1 Travelling Wave Pulses (Part I)

In typical programming style, this week’s project will build on the code and ideas that we have been developing in the last month. Our goal is to implement Fourier analysis to a pulse shape of our choosing, and then create a visualisation as we allow the pulse to propagate.

1.1 Fourier Analysis

Fourier analysis is built on the orthogonality of the \( \{ \sin nx, \cos nx \} \) functions. To determine the contribution from each component making up a generic, nice function \( f(x) \), we take the inner product: \( a_n = \langle f(x) | \sin nx \rangle \).

Recall that the Fourier series for the function \( f(x) \) defined over the interval \([0, D]\) is given by:

\[
f(x) = b_0 + \sum_{n=1}^{\infty} \left[ a_n \sin \left( \frac{2\pi nx}{D} \right) + b_n \cos \left( \frac{2\pi nx}{D} \right) \right]
\]

where

\[
b_0 = \frac{1}{D} \int_0^D f(x) \, dx
\]

\[
a_n = \frac{1}{D/2} \int_0^D f(x) \sin \left( \frac{2\pi nx}{D} \right) \, dx
\]

and

\[
b_n = \frac{1}{D/2} \int_0^D f(x) \cos \left( \frac{2\pi nx}{D} \right) \, dx
\]

In the case of a continuous function \( f(x) \), the inner products are expressed as integrals. However, from a discrete perspective, we can write

\[
a_n = \frac{\langle f(x) | \cos nx \rangle}{\langle \sin nx | \cos nx \rangle} \quad \text{and} \quad b_n = \frac{\langle f(x) | \cos nx \rangle}{\langle \cos nx | \cos nx \rangle}.
\]

2 The Plan

We will use our previous code to build a function that returns the normal mode \( n \) for the entire system as an array. (This is like a vector representing the \( n^{th} \) mode.)

After some warm-up exercises working with inner products, we will select a specific wave pulse and extract its first few Fourier components. Once we calculate those contributions, we will reconstruct the wave pulse using the normal modes (i.e., Fourier components). By including the time dependence of the normal modes, we will then visualise the pulse as it propagates in time.

This exercise will be done in pairs, as an exercise in pair programming.

3 Pair Programming

Pair programming is “two people working together at a single computer”. The practice has been popularized by a software development methodology called Extreme Programming (XP).  

---

1 ref. Pair-Programming in CS 17, Brown University
There are many advantages to pair programming, including

- You will produce better code;
- You will learn more, sharing your ideas with your peers and benefiting from their insights;
- You will become better at articulating your thoughts;
- You will enjoy your work more and spend less time frustrated;
- You will be better prepared for more complicated tasks, beyond this course.

When two programmers work with one keyboard, a division of labor is necessary. Two roles are defined and the programmers alternate working in these positions every 10 to 15 minutes.

**Driver:** responsible for typing, moving the mouse, etc.

**Navigator:** responsible for reviewing the Driver’s work; catching little mistakes, typos, etc. The navigator considers the code at a more strategic level:
- How will this fit with the rest of the code?
- Will this implementation require changes elsewhere?
- Could we design this program better?

NB: “Divide-and-Conquer” is NOT pair-programming.

## 4 Inner Products: Warm-Up

Let’s consider an example involving inner products in 3-dimensions. NB: In this context, inner products are also called *dot products.*

Let $v = (7, -2, 3)$. If we wanted to write $v$ in the standard basis

$$\{b_1, b_2, b_3\} = \{(1,0,0), (0,1,0), (0,0,1)\},$$

then that is easy: $v = 7b_1 - 2b_2 + 3b_3$. Suppose we wanted to write it in another basis, such as

$$\{c_1, c_2, c_3\} = \{(1,2,3), (1,1,-1), (5,-4,1)\}.$$  

In this case writing any vector in terms of that basis is easy because that basis is *orthogonal:* we have

$$\langle c_i, c_j \rangle = 0 \quad \text{if} \quad i \neq j.$$  

Check that the given basis is orthogonal (there are three things to verify).

If we want to write $v = \sum_{j=1}^{3} a_j c_j$, then using exactly the same reasoning as in the case of Fourier analysis, we have

$$a_j = \frac{\langle v, c_j \rangle}{\langle c_j, c_j \rangle}.$$  

**Task 1:** By hand, calculate the inner products to determine the contributions to to $v$ from the basis set $\{c_i\}$.

**Task 2:** Open a python shell and perform the necessary calculations to extract the contributions to $v$ from the basis set $\{c_i\}$. 

Page 2 of 3
Task 3: Pair up with a partner and write a function that performs an inner product in the most general way possible, that is, allows for a basis set of any dimension. Have the function require two inputs: the general vector (of any dimension) and the basis vector of interest. Check this with the case of the above example.

Task 4: With your partner, write a function that extracts the contribution of a particular basis vector for a given general vector. Again, check this with the case of the above example. Try another example.

5 Final Report (due Monday, March 7)

Submit Tasks 1, 3, and 4. Only one submission per pair is necessary, but be sure to include both authors in your opening commentary. (Task 1 may be submitted on paper.)