

17.  $c_k = 2\langle\phi(t), \phi(2t - k)\rangle$ .  
Hint: Use the Orthogonal Decomposition Theorem.
18. Parseval's formula:  $\sum_{k=-\infty}^{\infty} c_k^2 = 2$   
Hint: Use the fact that  $\langle\phi(t), \phi(t)\rangle = \langle\sum_k c_k \phi(2t - k), \sum_k c_k \phi(2t - k)\rangle$ .
19. For every integer  $j$ , except zero,  $\sum_{k=-\infty}^{\infty} c_k c_{k-2j} = 0$ .  
Hint: Use (3.3) to write a dilation equation for  $\phi(t - j)$ .
- 
- 

### 3.4 THE HAAR SCALING FUNCTION REDISCOVERED

As mentioned in the previous section, when working with a multiresolution analysis we often don't have a simple formula for the scaling function. In these situations, however, we usually do know the refinement coefficients. Next, we will consider how such information enables us to determine some characteristics of the scaling function.

One of the standard techniques to deal with problems like this is a numerical one known as the *cascade algorithm* [10]. This algorithm will provide approximations to the scaling function and is an example of a *fixed-point method*.

A *fixed point* of a function  $f$  is a value  $a$  such that  $f(a) = a$ . A simple example of a fixed point method follows, which shows how to determine the fixed point of the cosine function. Let  $f(t) = \cos(t)$ . To find a solution to the equation  $\cos(t) = t$ , start with a guess,  $t_0$ , of a fixed point. Let  $t_1 = f(t_0) = \cos(t_0)$ , then compute another number  $t_2 = f(t_1) = \cos(t_1)$ , and continue. In this way, construct a sequence

$$\{t_0, t_1 = f(t_0), t_2 = f(t_1), \dots, t_{n+1} = f(t_n), \dots\}$$

that will converge to the fixed point of the cosine function. This process will work for functions  $f$  that satisfy certain conditions.

---

#### Problems

---

20. Use the algorithm to determine, to six decimal places of accuracy, the solution of

$$t = \cos(t).$$

21. Use the algorithm to determine, to five decimal places of accuracy, the solution of

$$t = 1 + e^{-t}.$$

22. Explain what happens when you apply the algorithm to attempt to determine the solution of

$$t = 3.7t(1 - t) + 0.2.$$

How is this function different than the other two?

---

The cascade algorithm is a fixed-point method, except that instead of generating a sequence of numbers, it creates a sequence of functions. When given refinement coefficients, this algorithm creates a sequence of functions  $\{f_i\}$  so that, for every value of  $t$ ,  $f_i(t) \rightarrow \phi(t)$  as  $i \rightarrow \infty$ . Recall that  $\phi$  satisfies the dilation equation (3.3). If we let  $F$  be the function that assigns the expression

$$F(\gamma)(t) = \sum_n c_n \gamma(2t - n)$$

to any function  $\gamma$ , then we can consider  $\phi$  as a fixed point of  $F$ ! This process will be illustrated using the Haar dilation equation (3.4).

Begin with a guess,  $f_0(t)$ , of the graph of  $\phi(t)$  (imagining for a moment that we don't know the Haar scaling function). Every scaling function that we have seen so far has a maximum near  $t = 0$  and tends to get smaller as we move away from that maximum. So, a good first guess for  $f_0(t)$  could be the normalized tent function,

$$f_0(t) = \begin{cases} 1 + t, & \text{if } -1 \leq t < 0 \\ 1 - t, & \text{if } 0 \leq t < 1 \\ 0, & \text{otherwise.} \end{cases}$$

---

### Problems

---

**23.** Graph  $f_0(t)$ .

---

There is an alternate way to define  $f_0(t)$  that illuminates how the cascade algorithm works. Think of creating  $f_0(t)$  as a three-step process reminiscent of the trapezoid rule from calculus. First, divide the  $t$ -axis into subintervals with breaks at each integer. Second, give the function a value at each of the integers. In this case, it is zero at every integer except at 0, where the function is equal to 1. Finally, use linear segments to connect the function values on the integers. For the current  $f_0$  we use  $1 + t$  on the interval from 1 to 0,  $1 - t$  on the interval from 0 to 1, and zero everywhere else. The resulting function is called a *linear spline*.

---

### Problems

---

**24.** Use a CAS to graph  $f_0(t)$  in the following way. First, define a list of coordinates, based on the values of  $f_0$  at the integers. (You do not need all of the integers. Focus on the interval  $[-1, 4]$ .) Then, plot these points, connecting them with lines.

**Maple Hint:** Using the `style=line` option with the `plot` command will draw straight lines between the points.

---

We now use  $f_0(t)$  to create a new and better approximation  $f_1(t)$ . (Daubechies calls this step “cranking the machine”.) From (3.4) and the fact that  $f_1 = F(f_0)$ , it follows that

$$f_1(t) = f_0(2t) + f_0(2t - 1).$$

From here, use a three step process: *update*, *extend*, and *connect* to more fully define  $f_1$ . First, find the value of  $f_1$  on the integers. For example,

$$f_1(1) = f_0(2) + f_0(1) = 0 + 0 = 0.$$

Repeating this for each integer shows that  $f_1$ , like  $f_0$ , is equal to zero on all of the integers except 0, where it is equal to 1. Not terribly impressive yet, but this is just the first step.

Now that we have values for  $f_1$  at the integers, extend the function to the points halfway between the integers. For example,

$$f_1\left(\frac{1}{2}\right) = f_0(1) + f_0(0) = 0 + 1 = 1.$$

Calculations like this one show that  $f_1$  is zero at the odd multiples of  $\frac{1}{2}$  with the single exception that  $f_1\left(\frac{1}{2}\right) = 1$ . Finally, complete the definition of  $f_1$  by using linear segments to connect the function values on the integers and the halves, as was done with  $f_0$ .

---

### Problems

---

25. Graph  $f_1$  using the method of exercise 24.

---

Now that  $f_1$  has been determined, we can then “crank the machine” again to determine  $f_2$ . This time, first update the values on the set

$$\left\{ \dots, -1, -\frac{1}{2}, 0, \frac{1}{2}, 1, \dots \right\},$$

and then extend to the odd multiples of  $\frac{1}{4}$ . Finally, connect the function values on all of the multiples of  $\frac{1}{4}$ .

---

### Problems

---

26. Find  $f_2$  and sketch its graph. Note that each graph that you have been generating looks more and more like the Haar scaling function with which we are familiar.

27. “Cranking the machine” one step at a time is time consuming and tiresome. Create a CAS worksheet with loops that will perform the cascade algorithm and plot the graph of  $f_8$ . How similar are  $f_8$  and the Haar scaling function?

**Maple Hint:** One can use either the symbolic or numeric power of *Maple* to generate  $f_8$ . To force *Maple* to use floating-point arithmetic, rather than symbolic algebra, use a decimal point when defining the refinement coefficients. For example,

```
> c[0]:=1.0;
```

Here is one way to create a loop (in *Maple*) to perform this pointwise algorithm. Begin with the refinement coefficients. Here, we use the Haar coefficients

```
> c[0]:=1.0; > c[1]:=1.0;
```

Label the  $i^{\text{th}}$  approximation of  $\phi$  as  $f[i]$ . We will plot these approximations on the interval  $[-1, 4]$ . Remember, at each step in the algorithm,

$$f[i+1](t) = c[0]*f[i](2*t) + c[1]*f[i](2*t-1).$$

Note that to define an approximation on  $[-1, 4]$ , we need values for the prior approximation from  $-3$  to  $8$ . First define  $f[0]$  at integer points from  $-3$  to  $8$ .

```
> for i from -3 to 8 do f[0](i):=0.: od: f[0](0):=1.:
```

To plot this approximation we construct a list of points, called `points[0]`, and connect them with line segments.

```
> points[0]:=[ [ t, f[0](t) ] $t=-1..4];
> plot(points[0], style=line);
```

The following loop generates successive pointwise approximations to the scaling function  $\phi$ . The first approximation is defined on the halves. The second on the quarters, and so on. All of these approximations are restricted to  $[-1, 4]$ .

```
> for j from 1 to 8 do deltak := 2^(-j):
> for k from 0 to 11/deltak do
> x:=-3+k*deltak:
> if -1<=x and x<=4 then
> f[j](x) := c[0]*f[j-1](2*x) + c[1]*f[j-1](2*x-1):
```

```

> else f[j](x):=0: fi:
> od:
> points[j] := [[t*2^(-j),f[j](t*2^(-j))]]$t=-1*2^j..4*2^j:
> od:

```

Now `points[j]` is the  $j^{\text{th}}$  approximation. The successive approximations can be animated in *Maple*; the `plots` package is needed to do this.

```

> with(plots):
> for j from 0 to 8 do
> myplot[j]:=plot(points[j], t=-1..4, style=line,axes=box):
> od:
> display(seq(myplot[i], i=0..8), insequence=true);

```

28. After how many iterations does it make sense to stop “cranking the machine”? Justify your answer.
- 

Many of the examples of wavelet families from section 3.2 have simple refinement coefficients [38]. The scaling function for the hat wavelet satisfies the dilation equation

$$\phi(t) = \frac{1}{2}\phi(2t) + \phi(2t - 1) + \frac{1}{2}\phi(2t - 2),$$

while the quadratic Battle-Lemarié scaling function satisfies

$$\phi(t) = \frac{1}{4}\phi(2t) + \frac{3}{4}\phi(2t - 1) + \frac{3}{4}\phi(2t - 2) + \frac{1}{4}\phi(2t - 3). \quad (3.5)$$

---

### Problems

---

29. Modify your CAS worksheet from problem 27 to approximate the hat wavelet scaling function.
30. Modify your CAS worksheet to approximate the quadratic Battle-Lemarié wavelet scaling function. (Note: the support of this scaling function is  $[0,3]$ .)
31. The cubic Battle-Lemarié wavelet scaling function satisfies the dilation equation

$$\phi(t) = \frac{1}{8}\phi(2t) + \frac{1}{2}\phi(2t - 1) + \frac{3}{4}\phi(2t - 2) + \frac{1}{2}\phi(2t - 3) + \frac{1}{8}\phi(2t - 4).$$

Modify your CAS worksheet to approximate the cubic Battle-Lemarié wavelet scaling function. (Note: the support of this scaling function is  $[0,4]$ .)

32. Determine which, if any, of the functions in problems 29 through 31 are normalized.
33. Modify your CAS worksheet to begin with the function  $f_0$  defined by

$$f_0(t) = \begin{cases} 2(1+t), & \text{if } -1 \leq t < 0 \\ 2(1-t), & \text{if } 0 \leq t < 1 \\ 0, & \text{otherwise.} \end{cases}$$

(Note that this  $f_0$  is twice the tent function and is not normalized.) How does this change affect your results in problems 29 through 31? What conjecture can you draw about the cascade algorithm and normalization? Can you prove your conjecture?

Both Battle-Lemarié wavelet scaling functions are examples of what are called *bell-shaped splines*, or simply *B-splines*. A spline is a function where several polynomials, defined on different sub-intervals, are joined together to create a continuous function. For example, the quadratic Battle-Lemarié scaling function is created by joining three different quadratic functions together, along with the zero function. A project featuring *B-splines* can be found in chapter 4.

### Problems

34. (a) Explain why the hat scaling function could be called the “Linear Battle-Lemarié scaling function.” What about the Haar scaling function?
- (b) The space  $C^q$  consists of all functions whose  $q^{\text{th}}$  derivative is continuous. For each of the four scaling functions (Haar, hat, quadratic Battle-Lemarié, cubic Battle-Lemarié), determine the value of  $q$  so that it is correct to say that the function is a  $C^q$  function. What is the pattern?
35. (a) Show that the function

$$\phi(t) = \begin{cases} \frac{3+t}{3}, & \text{if } -3 \leq t \leq 0 \\ \frac{3-t}{3}, & \text{if } 0 < t \leq 3 \\ 0, & \text{otherwise} \end{cases}$$

is a solution to the dilation equation

$$\phi(t) = \frac{1}{2}\phi(2t-3) + \phi(2t) + \frac{1}{2}\phi(2t+3).$$

- (b) Modify your CAS worksheet to explore what happens when you try to solve this dilation equation [12]. Explain what happens and the relationship to problem 22 of this chapter.

36. Start with a function different than the tent function, and use the cascade algorithm to try to create the quadratic Battle-Lemarié wavelet scaling function. What is the result? A function,  $f$ , that leads to some neat pictures is

$$f(t) = \begin{cases} \frac{1}{2}, & \text{if } -1 < t < 1 \\ 0, & \text{otherwise.} \end{cases}$$

Another approach to find the scaling function  $\phi$ , introduced by Strang, uses matrices [34]. For example, suppose we start with the dilation equation for the quadratic Battle-Lemarié scaling function (3.5) and *assume* that the support of  $\phi(t)$  is  $0 \leq t \leq 3$  and  $\phi(0) = 0 = \phi(3)$ . Substituting  $t = 1$  and  $t = 2$  into (3.5), gives us the following two equations:

$$\phi(1) = \frac{1}{4}\phi(2) + \frac{3}{4}\phi(1)$$

$$\phi(2) = \frac{3}{4}\phi(2) + \frac{1}{4}\phi(1).$$

These equations can be viewed in matrix form as  $\mathbf{x} = L\mathbf{x}$ , where

$$\mathbf{x} = \begin{bmatrix} \phi(1) \\ \phi(2) \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{bmatrix}.$$

Any solution  $\mathbf{x}$  of this linear equation will be an eigenvector of  $L$  with eigenvalue 1. In this case it turns out that  $\phi(1) = \phi(2)$ . After choosing an arbitrary value for  $\phi(1)$ , and hence  $\phi(2)$ , we can then use the dilation equation to determine the values of  $\phi$  on the halves, quarters, etc., in a fashion similar to the cascade algorithm. Finally, all values are multiplied by an appropriate constant so that the norm of  $\phi$  is 1.

### Problems

37. Apply Strang's approach to the dilation equation for the cubic Battle-Lemarié scaling function to determine the relationship between the function values at 1, 2, and 3.