I. Vector Spaces and Linear Representations

A first look at linear representations

Many problems in machine learning (and statistics, and signal processing, and imaging, ...) can be abstracted as fitting a function to a series of data points. That is, we are given pairs of points $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, where $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y \in \mathbb{R}$, and want to find function $f : \mathbb{R}^D \to \mathbb{R}$ such that

$$f(\boldsymbol{x}_n) \approx y_n, \quad \text{for all } \; n = 1, \ldots, N.$$

We will talk a lot about this problem later in these notes and throughout the course, but for now our main concern is this: how do we represent a function in such a way that a digital computer can manipulate and store it?

In this section of the course, we will develop a general framework for discretizing functions. We will be extending and abstracting the key concepts from linear algebra:

- linear subspaces
- norms
- bases / change of bases
- inner products / orthogonality / projections
- linear operators (matrices in finite dimensions)
- eigenvalues / singular values

Good sources for the material in the rest of Section I include:

- G. Strang, "Linear Algebra and its Applications"
- N. Young, "An Introduction to Hilbert Space"
- Naylor and Sell, "Linear Operator Theory in Engineering and Science"

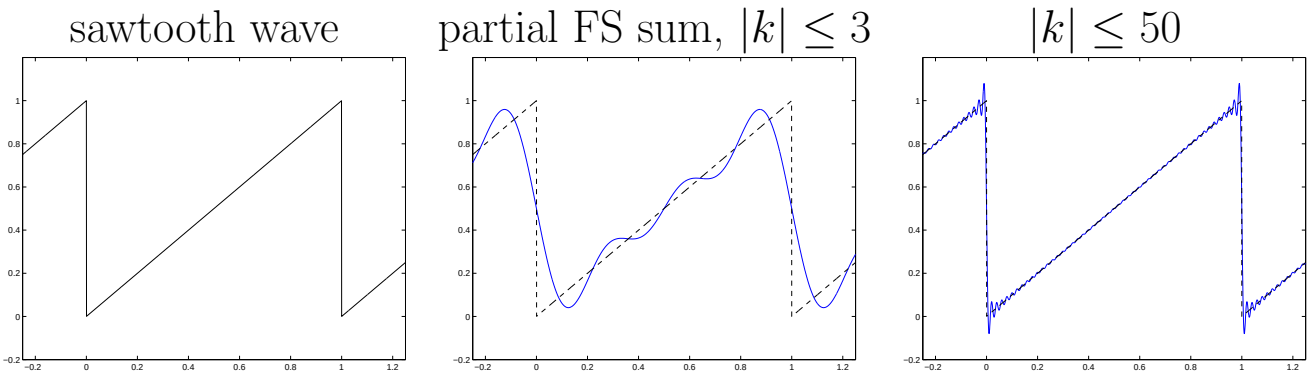
Example: Fourier series

Recall that any periodic signal can be written as a (possibly infinite) superposition of **harmonic sinusoids**. If $f(t)$ has period T, we can write

$$f(t) = \sum_{k=-\infty}^{\infty} \alpha_k \, e^{j2\pi kt/T}, \tag{1}$$

$$\text{where} \quad \alpha_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \, e^{-j2\pi kt/T} \; \mathrm{d}t. \tag{2}$$

(The integral above can be computed over any interval of length T.) The two equations above are another example of a reproducing formula — (2) shows how to systematically take a signal and map it into a discrete list of numbers, while (1) shows how to take that list of numbers and synthesize the signal.



Equivalently, we can think of the Fourier series as building up a function that is time-limited to $[-T/2, T/2]$. That every ("reasonable") function can be represented this way is a deep result in analysis. For those of you with a signal processing background, the fact that

a time-limited function can be written as a sum of harmonic sinusoids is actually mathematically equivalent to the Shannon-Nyquist sampling theorem.

Sin/Cos for real-valued functions

When $f(t)$ is real-valued, we can re-write the Fourier series expansion above as

$$f(t) = \sum_{k=0}^{\infty} a_k \cos(2\pi kt/T) + \sum_{\ell=1}^{\infty} b_\ell \sin(2\pi \ell t/T),$$

where[1] $a_k = 2\,\mathrm{Re}\,\{\alpha_k\}$ and $b_\ell = 2\,\mathrm{Im}\,\{\alpha_k\}$.

This expansion is slightly more satisfying in that we are using real-valued functions to represent other real-valued functions, but slightly less satisfying in that we now have two series to worry about. But of course, we could always rewrite the above as

$$f(t) = \sum_{k=0}^{\infty} \beta_k \psi_k(t),$$

where

$$\psi_k(t) = \begin{cases} \cos(\pi kt/T), & k \text{ even}, \\ \sin(\pi(k+1)t/T), & k \text{ odd}, \end{cases}$$

with an appropriate mapping $(\{a_k\}, \{b_\ell\}) \to \{\beta_k\}$.

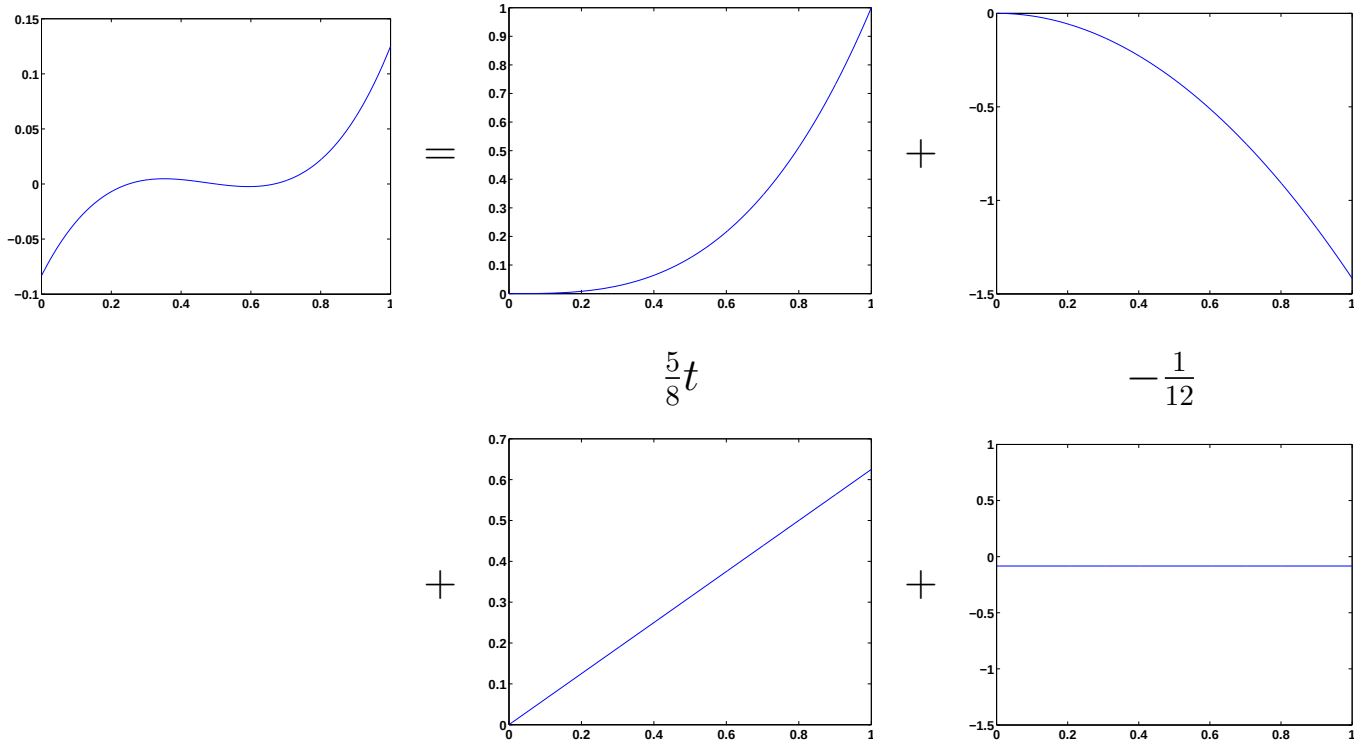
[1] Hopefully it is clear that we are using $\mathrm{Re}\,\{z\}$ and $\mathrm{Im}\,\{z\}$ to denote the real and imaginary parts of a complex number, respectively.

Example: Taylor series

It is almost too obvious that any mth order polynomial can be written as a super position of the $m+1$ functions $1, t, t^2, \ldots, t^m$. (Indeed, this is pretty much the definition of polynomial.)

For example:

$t^3 - \frac{17}{12}t^2 + \frac{5}{8}t - \frac{1}{12}$ $= 1 \cdot t^3 + -\frac{17}{12}t^2 + \frac{5}{8}t - \frac{1}{12}$



More generally, "nice" functions (i.e. analytic functions, see below) can be re-written as "infinite degree" polynomials using a Taylor

series. For instance, on the interval $[-1/2, 1/2]$, we can write

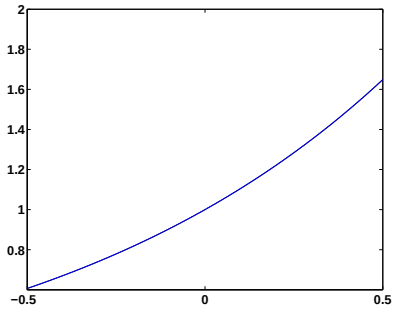
$$e^t = \sum_{k=0}^{\infty} \alpha_k \cdot t^k, \quad \text{where } \alpha_k = \frac{1}{k!},$$

$$\log(1+t) = \sum_{k=0}^{\infty} \alpha_k \cdot t^k \quad \text{where } \alpha_k = \frac{(-1)^{k+1}}{k},$$

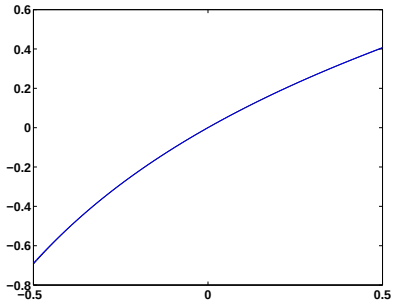
$$\sin(2\pi t) = \sum_{k=0}^{\infty} \alpha_k \cdot t^k \quad \text{where } \alpha_k = \begin{cases} \frac{(-1)^{(k+3)/2}(2\pi)^k}{k!} & k \text{ odd} \\ 0 & k \text{ even} \end{cases}$$

Here are the three examples above with the series truncated to the first six terms:

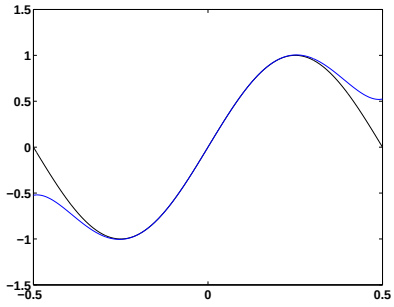
$e^t \approx \sum_{k=1}^{6} \frac{1}{k!}\, t^k$



$\log(1+t) \approx \sum_{k=1}^{6} \frac{(-1)^{k+1}}{k}\, t^k$



$\sin(2\pi t) \approx 2\pi t - \frac{(2\pi)^3}{6} t^3 + \frac{(2\pi)^5}{120} t^5$



The exp and log examples are pretty much spot-on with only six terms, while the sin example is still suffering a little on the edges.

Well-defined "infinite degree" polynomials are called **analytic functions**[2]. For these functions, there is a systematic way of computing the expansion coefficients to represent $\boldsymbol{x}$ on some interval (say $[-1/2, 1/2]$ again),

$$f(t) = \sum_{k=0}^{\infty} \alpha_k \cdot t^k, \quad \text{where} \quad \alpha_k = \frac{f^{(k)}(0)}{k!},$$

where $f^{(k)}$ is the k-th derivative of f.

It is important to realize that Taylor series is not the only way to build up a function as a sum of polynomials, and despite its convenience, it has a few unsatisfying properties (e.g. there are infinitely differentiable functions whose Taylor series converges, but does not equal the original function anywhere). Moreover, it is unclear how to use Taylor series for signals that only have a small number of derivatives.

Example: Lagrange polynomials

Truncating the Taylor series gives us a way to approximate a function by a polynomial (again, this is by no means the best way). But let's consider a slightly different problem: what if I have a set of $M+1$ samples $y_0, \ldots, y_M$ at locations $t_0, \ldots, t_M$ and I want to find a polynomial that passes through these points.

It is a fact that there is a unique polynomial of order M that passes through $M+1$ distinct points (2 points define a line, 3 point define

[2]More technically, a function $f(t)$ is called analytic on an interval $[a, b]$ if for any $t_0 \in [a, b]$, the infinite sum $\sum_{k \geq 0} a_k (t - t_0)^k$ converges to $f(t)$ for some choice of $\{a_k\}$.

a parabola, etc). In fact, there is actually fairly straightforward expression for this polynomial; we take

$$f(t) = \sum_{m=0}^{M} \alpha_m \, p_m(t)$$

where $\alpha_m = y_m$, and

$$p_m(t) = \prod_{\substack{0 \le k \le M \\ k \neq m}} \frac{t - t_k}{t_m - t_k}.$$

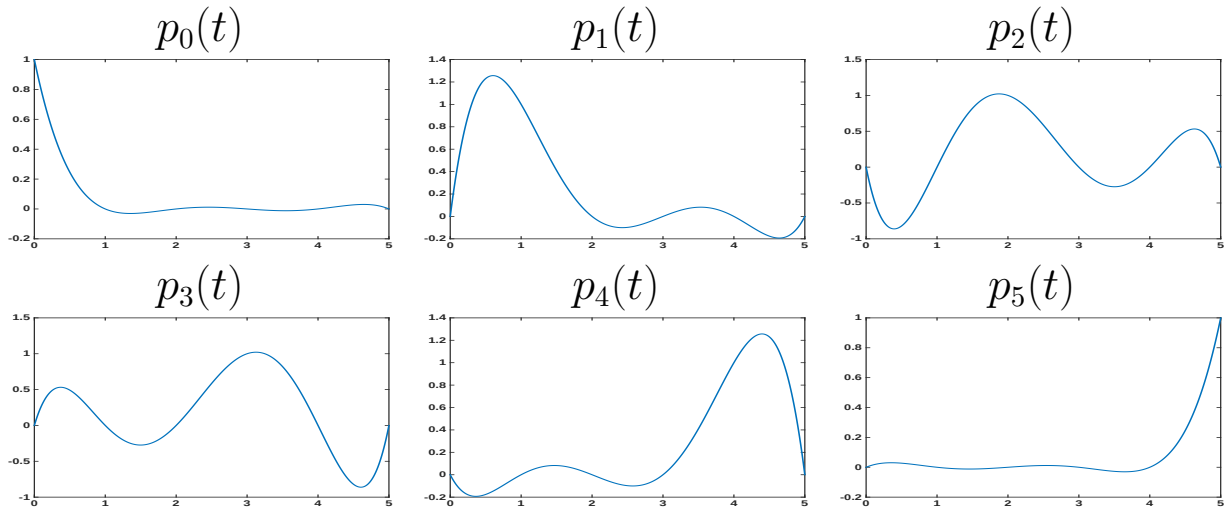
It should be clear from the expression above that each $p_m(t)$ is a different M-th order polynomial, and

$$p_m(t_k) = \begin{cases} 1, & m = k \\ 0, & m \neq k. \end{cases}$$

One consequence of these expressions is that we can **reproduce** an M-th order polynomial from $M+1$ samples — if $f(t)$ is a polynomial of order M and $t_0, \ldots, t_M$ obey $t_i \neq t_j$ for $i \neq j$, then

$$f(t) = \sum_{m=0}^{M} f(t_m) \, p_m(t) \quad \text{for all} \;\; t \in \mathbb{R}.$$

Here are the 6 basis functions $p_k(t)$ for $M = 5$ and t_m on the integers ($t_m = m, \; m = 0, \ldots, M$):



One problem with Lagrange polynomials is that they are extremely unstable outside of the interval $[t_0, t_M]$ — they diverge very quickly to either ∞ or $-\infty$ (as all polynomials must do).

Example: Splines

A more stable way to interpolate between a sequence of discrete points is by using a **polynomial spline**. Given a sequence of locations $t_1, t_2, \ldots, t_K$ and function values at those locations $y_1, y_2, \ldots, y_K$, the ℓth order polynomial spline is the function $f(t)$ which obeys:

$$f(t_k) = y_k, \text{ for } k = 1, \ldots, K,$$

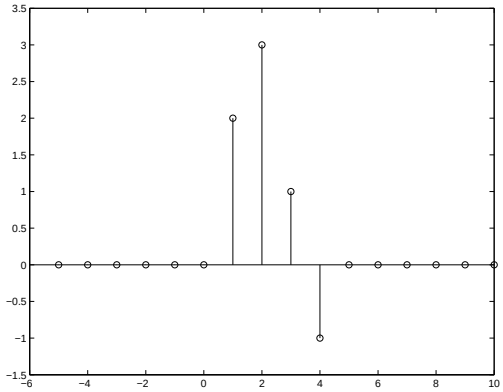
and

$f(t)$ is an ℓth order polynomial between the t_k

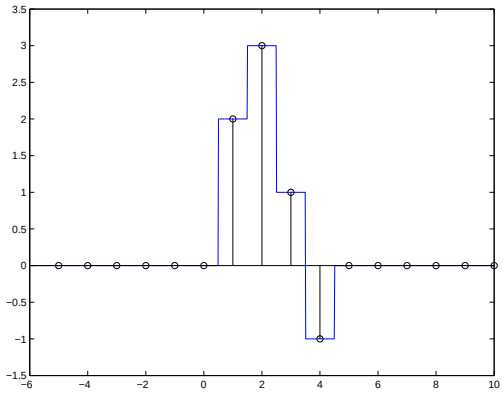
For $\ell \geq 1$, the spline function is continuous and will have $\ell - 1$ derivatives which are continuous at the t_k.

For example, if we have data points at the integers
$t_1 = 1,\ t_2 = 2,\ t_3 = 3,\ t_4 = 4$

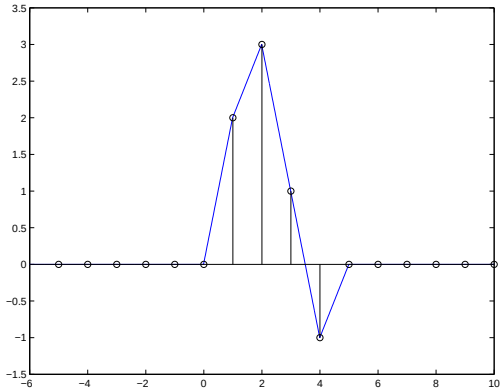
with values
$y_1 = 2,\ y_2 = 3,\ y_3 = 1,\ y_4 = -1$
and values of zeros at the other integers,



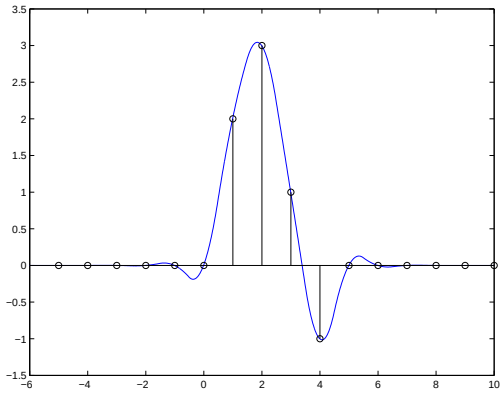
then here is the zero-th order interpolation,



the linear interpolation,



and the quadratic interpolation,



Any ℓth order polynomial spline can be written as a superposition of B-spline functions (the 'B' is for basis!).

The piecewise constant function above can be written as

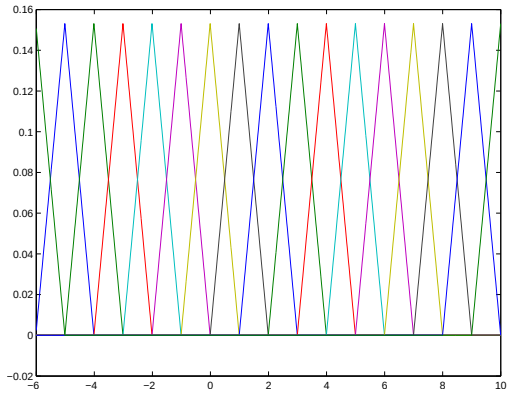
$$f(t) = \sum_{k=1}^{4} \alpha_k \, b_0(t-k), \quad b_0(t) = \begin{cases} 1 & -1/2 \le t < 1/2 \\ 0 & \text{else} \end{cases}$$

for $\alpha_1 = 2,\ \alpha_2 = 3,\ \alpha_3 = 1,\ \alpha_4 = -1$. The piecewise linear function above can be written as

$$f(t) = \sum_{k=1}^{4} \alpha_k \, b_1(t-k), \quad b_1(t) = \begin{cases} t+1 & -1 \le t \le 0 \\ 1-t & 0 \le t \le 1 \\ 0 & \text{else} \end{cases}$$

for $\alpha_1 = 2,\ \alpha_2 = 3,\ \alpha_3 = 1,\ \alpha_4 = -1$. In this case, the building blocks $b_1(t)$ are 'hat' functions:

$b_1(t-k),\ k=-6,\ldots,10$



For spline expansions using an order greater than 1, the expansion coefficients α_k will not be equal to the sample values. However, given a set of M samples values, the α_k that interpolate these samples can be found by solving a system of equations.

Notice that we can generate b_1 by convolving[3] b_0 with itself:

$$b_1(t) = (b_0 * b_0)(t) = \int_0^1 b_0(s) b_0(t-s)\,\mathrm{d}s.$$

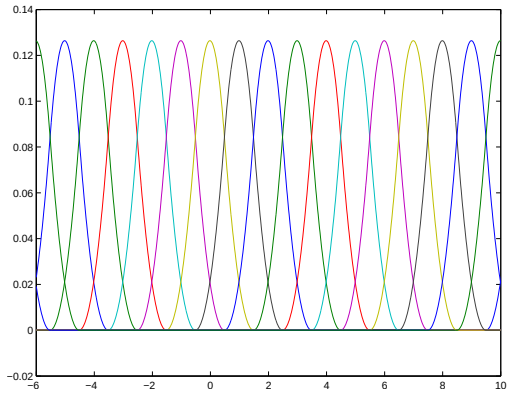
The expansion for the piecewise quadratic spline above is a little more complicated:

$$f(t) = \sum_{k=-\infty}^{\infty} \alpha_k\, b_2(t-k), \quad b_2(t) = \begin{cases} (t+3/2)^2/2 & -3/2 \le t \le -1/2 \\ -t^2 + 3/4 & -1/2 \le t \le 1/2 \\ (t-3/2)^2/2 & 1/2 \le t \le 3/2 \\ 0 & |t| \ge 3/2 \end{cases}$$

where the α_k are all non-zero. The expansion has an infinite number of terms because the basis functions overlap on the integers:

[3] Recall the general definition of convolution of two functions whose domain is the real-line: $(g*h)(t) = \int_{-\infty}^{\infty} g(s)h(t-s)\,\mathrm{d}s$. The limits of the integral are determined by the support of g.

$b_2(t-k),\ k=-6,\ldots,10$



Just as before, we can generate the basis function $b_2(t)$ from the lower order ones:

$$b_2(t) = (b_1 * b_0)(t) = (b_0 * b_0 * b_0)(t).$$

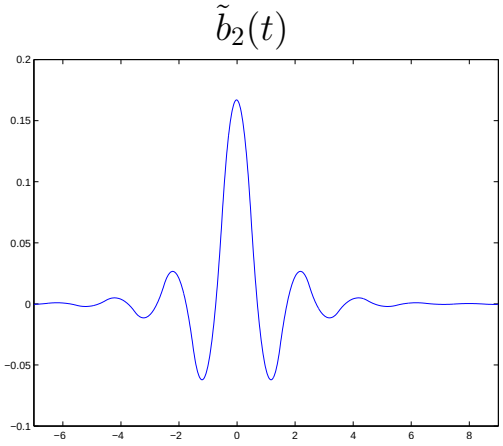
In general, any ℓth order polynomial spline $x(t)$ is uniquely represented by a list of numbers $\{\alpha_k,\ k \in \mathbb{Z}\}$, which correspond to the weights needed to re-synthesize the spline from the building blocks $\{b_\ell(t-k),\ k \in \mathbb{Z}\}$:

$$f(t) = \sum_{k=-\infty}^{\infty} \alpha_k b_\ell(t-k), \quad b_\ell(t) = \underbrace{b_0(t) * \cdots * b_0(t)}_{\ell \text{ times}}$$

If we are given an ℓth order spline $f(t)$, there is a systematic way to compute the corresponding α_k (and hence gives us another reproducing formula). Without getting too much into the details at this point, there is a complementary function $\tilde{b}_\ell(t)$ such that

$$\alpha_k = \int_{-\infty}^{\infty} f(t)\tilde{b}_\ell(t-k)\ \mathrm{d}t.$$

In the quadratic case $\ell = 2$, this function looks like:



We know how to compute these complementary $\tilde{b}_\ell(t)$, but they are not easy to write down as nice expressions.

There are also functions that produce the spline directly from the samples — these are called the **cardinal spline** functions and we will denote them as $\hat{b}_\ell(t)$. This allows us to write

$$f(t) = \sum_{k=1}^{K} y_k \, \hat{b}_\ell(t-k),$$

and if a function $f(t)$ is an ℓth order spline, we can reproduce it from its samples by taking $y_k = f(t_k)$ above.

Bases and discretization

All of the examples above have a common theme: we take a signal in a certain class (zero outside of $[0, T]$, polynomial, polynomial spline) and represent it using a discrete list of numbers $\{\alpha_k,\ k \in \mathbb{Z}\}$.

These numbers represent weights used to build up the signal out of a set of pre-determined building blocks ("basis functions"). This

framework gives us a systematic way to manipulate continuous time signals by operating on discrete vectors. This allows us to unleash the power of **linear algebra**.

It often times also gives us a straightforward way to simply or compress signals. As you can see from the sawtooth Fourier series example, although it technically takes an infinity of sinusoids to build up the signals, we can get away with 50 if we are willing to suffer some loss. We will see some more examples of this later in this section.

In this set of notes, we have just gotten our first taste of basis expansions. What we will do next is develop a systematic method for taking a function and breaking it down into a superposition of basis functions. We will also discuss how to optimally approximate a function using a fixed number of basis functions — this simple idea has an incredible number of applications.

To do these things correctly, we need to first build up some mathematical machinery so we can avoid talking in hazy terms. We start in the next section with precise (but abstract) definitions of **linear vector space**, **norm**, and **inner product**.

Exercises

1. Consider the set of signals that can be written as

$$f(t) = \alpha_0 b_2(t) + \alpha_1 b_2(t-1).$$

Suppose you know that $f(0) = 1$ and $f(1) = -1$. What must α_0 and α_1 be?

2. Consider the set of signals that can be written as

$$f(t) = \alpha_0 b_0(t) + \alpha_1 b_0(t-1) + \alpha_2 b_0(t-2) + \alpha_3 b_0(t-3).$$

How to I find the $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ such that $f(t)$ is as close[4] to $\cos(t)$ as possible?

[4] Of course, the answer will depend on how you define "close"...