
Table of Contents

| | |
|--|---|
| ECES T580 Lab 2 - Tyler Bradley | 1 |
| Lab 2.1.1 | 1 |
| Lab 2.1.2: FFT review: linspace, Fs, fftshift, NFFT: | 1 |
| Lab 2.2.1 | 3 |
| Lab 2.2.2 | 4 |
| Lab 2.2.3 | 5 |
| Functions for the Lab | 6 |

ECES T580 Lab 2 - Tyler Bradley

```
clc;close all;clear;
```

Lab 2.1.1

```
% 1. Enter hbb into the search - how many hits did you get?
% Performing this search resulted in thousands of results across
% multiple
% databases. There were 9,004 nucleotide, 1,561 genes, 11 genomes, and
% 8,678 proteins

% 2. Click on the nucleotide link. To the right (in the Top Organisms
% box),
% they suggest the human,mouse, chimpanzee, etc. HBB genes
% 3. Type hbb AND human[orgn] into the search. Now how many hits do
% you get?
% Note: the AND has to be capitalized. The human[orgn] means that you?
% re
% narrowing your search just to humans.
% There are 280 nucleotide results when typing in hbb AND human [ORGN]

% 4. Click on the HBB (Homo sapiens): hemoglobin, beta in the Gene
% search
% (you can also use the link: http://www.ncbi.nlm.nih.gov/gene/3043).
% Explore this record.
% 5. Search for hbb AND human[orgn] AND RefSeqGene in the Nucleotide
% search.
% (this should lead you to http://www.ncbi.nlm.nih.gov/
% nuccore/28380636)
% 6. We will actually be obtaining the whole region and doing analysis
% on
% the 81,706 bp linear DNA.
```

Lab 2.1.2: FFT review: linspace, Fs, fftshift, NFFT:

1. Generate one second of a cosine of $f_s = 10\text{Hz}$ sampled at $F_s = 100\text{Hz}$ and assign it to x. Define a tt as your time axis.

```

Fs = 100;
T = 1/Fs;
t = (0:Fs-1)*T;

x = cos(2*pi*10*t);

plot(t, x)
xlim([0, 1])
xlabel('time (sec)')

% 2. Take 64 points FFT.
fft_64 = fft(x, 64);
abs_Y = abs(fft_64);

% 3. As you remember, the DFT (which the FFT implements) computes N
    samples
% of  $k = 0, 1, 2, 3, \dots, N-1$ . Plot the magnitude of this 64-
points FFT at range 0 to 63, what do
% you think of this graph?
plot(0:63, abs_Y)
% The peaks of the graph appear to be where the wavelength of the
    cosine
% located

% 4. To get the x-axis into a Hz-frequency form, plot this 64-points
    FFT between -50 to 50 (the 100Hz
% sampling rate) and have N-points between them.
x_lims = linspace(-50, 50, 64);
plot(x_lims, fftshift(abs_Y));

% 5. According to your figure, what frequency is this cosine wave at?
% Answer: The frequency of the cosine is 10Hz

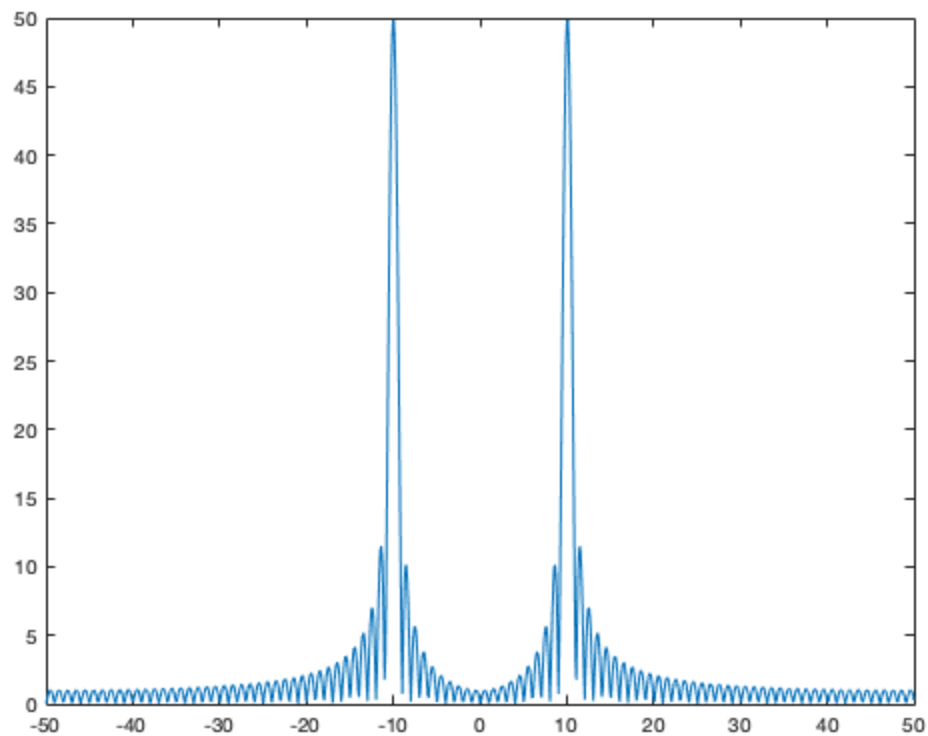
% 6. Remember that the FFT is evaluating from 0 to  $N-1$ . We are used to
    viewing graphs from -N/2 to N/2.
% Therefore, you need to shift your graph.
% This is done above in the graph from -50 to 50

% 7. Now according to your shifted graph. what frequency is this at?
% Answer: The frequency is 10 Hz

% 8. Note that the spikes have long drop-offs? Try a 1024-point DFT.
    Note that the peak is closer to
% 10 and the drop-off is quicker. Although, now sidelobes are an
    issue.
fft_1024 = fft(x, 1024);
abs_fft_1024 = abs(fft_1024);

x_lims = linspace(-50, 50, 1024);
plot(x_lims, fftshift(abs_fft_1024));

```



Lab 2.2.1

```
hbb = getgenbank("NG_000007.3");

% 1. How many CDS regions are there in this sequence? (A CDS sequence
% is a coding sequence that
% results in a protein product).
CDS = hbb.CDS;

% There are 5 CDS regions in this sequence

% 2. Look at hbb.CDS(1), what is the length of this region? How would
% you identify the coding and
% non-coding sequences for this hbb.CDS(1) region on Chromosome 11?
CDS(1);

% length of the region
CDS(1).indices(6) - CDS(1).indices(1)

% The coding regions are determined by looking at the sequence between
% the
% odd and even indices, while the non-coding regions is the portion
% between
% the odd indices + 1 and the even indices - 1
% CDS(1) first coding region:
```

```

hbb.Sequence(CDS(1).indices(1):CDS(1).indices(2));

% CDS(1) first non-coding region:
hbb.Sequence((CDS(1).indices(2)+1):(CDS(1).indices(3)-1));

% 3. Write a function that will take a sequence and CDS indices as
    input and outputs the corresponding
% coding and non-coding DNA sequences
% The code for get_coding function is at the bottom of this script
cdsl_regions = get_coding(hbb.Sequence, CDS(1).indices);

```

*Warning: The record NG_000007.3 has been replaced by NG_000007.
Returning record 28380636.*

```

ans =

    1421

```

Lab 2.2.2

```

coding = strjoin(cdsl_regions.coding, " ");

% This does not work
% coding_A = (upper(coding)=='A');

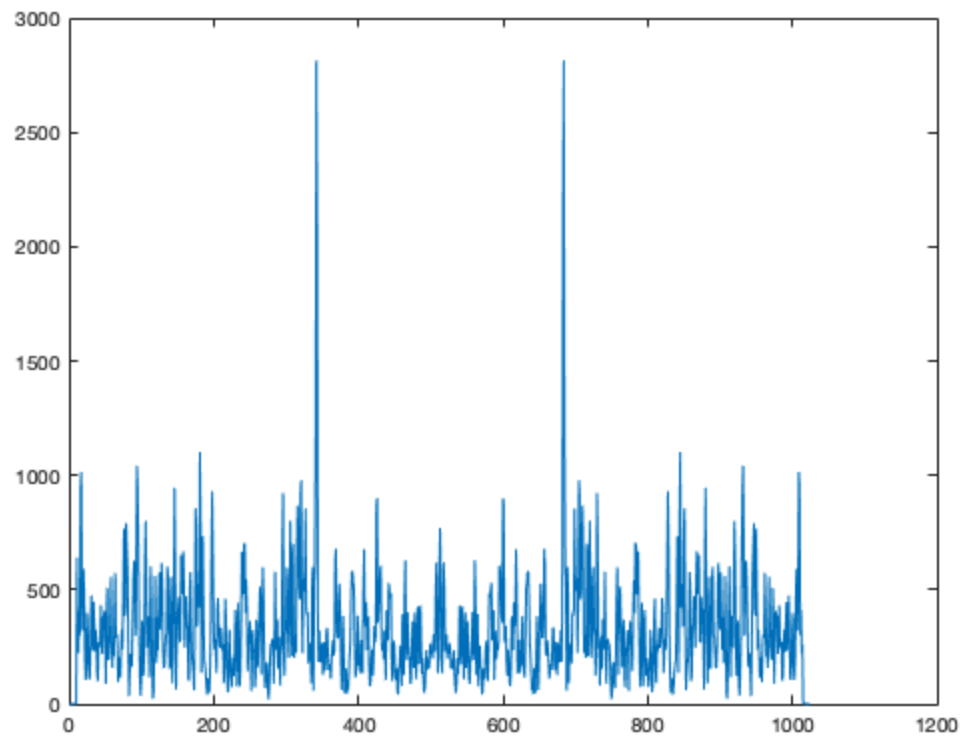
% so I wrote a function is_base to do this
% The code for this function can be found at the bottom of this file
coding_A = is_base(coding.char, "a");
coding_T = is_base(coding.char, "t");
coding_C = is_base(coding.char, "c");
coding_G = is_base(coding.char, "g");

% calculating the fft for the sequence
coding_FT = abs(fft(coding_A, 1024)).^2 + abs(fft(coding_T, 1024)).^2
    + abs(fft(coding_C, 1024)).^2 + abs(fft(coding_G, 1024)).^2;

% Set the first ten and last ten elements to zero so they do not weigh
% down the plot
coding_FT(1:10) = repelem(0, 10);
coding_FT(1015:1024) = repelem(0, 10);
figure(1)
plot(coding_FT)

nfft_coding_3 = coding_FT(round(1024/3));

```



Lab 2.2.3

Now, compute the magnitude spectrum of the noncoding sequence (use the same NFFT value as above)? what does it look like? What is the magnitude of the NFFT/3 point compared to the coding sequence? Show your two plots, explain the peaks to your instructor, and the difference between coding and noncoding regions.

```
non_coding = strjoin(cds1_regions.non_coding, " ");

non_coding_a = is_base(non_coding.char, "a");
non_coding_t = is_base(non_coding.char, "t");
non_coding_g = is_base(non_coding.char, "g");
non_coding_c = is_base(non_coding.char, "c");

non_coding_FT = abs(fft(non_coding_a, 1024)).^2 +
    abs(fft(non_coding_t, 1024)).^2 + ...
    abs(fft(non_coding_c, 1024)).^2 + abs(fft(non_coding_g, 1024)).^2;

nfft_3 = non_coding_FT(round(1024/3));

% Set the first ten and last ten elements to zero so they do not weigh
% down the plot
non_coding_FT(1:10) = repelem(0, 10);
non_coding_FT(1015:1024) = repelem(0,10);

subplot(2, 1, 1)
```

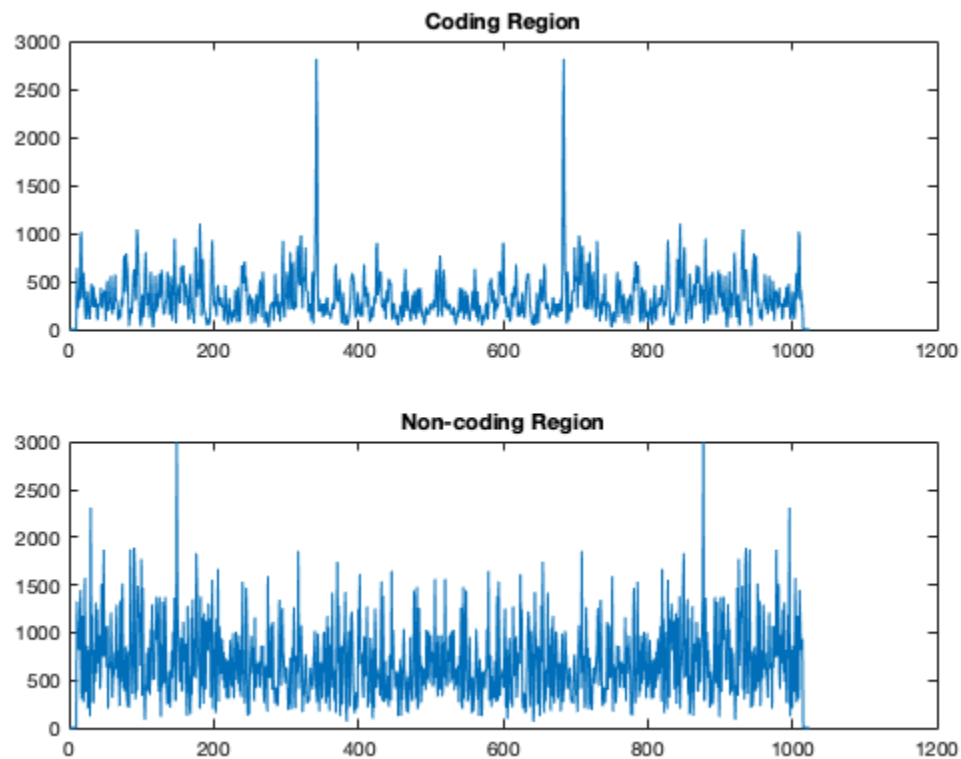
```

plot(coding_FT)
title("Coding Region")

subplot(2, 1, 2)
plot(non_coding_FT)
title("Non-coding Region")

% The non coding regions seems to have very minor peaks every 200
% bases but
% no clear spikes at NFFT/3. Conversely, the coding region has clear
% peaks
% at the NFFT/3 and 2*NFFT/3 positions.

```



Functions for the Lab

This function corresponds to Lab 2.2.1 #3 and returns the coding and non-coding regions of the input sequence in a struct object with coding and non_coding character arrays

```

function output = get_coding(seq, indices)
num_indices = length(indices);
coding = repelem("a", num_indices/2);
non_coding = repelem("a", (num_indices/2)-1);
n_code = 1;
n_non_code = 1;
for i = 1:num_indices-1
    if mod(i,2) == 1

```

```

        coding(n_code) = seq(indices(i):indices(i+1));
        n_code = n_code + 1;
    else
        non_coding(n_non_code) = seq(indices(i)+1:indices(i+1)-1);
        n_non_code = n_non_code+1;
    end
end

output.coding = coding;
output.non_coding = non_coding;
end

% This function is for Lab 2.2.2 and 2.2.3
% It takes the character string and looks at each of the bases and
% determines if it matches the specified base
function output = is_base(seq, base)
seq_length = length(seq);
out_binary = repelem(0, seq_length);

for i = 1:seq_length
    if seq(i) == base
        out_binary(i) = 1;
    else
        out_binary(i) = 0;
    end
end
output = out_binary;
end

% 1. Assume that the windows have full overlap meaning that each
    window of WINDOW_LENGTH
% overlaps by WINDOW_LENGTH ? 1 data points:
% 2. DNA_SEQUENCE is the DNA sequence of letters (before the binary
    indicator operation).
% 3. NFFT is the number of points to take in the Fourier transform.
% 4. The output will give you the magnitude of the N/3
% point for each position (or consecutive window) in the sequence.
function output = threebasefreq_stft(seq, window_length, nfft)
% calculate the sequence length
seq_length = length(seq);

% create a vector that is preallocated to the length of the expected
    output
fft_vec = repelem(0, seq_length-window_length);
for i = 1:seq_length-window_length
    % create a sub sequence
    sub_seq = seq(i:i+window_length);

    % calculate the u[base] for each of the bases
    sub_a = is_base(sub_seq, "a");
    sub_t = is_base(sub_seq, "t");
    sub_c = is_base(sub_seq, "c");
    sub_g = is_base(sub_seq, "g");

```

```
% calculate the fft
sub_ft = abs(fft(sub_a, nfft)).^2 + abs(fft(sub_t, nfft)).^2 + ...
abs(fft(sub_c, nfft)).^2 + abs(fft(sub_g, nfft)).^2;

% find the nfft/3
nfft_3 = sub_ft(round(nfft/3));
% square the result to make the effect more clear and put it into
the
% output vector
fft_vec(i) = nfft_3^2;
end
% assign the output vector
output = fft_vec;
end
```

Published with MATLAB® R2018b