# Numerical Methods

**Math 3338 – Spring 2022**

## Worksheet 25

## Discrete Fourier Transforms

## 1 Reading

| | |
|---|---|
| CP | 7.2, 7.3, 7.4 |
| NMEP | - |

Table 1: Sections Covered

## 2 Motivation

Most functions that you'll encounter don't have a nice closed form expression. In fact, most functions will be given by sampling some event. For example, music. Music is typically sampled at 44.1 kHz or 44,100 sample per second. When you're listening to music you're not hearing a continuous stream of music you're hearing a new data point every 1/44,100 seconds. This is also common in the sciences when you are measuring an experiment.

Figure 1 shows an example of a function obtained through measurement. Notice this data is very noisy,

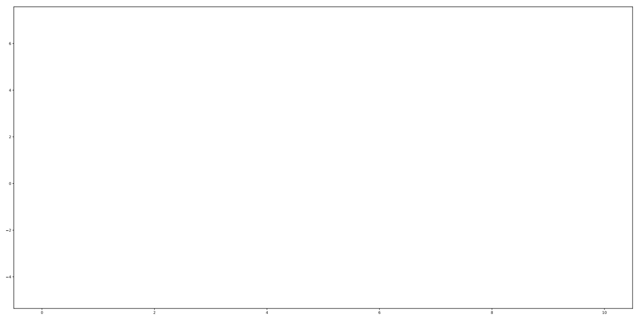


Figure 1: The raw data

there is definitely an underlying pattern but at the moment it is difficult to spot. This noise could be considered a consequence of a high frequency pattern in the data. If we had a way to identify the underlying frequencies and remove the higher ones, we could clean up this data.

Figure 2 shows the above data with any frequency with an amplitude less than 30 filtered out. In other words, frequencies that weren't contributing much to the data. Notice, the orange data is much closer to the exact black line. The orange data is constructed from 7 frequencies. There are 200 blue data points. This means we can reconstruct the 200 data points using 7 data points and get a "better" approximation of the actual curve. This is one of the key concept behind data compression, represent a lot of data with a little data.

## 3 The Discrete Cosine Transform

This was difficult to research. Any computer science source cared only about the implementation with limited detail on WHY it worked. Any math source cared only for the generalizations with very little

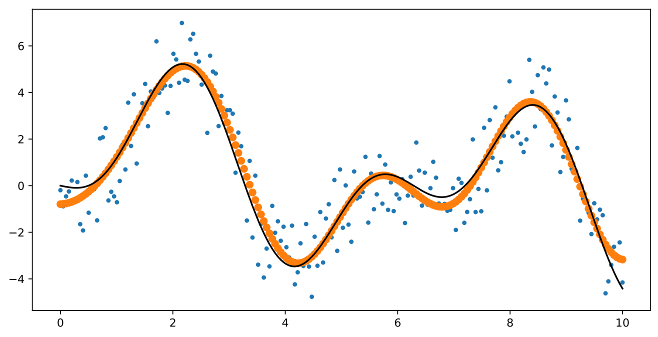


Figure 2: A better fit and the exact line

detail on HOW it worked. I'll try to bridge the gap.

Suppose we have a sequence of points $x_n$ sampled at equally spaced points. These $x_n$ could be the blue points in Figure 1. These $x_n$ are y-values, yes that is confusing. The corresponding $x$-values turn out to not be relevant, they disappear in the calculations. This does rely on the $x$-values being equally spaced, if the points aren't equally spaced it makes the computations more complicated[1].

We want to find the frequencies $k$ and the amplitudes $X_k$. The basic idea to start with the Fourier transformation from last time, change the integral to a sum, and do some algebra to simplify. This is fully worked in the text.

Suppose we have $N$ data points. The *discrete Fourier transform* is given by,

$$X_k = 2\sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k}{N}\left(n+\frac{1}{2}\right)\right) \tag{1}$$

and the inverse discrete Fourier transform is given by,

$$x_n = \frac{1}{2N}\left(X_0 + 2\sum_{k=1}^{N-1} X_k \cos\left(\frac{\pi k}{N}\left(n+\frac{1}{2}\right)\right)\right) \tag{2}$$

All this is doing is transforming the points $x_n$ into points $X_k$. The real power is what these point represent. The $X_k$ represent amplitudes of frequencies. By setting some of these to be zero (or deleting them) we can remove high (or low) frequency signals, or compress the data. That's how Figure 2 was created. Any frequency with an amplitude less than 30 was killed.

[1]The first step to solving non-uniform problems is a transformation to a uniform problem

# Numerical Methods

**Math 3338 – Spring 2022**

## Homework 25 (Due: Tuesday, April 19)

Include all graphs in your write up of the problems.

**Problem 1 (1 pt)** Implement two functions `discrete_fourier` and `inverse_discrete_fourier`.

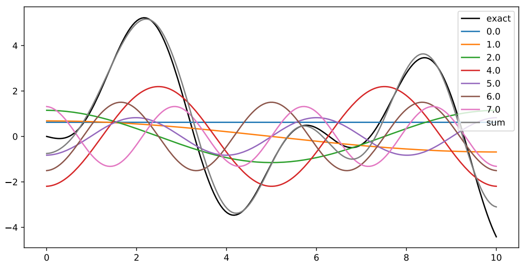
`discrete_fourier` The only input is an array $x$ that contains the $x$-values. The output should be an array with 2 columns, the first column is the frequency and the second is the $X_k$ values.

`inverse_discrete_fourier` The inputs should be a two column array (similar to the output of the previous) and the number of data points you want returned (this should be the length of the $x$ vector from the previous).

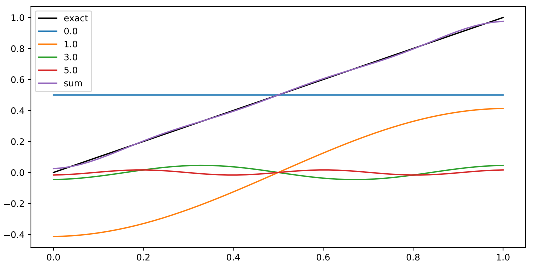
**Problem 2 (1 pt)** On Canvas you'll find a file called "prob2_data.txt". This is the data from Figure 1. Take the discrete Fourier transform of this data.

1. Make a plot of the amplitude vs the frequency.
2. Filter some of the higher frequencies (or lower amplitudes). You should use array manipulation for this. Check Worksheet 10 for a description. Do not use a list and do not use a for loop.
3. Run the inverse Fourier transformation on the filtered array.
4. Plot the data and the result of the inverse Fourier transform on the same axes.
5. Describe what you see.

**Problem 3 (1 pt)** The inverse Fourier transform is just a summation of a bunch of cosines of different frequencies. To prove this point, you'll be plotting each individual cosine graph, the exact function, and the sum of the cosines on the same axes. See Figure 3. Your goal is to recreate these images. Write a function called `base_curves` that has inputs $f$ (a function), $x$ (a sequence of $x$-values), max_amp (a cut-off value so you only plot certain frequencies), and out_name (defaults to `None`, the name of the saved graph if it's not None).



(a) The base frequencies



(b) The base frequencies

Figure 3: The Figures for Problem 3

The two functions depicted in the graphs are $\sin(x/2) + 3\sin(x) - 2\sin(2x)$ and $g(x) = x$. For $f(x)$ I used 200 data points between 0 and 10 and my cutoff amplitude was 100. For $g(x)$ I used 50 data points between 0 and 1 and my cutoff amplitude was .5.