

## ECE 3640

### Lecture 3 — Building blocks for signals: Vector spaces

**Objective:** To learn about how to construct signals (and other things) from basic building blocks as a preparation for studying Fourier series.

## An informal example

Before getting to that formality, an analogy will be presented.

We will consider making things out of a bunch of building blocks. The things might be, for example cakes, shampoo, or signals. For the case of cakes, the set of possible ingredients might look like this:

Ingred. symbol	Name
$\mathbf{i}_1$	white flour
$\mathbf{i}_2$	wheat flour
$\mathbf{i}_3$	granulated sugar
$\mathbf{i}_4$	Sweet and Low
$\mathbf{i}_5$	Powdered sugar
$\mathbf{i}_6$	Baking soda
$\mathbf{i}_7$	Baking power
$\mathbf{i}_8$	Hershey's cocoa
$\mathbf{i}_9$	egg
$\mathbf{i}_{10}$	milk
$\mathbf{i}_{11}$	water
$\mathbf{i}_{12}$	vegetable oil
$\vdots$	$\vdots$
$\mathbf{i}_N$	Xanthum gum.

The ingredients are denoted by  $\mathbf{i}_k$ . The mixture for the cake batter consists of certain amounts of each ingredient. The ingredients for a cake might be, for example: 500 ml white flour, 300 ml granulated sugar, 2 eggs, 20 ml vegetable oil, 200 ml water, and 10 ml baking powder. (Hint: don't try to make this at home!). Assuming that the quantities in the table above are placed in the correct units ("normalized"), this recipe could be written as follows:

$$\mathbf{c} = 500\mathbf{i}_1 + 300\mathbf{i}_3 + 10\mathbf{i}_7 + 2\mathbf{i}_9 + 200\mathbf{i}_{11} + 20\mathbf{i}_{12}.$$

The set of all possible cakes forms a "**vector space**". If the set of ingredients is able to make every element in the space (i.e. every cake), the said of ingredients is **complete**. Notice that a complete cake space is not necessarily able to make everything else: we could not, for example, make every possible shampoo with the set of ingredients to make cakes. (We don't have, for example, any aloe in our list above, or even any FD&C Red # 21.)

Several interesting questions now arise. Given a cake, is it possible to determine the quantities of each ingredient that goes into it? (This is the analysis question.) Suppose that we only want to use a certain subset of the ingredients (say, we have run out of ingredients and don't want to run to the store). What is the best approximation to a desired cake that we can make? What is the error between the desired cake and the cake we actually get? (This is the approximation question.) Obviously, some of these questions don't make a lot of sense when applied to cakes. However, these kinds of things will be very applicable when it comes to analyzing signals, which may also be built up from a set of building blocks.

## Vector spaces

We briefly review the concept of a vector space. A vector space  $V$  has the following key property: If  $v, w \in V$  then  $av + bw \in V$  for any scalars  $a$  and  $b$ . That is, linear combinations of vectors give vectors.

Most of your background with vectors has been for vectors in  $\mathbb{R}^n$ . But: **the signals that we deal with are also elements of a vector space**, since linear combinations of signals also gives a signal. This is a very important and powerful idea.

Recall that in vector spaces we deal with concepts like the **length** of a vector, the **angle** between vectors, and the idea of **orthogonal** vectors. **All** of these concepts carry over, by suitable definitions, to vector spaces of signals.

This powerful idea captures most of the significant and interesting notions in signal processing, controls, and communications. This is really the reason why the study of linear algebra is so important.

In this lecture we will learn about geometric representations of signals via signal space (vector) concepts. This straightforward idea is the key to a variety of topics in signals and systems:

1. It provides a distance concept useful in many pattern recognition techniques.
2. It is used in statistical signal processing for the filtering, smoothing, and prediction of noisy signals.
3. It forms the heart and geometric framework for the tremendous advances that have been made in digital communications.
4. It is every waveform-based transform you ever wanted (Fourier series, FFT, DCT, wavelet, etc.)
5. It is also used in the solution of partial differential equations, etc.
6. It relies on our old friend, linearity. One might even say it is the reason that we care so much about linearity in the first place.

We will soon turn our attention to Fourier series, which are a way of analyzing and synthesizing signals.

Vectors will be written in **bold** font (like the ingredients above). Initially, we can think of a vector  $\mathbf{v}$  as an ordered set of  $n$  numbers, written in a column:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

Often to conserve writing, this will be written in transposed form,

$$\mathbf{v} = [v_1, v_2, \dots, v_n]^T.$$

Vector addition is component-by-component.

While we have written a vector as an  $n$ -tuple, that is not what defines a vector. A vector is an element of a vector space, which is to say, it satisfies the linearity property given above.

Scalar multiplication of vectors is in the usual fashion. Matrix multiplication is also taken in the traditional manner.

Let

$$\mathbf{g} = [g_1, g_2, \dots, g_n]^T$$

and

$$\mathbf{f} = [f_1, f_2, \dots, f_n]^T$$

be two vectors. The **inner product** (known to many of you as the **dot product**) of the vectors  $\mathbf{g}$  and  $\mathbf{f}$  is written as

$$\langle \mathbf{g}, \mathbf{f} \rangle = \mathbf{g}^T \mathbf{f} = \sum_{i=1}^n g_i f_i.$$

In words, multiply component by component, and add them up. Two vectors are said to be **orthogonal** or perpendicular if their inner product is zero:

$$\langle \mathbf{f}, \mathbf{g} \rangle = 0 \quad \text{iff } \mathbf{f} \text{ and } \mathbf{g} \text{ are orthogonal.}$$

If  $\mathbf{f}$  and  $\mathbf{g}$  are orthogonal, this is sometimes written  $\mathbf{f} \perp \mathbf{g}$ .

**Example 1** Let  $\mathbf{f} = [1, -2, 3]$  and  $\mathbf{g} = [-3, 3, 3]$ . Then

$$\langle \mathbf{f}, \mathbf{g} \rangle = 0$$

so  $\mathbf{f}$  and  $\mathbf{g}$  are orthogonal. □

The inner product can be expanded using the following rules:

1. For a scalar  $c$ ,

$$\langle c\mathbf{f}, \mathbf{g} \rangle = c\langle \mathbf{f}, \mathbf{g} \rangle$$

- 2.

$$\langle \mathbf{f} + \mathbf{g}, \mathbf{h} \rangle = \langle \mathbf{f}, \mathbf{h} \rangle + \langle \mathbf{g}, \mathbf{h} \rangle$$

3. For real vectors (which is all we will be concerned about for the moment)

$$\langle \mathbf{f}, \mathbf{g} \rangle = \langle \mathbf{g}, \mathbf{f} \rangle.$$

The (Euclidean) norm of a vector is given by

$$\|\mathbf{x}\| = \langle \mathbf{x}, \mathbf{x} \rangle^{1/2} = \left( \sum_{i=1}^n x_i^2 \right)^{1/2}$$

The distance between two vectors is given by

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \left( \sum_{i=1}^n (x_i - y_i)^2 \right)^{1/2}.$$

The projection of a vector  $\mathbf{f}$  onto a vector  $\mathbf{x}$  is given by

$$\text{proj}(\mathbf{f}, \mathbf{x}) = \frac{\langle \mathbf{f}, \mathbf{x} \rangle}{\|\mathbf{x}\|^2} \mathbf{x} = \frac{\langle \mathbf{f}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} \mathbf{x}$$

Geometrically, this is the amount of the vector  $\mathbf{f}$  in the direction of  $\mathbf{x}$ . (Show a picture.) Obviously, if  $\mathbf{f}$  and  $\mathbf{x}$  are orthogonal, then the projection of  $\mathbf{f}$  onto  $\mathbf{x}$  is 0.

Now suppose that we have a vector  $\mathbf{i}_1$  (an “ingredient”) and we have a vector  $\mathbf{f}$  and we want to make the best approximation to  $\mathbf{f}$  using some amount of our ingredient. Draw a picture. We can write

$$\mathbf{f} = c_1 \mathbf{i}_1 + \mathbf{e}_1$$

where  $c_1$  is the amount of  $\mathbf{i}_1$  we want and  $\mathbf{e}_1$  is the error between the thing we want and our approximation of it. To get the best approximation we want to minimize the length of the error vector. Before we go through and do it the hard way, let us make a geometric observation. The length of the error is minimized when the error vector is orthogonal to our ingredient vector  $\mathbf{i}_1$ :

$$\mathbf{i}_1 \perp \mathbf{e}_1.$$

or

$$\begin{aligned} (\mathbf{f} - c_1 \mathbf{i}_1) &\perp \mathbf{i}_1 \\ \langle (\mathbf{f} - c_1 \mathbf{i}_1), \mathbf{i}_1 \rangle &= 0 \\ \langle \mathbf{f}, \mathbf{i}_1 \rangle &= c_1 \langle \mathbf{i}_1, \mathbf{i}_1 \rangle \end{aligned}$$

Giving us

$$c_1 = \frac{\langle \mathbf{f}, \mathbf{i}_1 \rangle}{\langle \mathbf{i}_1, \mathbf{i}_1 \rangle} = \frac{\langle \mathbf{f}, \mathbf{i}_1 \rangle}{\|\mathbf{i}_1\|^2}$$

Note that this is simply the projection of  $\mathbf{f}$  onto the vector  $\mathbf{i}_1$ .

**Example 2** Suppose  $\mathbf{i} = [1, 1]^T$  and  $\mathbf{f} = [3, 4]^T$ . The representation

$$\mathbf{f} \approx c\mathbf{i}$$

has the least amount of error when

$$c = \frac{\langle [1, 1]^T, [3, 4]^T \rangle}{\langle [1, 1]^T, [1, 1]^T \rangle} = \frac{7}{2}.$$

□

Now let's do it the hard way: we want to find the amount of  $\mathbf{i}_1$  to minimize the (length of the) error. The squared length of the error is

$$E = \|\mathbf{e}_1\|^2 = \|\mathbf{f} - c_1 \mathbf{i}_1\|^2 = \langle \mathbf{f} - c_1 \mathbf{i}_1, \mathbf{f} - c_1 \mathbf{i}_1 \rangle = \langle \mathbf{f}, \mathbf{f} \rangle - 2c_1 \langle \mathbf{f}, \mathbf{i}_1 \rangle + c_1^2 \langle \mathbf{i}_1, \mathbf{i}_1 \rangle.$$

To minimize this, take the derivative with respect to the coefficient and equate to zero:

$$\frac{dE}{dc_1} = -2\langle \mathbf{f}, \mathbf{i}_1 \rangle + 2c_1 \langle \mathbf{i}_1, \mathbf{i}_1 \rangle = 0$$

Solving for the coefficient,

$$c_1 = \frac{\langle \mathbf{f}, \mathbf{i}_1 \rangle}{\langle \mathbf{i}_1, \mathbf{i}_1 \rangle}$$

This is the same one we got before.

We may actually have more than one “ingredient” vector to deal with. Suppose we want to approximate  $\mathbf{f}$  with the vectors  $\mathbf{i}_1$  and  $\mathbf{i}_2$ . As before write

$$\mathbf{f} = c_1 \mathbf{i}_1 + c_2 \mathbf{i}_2 + \mathbf{e}$$

where  $\mathbf{e}$  is the error in the approximation. Note that we can write this in the following way:

$$\mathbf{f} = \begin{bmatrix} \mathbf{i}_1 & \mathbf{i}_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \mathbf{e}.$$

using the usual matrix multiplication. We want to find the coefficients  $c_1$  and  $c_2$  to minimize the length of the error. We could do it the calculus way, or using our orthogonality idea. We will go for the latter: **The error is orthogonal to the data** means that

$$\langle \mathbf{f} - c_1 \mathbf{i}_1 - c_2 \mathbf{i}_2, \mathbf{i}_1 \rangle = 0$$

$$\langle \mathbf{f} - c_1 \mathbf{i}_1 - c_2 \mathbf{i}_2, \mathbf{i}_2 \rangle = 0$$

(that is, the error is orthogonal to each of the ingredient “data” points). Expanding these out gives

$$\langle \mathbf{f}, \mathbf{i}_1 \rangle = c_1 \langle \mathbf{i}_1, \mathbf{i}_1 \rangle + c_2 \langle \mathbf{i}_2, \mathbf{i}_1 \rangle$$

$$\langle \mathbf{f}, \mathbf{i}_2 \rangle = c_1 \langle \mathbf{i}_1, \mathbf{i}_2 \rangle + c_2 \langle \mathbf{i}_2, \mathbf{i}_2 \rangle$$

This is two equations in two unknowns that we can write in the form

$$\begin{bmatrix} \langle \mathbf{i}_1, \mathbf{i}_1 \rangle & \langle \mathbf{i}_2, \mathbf{i}_1 \rangle \\ \langle \mathbf{i}_1, \mathbf{i}_2 \rangle & \langle \mathbf{i}_2, \mathbf{i}_2 \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \langle \mathbf{f}, \mathbf{i}_1 \rangle \\ \langle \mathbf{f}, \mathbf{i}_2 \rangle \end{bmatrix}.$$

If we know  $\mathbf{f}$  and the ingredient vectors, we can solve for the coefficients.

**Example 3** Suppose  $\mathbf{f} = [1, 2, 3]^T$  and  $\mathbf{i}_1 = [1, 1, 0]^T$  and  $\mathbf{i}_2 = [2, 1, 0]^T$ . It is clear that there is no exact way to represent

$$\mathbf{f} = c_1 \mathbf{i}_1 + c_2 \mathbf{i}_2$$

since there is no way to match the third element. The formula for the best coefficient given above leads to

$$\begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

This can be solved to give

$$c_1 = 3 \quad c_2 = -1.$$

Then the approximation vector is

$$\hat{\mathbf{f}} = c_1 \mathbf{i}_1 + c_2 \mathbf{i}_2 = 3[1, 1, 0]^T - [2, 1, 0]^T = [1, 2, 0]^T.$$

Note that the approximation  $\hat{\mathbf{f}}$  is the same as  $\mathbf{f}$  in the first two coefficients. What has happened is the vector has been **projected** onto the plane formed by the vectors  $\mathbf{i}_1$  and  $\mathbf{i}_2$ . The error in this case has length 3.  $\square$

Of course, what we can do for two ingredient vectors, we can do for  $n$  ingredient vectors (and  $n$  may be infinite). We want to approximate  $\mathbf{f}$  as

$$\mathbf{f} = \sum_{i=1}^n c_i \mathbf{i}_i + \mathbf{e}$$

We can find the set of coefficients that minimize the length of the error  $\mathbf{e}$  using the orthogonality principle as before, applied  $n$  times. This gives us  $n$  equations in the  $n$  unknowns which may be written as

$$\begin{bmatrix} \langle \mathbf{i}_1, \mathbf{i}_1 \rangle & \langle \mathbf{i}_2, \mathbf{i}_1 \rangle & \cdots & \langle \mathbf{i}_n, \mathbf{i}_1 \rangle \\ \langle \mathbf{i}_1, \mathbf{i}_2 \rangle & \langle \mathbf{i}_2, \mathbf{i}_2 \rangle & \cdots & \langle \mathbf{i}_n, \mathbf{i}_2 \rangle \\ \vdots & & & \vdots \\ \langle \mathbf{i}_1, \mathbf{i}_n \rangle & \langle \mathbf{i}_2, \mathbf{i}_n \rangle & \cdots & \langle \mathbf{i}_n, \mathbf{i}_n \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \langle \mathbf{f}, \mathbf{i}_1 \rangle \\ \langle \mathbf{f}, \mathbf{i}_2 \rangle \\ \vdots \\ \langle \mathbf{f}, \mathbf{i}_n \rangle \end{bmatrix}.$$

This could be readily solved (say, using Matlab).

It would seem that if we take  $n$  large enough, we should be able to represent any vector, without any error. (Analogy: given enough ingredients, we could make *any* cake. We might not be able to make everything, but we could make everything some class of objects.) If this is true, the set of ingredient vectors are said to be **complete**. A more formal name for the ingredient vectors is **basis vectors**.

Although we have come up with a way of doing the approximation, there is still a lot of work to solve for the coefficients, since we have to first find a matrix and

then invert it. Something that is commonly done is to choose a set of basis vectors that is **orthogonal**. That is, if  $\mathbf{i}_j$  and  $\mathbf{i}_k$  are any pair of basis vectors, then

$$\langle \mathbf{i}_j, \mathbf{i}_k \rangle = 0 \quad \text{if } j \neq k$$

Let us return to the case of two basis vectors when the vectors are orthogonal. Then the equation for the coefficients becomes

$$\begin{bmatrix} \langle \mathbf{i}_1, \mathbf{i}_1 \rangle & 0 \\ 0 & \langle \mathbf{i}_2, \mathbf{i}_2 \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \langle \mathbf{f}, \mathbf{i}_1 \rangle \\ \langle \mathbf{f}, \mathbf{i}_2 \rangle \end{bmatrix}.$$

or

$$c_1 \langle \mathbf{i}_1, \mathbf{i}_1 \rangle = \langle \mathbf{f}, \mathbf{i}_1 \rangle$$

$$c_2 \langle \mathbf{i}_2, \mathbf{i}_2 \rangle = \langle \mathbf{f}, \mathbf{i}_2 \rangle$$

so the coefficients are

$$c_1 = \frac{\langle \mathbf{f}, \mathbf{i}_1 \rangle}{\langle \mathbf{i}_1, \mathbf{i}_1 \rangle} \quad c_2 = \frac{\langle \mathbf{f}, \mathbf{i}_2 \rangle}{\langle \mathbf{i}_2, \mathbf{i}_2 \rangle}$$

So solving for the coefficients in this case is as easy as doing it for the case of a single vector, and the coefficient is simply the projection of  $\mathbf{f}$  onto its corresponding basis vector. This generalizes to  $n$  basis vectors: If the basis vectors are orthogonal, then the coefficient is simply

$$c_n = \frac{\langle \mathbf{f}, \mathbf{i}_n \rangle}{\langle \mathbf{i}_n, \mathbf{i}_n \rangle}.$$

**Example 4** Let us repeat the previous example, then  $\mathbf{f} = [1, 2, 3]^T$ , only this time let  $\mathbf{i}_1 = [1, 1, 0]^T$  and  $\mathbf{i}_2 = [3, -3, 0]$ . Observe that  $\mathbf{i}_1$  and  $\mathbf{i}_2$  are orthogonal. Then

$$c_1 = \frac{3}{2} \quad c_2 = \frac{-3}{18} = -\frac{1}{6}.$$

The approximation is

$$\hat{\mathbf{f}} = c_1 \mathbf{i}_1 + c_2 \mathbf{i}_2 = [1, 2, 0]^T.$$

The approximation is the same as before, but the computations are easier because the ingredient vectors are easier.  $\square$

## Function spaces

One of the neat things about all of this is that we can do the same techniques for infinite-dimensional spaces, which includes spaces of functions. Now our ingredients are not vectors in the usual sense, but functions. Suppose we have a set of ingredient functions called  $i_1(t), i_2(t), \dots, i_n(t)$ . We want to find a representation of some other function  $f(t)$  in terms of these functions:

$$f(t) = \sum_{k=1}^n c_k i_k(t)$$

We can ask the same kinds of questions as before: What kinds of functions can be represented? How do we choose the set of coefficients? To get the whole thing going, we need to introduce some kind of geometry into the problem. We will define the inner product between two functions as

$$\langle f(t), g(t) \rangle = \int_{-\infty}^{\infty} f(t)g(t) dt$$

(the limits will usually be dropped for ease of writing, indicating that the integral is to be taken over all applicable values). Compare this inner product with the inner product from before: we are multiplying things together and adding them up. Two functions are said to be orthogonal if

$$\langle f(t), g(t) \rangle = 0.$$

The projection of one function onto another one is defined as before:

$$\text{proj}(f(t), g(t)) = \frac{\langle f(t), g(t) \rangle}{\langle g(t), g(t) \rangle}$$

The length of a function is defined by

$$\|f(t)\| = \langle f(t), f(t) \rangle^{1/2} = \left( \int f^2(t) dt \right)^{1/2}$$

and the distance between two functions is defined as

$$d(f, g) = \langle f - g, f - g \rangle^{1/2}.$$

Getting back to our representation,

$$f(t) = \sum_{k=1}^n c_k i_k(t)$$

the kinds of functions that we can represent without error depends upon the kinds of basis functions that we choose. If the set of basis functions is able to represent all functions in a given class, then the set of functions is said to be complete (even if it cannot produce all possible functions. Analogy: a set of ingredients for cakes may be complete, even if it cannot produce all kinds of shampoo.)

To find the coefficients, we proceed as we did in the vector case: we want the error between the function  $f(t)$  and its representation to be as small as possible. This produces again the **orthogonality theorem**: the error is orthogonal to the data. This leads to the following equation for the  $n$  coefficients:

$$\begin{bmatrix} \langle i_1(t), i_1(t) \rangle & \langle i_2(t), i_1(t) \rangle & \cdots & \langle i_n(t), i_1(t) \rangle \\ \langle i_1(t), i_2(t) \rangle & \langle i_2(t), i_2(t) \rangle & \cdots & \langle i_n(t), i_2(t) \rangle \\ \vdots & & & \vdots \\ \langle i_1(t), i_n(t) \rangle & \langle i_2(t), i_n(t) \rangle & \cdots & \langle i_n(t), i_n(t) \rangle \end{bmatrix} \begin{bmatrix} c_1(t) \\ c_2(t) \\ \vdots \\ c_n(t) \end{bmatrix} = \begin{bmatrix} \langle f(t), i_1(t) \rangle \\ \langle f(t), i_2(t) \rangle \\ \vdots \\ \langle f(t), i_n(t) \rangle \end{bmatrix}.$$

Comparison of this equation with the one above reveals that they are identical in form: **It doesn't matter whether you are dealing with vectors or functions: the result is the same.** This means that you can use any geometric insight that you might obtain from vectors and apply it to functions when represented in this way. This is an extremely powerful notion and, in a sense, forms the very heart of digital communications and a good part of signal processing theory.

As before, it is often convenient to deal with orthogonal functions. Suppose that the set of basis functions that we choose is orthogonal, so that

$$\langle i_j(t), i_k(t) \rangle = 0 \quad \text{when } j \neq k.$$

Then the equation for the coefficients reduces down to

$$\begin{bmatrix} \langle i_1(t), i_1(t) \rangle & 0 & \cdots & 0 \\ 0 & \langle i_2(t), i_2(t) \rangle & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & \langle i_n(t), i_n(t) \rangle \end{bmatrix} \begin{bmatrix} c_1(t) \\ c_2(t) \\ \vdots \\ c_n(t) \end{bmatrix} = \begin{bmatrix} \langle f(t), i_1(t) \rangle \\ \langle f(t), i_2(t) \rangle \\ \vdots \\ \langle f(t), i_n(t) \rangle \end{bmatrix}.$$

The coefficients may then be solved for as

$$c_k = \frac{\langle f(t), i_k(t) \rangle}{\langle i_k(t), i_k(t) \rangle}.$$

**Example 5** Suppose we have a function

$$f(t) = \begin{cases} 1 & 0 \leq t \leq \pi \\ -1 & \pi < t \leq 2\pi \end{cases}$$

Suppose we want to find an approximation to  $f(t)$  using only the function  $i_1(t) = \sin(t)$ , and we want to choose the coefficient  $c$  so that the approximation is as close as possible to the original function. That is,

$$f(t) \approx c \sin(t)$$

The best coefficient (in the minimum error sense) is

$$c = \frac{\int_0^{2\pi} f(t) \sin(t) dt}{\int_0^{2\pi} \sin^2(t) dt} = \frac{1}{\pi} \left[ \int_0^{\pi} \sin(t) dt + \int_{\pi}^{2\pi} -\sin(t) dt \right] = \frac{4}{\pi}.$$

So the best approximation is

$$f(t) \approx \frac{4}{\pi} \sin(t)$$

Now suppose we want to approximate  $f(t)$  with two functions,

$$i_1(t) = \sin(t)$$

$$i_2(t) = \sin(2t)$$

$$f(t) \approx c_1 i_1(t) + c_2 i_2(t)$$

and we want to choose the coefficients so the error is as small as possible. Note that  $i_1(t)$  and  $i_2(t)$  are orthogonal. In fact,

$$\int_0^{2\pi} \sin(mt) \sin(nt) dt = \begin{cases} 0 & m \neq n \\ \pi & m = n \end{cases}$$

So we can use the simple formula

$$c_k = \frac{\int_0^{2\pi} f(t) i_k(t) dt}{\int_0^{2\pi} i_k^2(t) dt}$$

This gives us

$$c_1 = \frac{4}{\pi} \quad c_2 = 0.$$

Note that  $c_1$  has the same value as it did before: this always happens when we use orthogonal functions. We will learn later what it means that  $c_2 = 0$ .  $\square$

Now we are ready to deal with Fourier series!