Numerical Mathematics Assignment 4 Peter-Jan Derks

Exercise 1

The function [Fouriertransform] = cooleytuckey(inputvector,type) computes the fft or ifft of the inputvector. Inputvector should be a row vector with length of some power of 2. type should be either '1' or '2', if the users inputs 1, the discrete Fourier transform will be calculated, if the user selects 2 the inverse Fourier transform will be calculated. If the user does not enter 1 or 2 as the type an error message is displayed. The function returns the discrete or inverse Fourier transform a column vector.
An example of use of the function is:

Input:
```
>> p = [1:2^10]
>> cooleytuckey(p,1)
```

Output:
```
ans =

   1.0e+05 *

   5.2480 + 0.0000i
  -0.0051 + 1.6689i
  -0.0051 + 0.8344i
  -0.0051 + 0.5563i
```
……
(I decided not to show more rows of the output in this report because this example is just to show what the input and output looks like)

Description

The function calculates the Fourier transform recursively. The recursive step is the following: the input vector is split up into even and odd parts and the function itself is called on these even and odd parts. The base case is when the length of the input vector is 1, then both Fourier transforms are the input vector itself. From finding the Fourier transform of the vector with length 1 the Fourier transform of the next step can be calculated with the following formula:

$$inverse\ Fourier\ transform = [even + omega .*\ odd; even - omega\ .* odd]\ /2$$

$$with\ omega = e^{\frac{2\pi i}{N}}\ .\hat{}(0{:}\frac{N}{2} - 1)$$

$discrete\ Fouriertransform = [even + omega .*\ odd; even - omega\ .* odd]$
$with\ omega =\ e^{\frac{-2\,\pi\,i}{N}}.\hat{}(0{:}\frac{N}{2}-1)$

Picture

The following input produces the following picture:
>> time = lengthinputvector

time =

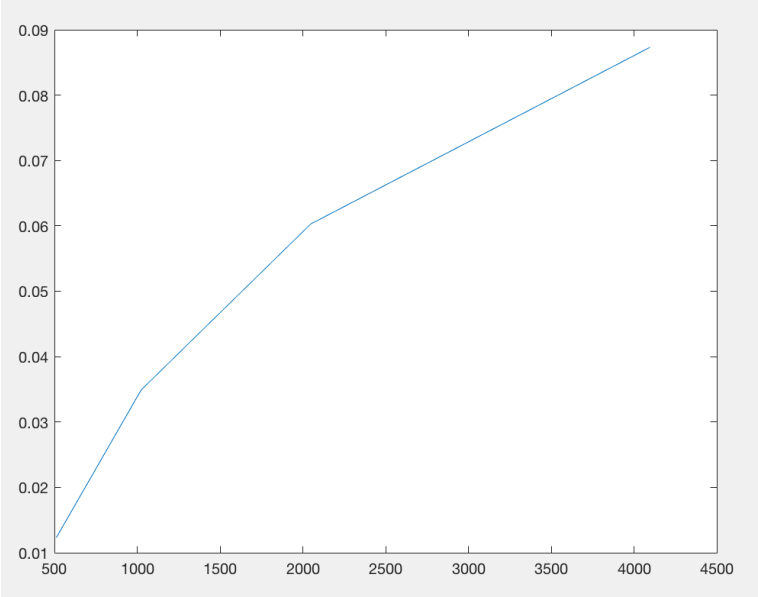
0.0123 0.0349 0.0603 0.0873

>> lengthinputvector = [512 1024 2048 4096]

lengthinputvector =

512 1024 2048 4096

>> plot(lengthinputvector, time)



I calculated the time it took my function to calculate the fft of a vector with a specific length using the tic toc function.

Questions

The discrete Fourier transform can be used to calculate the derivative in the following manner:

$$c_k(f') = \frac{2\pi}{L} ikc_k(f)\ \text{(from course notes page 34)}$$

$$f' =\ ifft(c_k(f'))$$

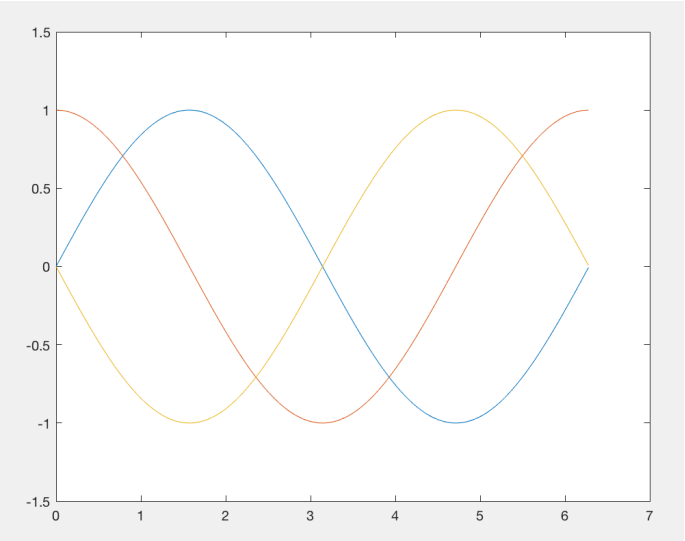
Similarly, the second derivative can be calculated:

$$c_k(f'') = -\frac{4\pi^2}{L^2} kc_k(f)\ \text{(from course notes page 34)}$$

$$f' =\ ifft(c_k(f'))$$

The following MATLAB code calculates the first and second derivative of sin(t):

```
L = 2*pi;
N = 1000;
dx = L/N;
t = dx*(0:N-1);
f = sin(t);
fftx = fft(f);
k =[0:N/2-1, 0, -N/2+1:-1];
dffft = 2*pi/L * 1i*k.*fftx;
df = ifft(dffft);
d2ffft = -4*pi^2/L^2 *
k.^2 .*fftx;
d2f = ifft(d2ffft);
plot(t,f,t,df,t,d2f)
```



Exercise 2

The function [digits] = phonenumber(signal, samplingrate) finds the phone number who's noise has been recorded. The input variable signal should be a one of the soundsignals from the files 'signal.mat' or 'phonenumbers.mat'. These soundsignals are column vectors with values ranging between -1 and 1. The sampling rate should be 8192 if the 'signal,mat' file is called and 4096 if the 'phonenumbers.mat' file is called. The function returns the fourier transform in a column vector.
An example of the use of the function is:

Input:
>> phonenumber(signal,8192)

Output:
ans =

1 5 0 8 6 4 7 7 0 0 1

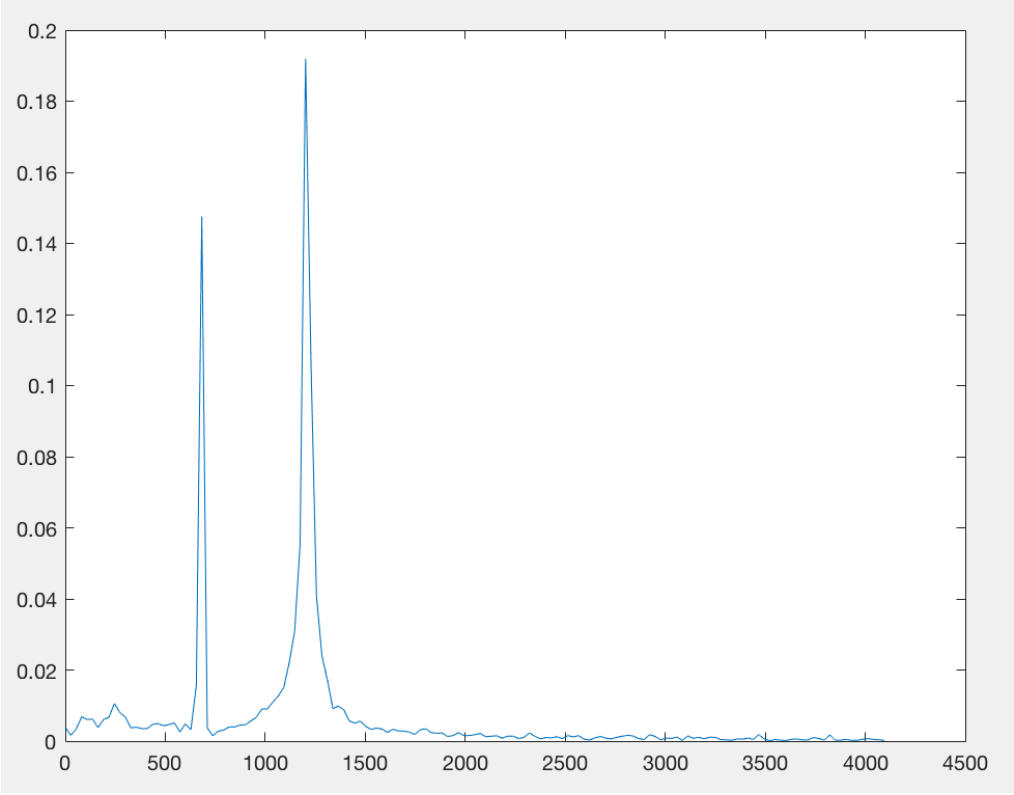
Description

The first step is to find the pieces in the signal which represent the digits (and to determine the number of digits). The function does this by finding the peaks of the sound signal. Each peak represents one digit and I found that these peaks can be found by looking for points of which the (absolute) sound is louder than 0.6. When the sound is louder than 0.6 the 300 datapoints coming after are put into a matrix called numbers. 2500 datapoints are skipt and using a while loop the function searches for the next datapoint with sound > 0.6

The second step is using the Fourier transform to extract the frequencies from the piece of signal representing one digit. The third step is to determine the digit associated with the frequencies. Both of these steps are done in a while loop for one piece of signal representing one digit at a time. The piece of signal representing one digit is split up into two parts. (I explain this in my picture section below). I split the signal into two parts so that I can use the max function two calculate both frequencies. I check for each of these frequencies which tone frequencies are closer using the min function. (line 64 and 65 of my code)

Picture

This is a plot of frequency of the signal piece representing phonenumber one from the signal.mat



I chose this picture to show that I can split the ‘P1’ vector in my function into two parts, at the point that represents f = 1100