signal compression for various M.



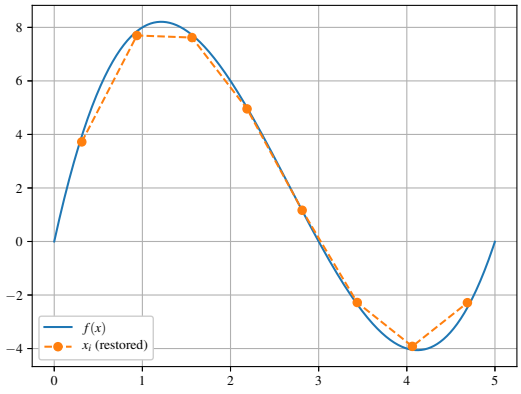
(a) Sample points x_i.



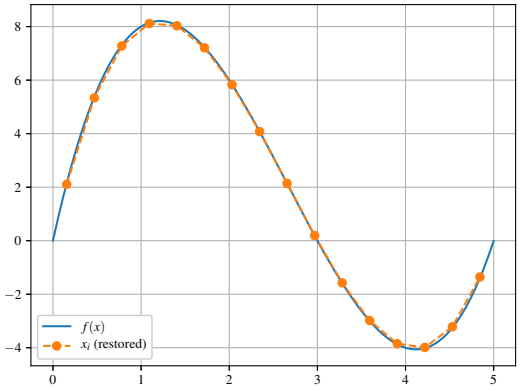
(b) Transformed points X_n in the *sequency domain.*



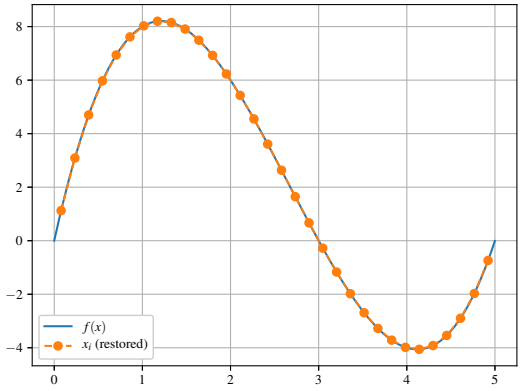
(c) $M = 4$



(d) $M = 8$



(e) $M = 16$



(f) $M = 32$

Figure 3.6: Approximating polynomial $f(x) = (x-5)(x-3)x$ with a discrete Walsh transform. We take N samples and then restructure f with only the first M transformations X_n.

Summary

Because the Rademacher functions are limited, we introduced the Walsh functions. They can be parameterized like the sines and are reasonably fast to compute. What really makes them useful is that the set of functions w_k is orthogonal and complete in the unit. As such, we are able to approximate periodic functions with sums of Walsh functions. Compared to the Fourier transform, approximating smooth waves can take more Walsh summands than Fourier summands, but Walsh functions should be easier to evaluate than the sine. The discrete Walsh transform transforms a vector of samples and has applications in data compression.

4 Conclusion

We started out trying to find a discrete alternative of the sine. Our first intuition lead to the set of Rademacher functions, simple square waves which look similar to a regular sine and easy to parametrize.

We also looked for alternatives to the traditional Fourier series. For a series of a set of functions to approximate periodic functions to arbitrary accuracy, this set of functions needs to be orthogonal and complete. The Rademacher functions are only orthogonal, but not complete and as such could not be used. The Walsh functions however are complete and orthogonal on the unit. Despite their awkward definition, we quickly convinced ourselves of its sine-like properties and capabilities. In particular, we looked at the Walsh series, an alternative to the Fourier series.

A Proofs

This appendix contains two technical proofs omitted in the previous sections. They are based on work by Knuth [Knu01, p. 7-8, 28, 40-41].

A.1 Sign Changes of the Walsh Function

Theorem. Walsh functions w_k,

$$w_0(x) = 1 \quad \text{and} \quad w_k(x) = (-1)^{\lfloor 2x \rfloor \lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x), \tag{A.1}$$

make k sign changes on $[0,1)$.

Proof. Proof by induction on k, we show this property once for even k and once for k that are odd. Starting with even k, that is $k = 2\ell$, we get

$$w_{2\ell}(x) = w_\ell(2x) \quad \text{for} \quad 0 \le x < \frac{1}{2}$$

because in this case

$$w_{2\ell}(x) = \underbrace{(-1)^{0\lceil \ell \rceil}}_{=1} w_\ell(2x).$$

On $[0, \frac{1}{2})$, $w_{2\ell}$ already accumulates $\ell = \frac{k}{2}$ sign changes as argument $2x$ of the recursive call compresses w_ℓ onto half the space. Next, we have to look at the second half, viz.

$$w_{2\ell}(x) = (-1)^{1 \cdot \frac{2\ell}{2}} w_\ell(2x) = (-1)^\ell w_\ell(2x) \quad \text{for} \quad \frac{1}{2} \le x < 1 \tag{A.2}$$

Again, a copy of w_ℓ is compressed into half the space. If ℓ is even, $w_{2\ell}(\frac{1}{2}) = +1$, the left side is simply copied to the right and the right curve starts at $+1$ like normal. If ℓ is odd, $w_{2\ell}(\frac{1}{2}) = -1$ and continues from there because the curve on the left is mirrored to right by mirroring on the vertical axis. Either way results in $\frac{k}{2}$ sign changes on the right and in consequence k sign changes on the unit when k is even.

Now we consider the case of odd k, that is $k = 2\ell + 1$. To prove the theorem for such k, we will show that $w_{2\ell+1}$ makes ℓ sign changes before $x = \frac{1}{2}$, ℓ sign changes after $x = \frac{1}{2}$ and exactly one sign change at $x = \frac{1}{2}$. Starting with the left side, we get

$$w_{2\ell+1}(x) = (-1)^0 w_\ell(2x) = w_\ell(2x) \quad \text{for} \quad 0 \le x < \frac{1}{2}$$

which is identical to what we got for even k, yielding ℓ sign changes. Continuing the argument, on the right side we get

$$w_{2\ell+1}(x) = (-1)^{1\lceil \ell+\frac{1}{2} \rceil} w_\ell(2x) = (-1)^{\ell+1} w_\ell(2x) \quad \text{for} \quad \frac{1}{2} < x < 1.$$

If ℓ is even, $\ell+1$ is odd and the curve on the right is flipped. If ℓ is odd, $\ell+1$ is even and the curves left and right are identical copies. Either way, we get ℓ sign changes on the right. In total, we have already accumulated 2ℓ sign changes. Finally, we need to show that for odd k, a sign change occurs at $x = \frac{1}{2}$. Plugging in $x = \frac{1}{2}$, we get

$$w_{2\ell+1}\left(\frac{1}{2}\right) = (-1)^{\left\lfloor 2 \cdot \frac{1}{2} \right\rfloor \left\lceil \frac{2\ell+1}{2} \right\rceil} w_{\left\lfloor \frac{2\ell+1}{2} \right\rfloor}(1) = (-1)^{\left\lceil \ell+\frac{1}{2} \right\rceil} w_\ell(1) = (-1)^{\left\lceil \ell+\frac{1}{2} \right\rceil} = (-1)^{\ell+1}$$

If ℓ is even, $\ell+1$ is odd and

$$w_{2\ell+1}\left(\frac{1}{2}\right) = -1$$

which works out because $w_{2\ell+1}$ starts at $+1$ for $x = 0$, makes ℓ sign changes and as such ends up at $+1$ again just before $x = \frac{1}{2}$. On the other hand, if ℓ is odd, $\ell+1$ is even and

$$w_{2\ell+1}\left(\frac{1}{2}\right) = +1$$

which works out as above, except there are an odd number of sign changes on the left, meaning that just before $x = \frac{1}{2}$ the curve is at -1. We see that either way a sign change has to occur at $x = \frac{1}{2}$. With this, we considered the full domain of $w_{2\ell+1}$ and count a total of $2\ell+1$ sign changes. □

A.2 Walsh-Rademacher Equality

Theorem. Walsh function $r_k(x)$ can be expressed as a product of Rademacher functions,

$$w_k(x) = \prod_{i \geq 0} r_{i+1}(x)^{b_i \oplus b_{i+1}}, \tag{A.3}$$

where $k = (b_{n-1} \dots b_1\, b_0)$ is the binary representation of k.

Proof. Proof by induction on k, split into two cases. Case 1 looks at arguments $0 \leq x < \frac{1}{2}$, Case 2 looks at $\frac{1}{2} \leq x < 1$. Beginning with Case 1, $0 \leq x < \frac{1}{2}$, we arrive at

$$w_k(x) = (-1)^0\, w_{\lfloor k/2 \rfloor}(2x) = w_{\lfloor k/2 \rfloor}(2x). \tag{A.4}$$

Applying the hypothesis, we arrive at the desired

$$w_{\lfloor k/2 \rfloor}(2x) = r_1(2x)^{b_1+b_2}\, r_2(2x)^{b_2+b_3} \cdots = \underbrace{r_1(x)^{b_0+b_1}}_{=1} \underbrace{r_2(x)^{b_1+b_2}}_{=r_1(2x)^{b_1+b_2}} \underbrace{r_3(x)^{b_2+b_3}}_{r_2(2x)^{b_2+b_3}} \cdots \tag{A.5}$$

because in the specific case of $0 \leq x < \frac{1}{2}$,

$$r_1(x) = (-1)^{\lfloor 2x \rfloor} = 1 \tag{A.6}$$

and in general

$$r_i(2x) = (-1)^{\lfloor 2^i\, 2x \rfloor} = (-1)^{\lfloor 2^{i+1}\, x \rfloor} = r_{i+1}(x) \tag{A.7}$$

as well as

$$(-1)^{a \oplus b} = (-1)^{a+b} \quad \forall\, a, b \in \{0, 1\}. \tag{A.8}$$

With this we have shown that for arguments x on the left of $\frac{1}{2}$, the theorem holds. Now we have to focus our attention on Case 2, $\frac{1}{2} \leq x < 1$. We get

$$w_k(x) = (-1)^{\lceil k/2 \rceil}\, w_{\lfloor k/2 \rfloor}(2x). \tag{A.9}$$

For any x in Case 2, we can rewrite above first factor like this,

$$(-1)^{\lceil k/2 \rceil} = \left((-1)^{\lfloor 2x \rfloor}\right)^{b_0+b_1} = r_1(x)^{b_0 \oplus b_1}, \tag{A.10}$$

because if we look at the exponents fixed point representation

$$\left\lceil \frac{k}{2} \right\rceil = \lceil \dots b_3\, b_2\, b_1\, .\, b_0 \rceil \tag{A.11}$$

we see that if $b_0 = 0$, b_1 remains unchanged when rounding up and if $b_0 = 1$, b_1 gets flipped. This is equivalent to $b_0 \oplus b_1$. Rewriting the first factor as discussed, we now look at

$$w_k(x) = r_1(x)^{b_0+b_1}\, w_{\lfloor k/2 \rfloor}(2x). \tag{A.12}$$

Applying the hypothesis, we arrive at our goal

$$w_k(x) = r_1(x)^{b_0+b_1} \left(r_1(2x-1)^{b_1+b_2} \, r_2(2x-1)^{b_2+b_3} \cdots \right) \tag{A.13}$$

$$= r_1(x)^{b_0+b_1} \, r_2(x)^{b_1+b_2} \, r_3(x)^{b_2+b_3} \cdots \tag{A.14}$$

as

$$r_i(2x-1) = (-1)^{\lfloor 2^i(2x-1) \rfloor} = (-1)^{\lfloor 2^{i+1}x - 2^i \rfloor} \tag{A.15}$$

where subtracting 2^i has no influence on the parity of the exponent; ergo we can rewrite

$$(-1)^{\lfloor 2^{i+1}x - 2^i \rfloor} = (-1)^{\lfloor 2^{i+1}x \rfloor} = r_{i+1}(x). \tag{A.16}$$

□

Bibliography

[Bea84] Beauchamp. *Applications of Walsh and Related Functions*. Academic Press London, 1984.

[BS15] Ilía Nikolaevich Bronshtein and Konstantin A Semendyayev. *Handbook of Mathematics*. Springer Science+Business Media, 6th edition, 2015.

[GES12] Boris Golubov, Aleksandr Efimov, and Valentin Skvortsov. *Walsh Series and Transforms: Theory and Applications*, volume 64. Springer Science & Business Media, 2012.

[Int19] Intel. *Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4*. 2019.

[Kah96] William Kahan. Ieee standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11, 1996.

[Kne17] Ronald T Kneusel. *Numbers and Computers*, volume 2. Springer, 2017.

[Knu01] Donald E Knuth. The Art of Computer Programming, volume 4, Pre-Fascicle 2a: A Draft of Section 7.2. 1.1: Generating all n-tuples, 2001.

[Moh77] Nirode Mohanty. Spread Spectrum and Time Division Multiple Access Satellite Communications. *IEEE Transactions on Communications*, 25(8):810–815, 1977.

[Pap09] Lothar Papula. *Mathematik für Ingeneure und Naturwissenschaftler Band 2*. Vieweg+Teubner, 2009.

[Rad22] Hans Rademacher. Einige Sätze über Reihen von allgemeinen Orthogonalfunktionen. *Mathematische Annalen*, 87(1):112–138, 1922.

[Sha95] Hagit Shatkay. The Fourier Transform - A Primer. *Brown University*, 1995.

[Wal23] Joseph L Walsh. A closed set of normal orthogonal functions. *American Journal of Mathematics*, 45(1):5–24, 1923.

[Weia] Eric W. Weisstein. "Complete Orthogonal System." From MathWorld–A Wolfram Web Resource. [Online; accessed 05-August-2019].

[Weib] Eric W. Weisstein. "Square Wave." From MathWorld–A Wolfram Web Resource. [Online; accessed 07-August-2019].