

Lab 8 – CTFT, FFT and Quantization

8.1 Continuous-time Fourier transform

(a) Write a matlab function `X = continuousFT(t, xt, a, b, w)` to numerically compute continuous-time FT of the given signal $x(t)$ which has finite support in $[a, b]$ and is zero outside. The inputs to this function are

- t – symbolic variable
- xt – signal whose FT is to be computed (function of symbolic variable t)
- a, b – the signal is equal to xt in the interval $[a, b]$ and zero outside
- w – the vector w contains the values of frequency where FT is to be computed.

>> The function should return a vector X which contains the FT of $x(t)$ for each of the frequencies in the input vector w .

(b) Write a matlab script that calls the function `continuousFT` for a rectangular pulse of unit amplitude in $[-T, T]$ where $T = 2$ and $w = -5:0.1:5$. In a single figure, using `subplot()` commands to get a 2x2 grid of subplots, plot the real part, imaginary part, absolute value and phase of the computed FT as function of w .

>> For phase use the command `angle()` in matlab. Can you explain each of the observed subplots?

(c) Repeat part (b) for $T = 1$ and $T = 4$. Use $w = -5:0.1:5$. What FT property supports your observations when T is changed?

(d) Repeat part (b) for $x(t) = e^{jt}$, and $x(t) = \cos(t)$. Limit signals to the interval $[-T, T]$ where $T = \pi$ and $w = -5:0.1:5$. What is the expected FT? What are the shapes you are observing?

(e) Repeat part (b) for a triangle pulse of height 1 and base/support $[-1, 1]$. How would you express xt for this case?

What is the expected FT? Hint: express the triangle pulse as convolution of two signals.

(f) Optional: play with some more signals $x(t)$ to test your function and verify whether standard properties of FT are satisfied as expected.

8.2 Fast Fourier Transform (Radix-2)

Write a matlab function `radix2fft` which takes as input an N-length vector `x` and returns an N-length vector `X`. To compute `X`, implement the decimation-in-time radix-2 FFT algorithm for an input vector whose length is a power of 2 i.e. assume that $N = 2^m$. Note that this function will have a recursive structure. Specifically, `radix2fft` is called within itself until it reaches the stopping criteria of $N = 2$. What is the DFT when $N = 2$? Verify that the output of your function matches with that of `fft` within numerical precision.

8.3 Quantization

Quantization involves discretization and encoding of samples of a signal. In this task we will take a closer look at discretization and how it effects the signal values. The encoding part which maps these levels to appropriately chosen binary code words is not considered here. For a discrete-time signal $x[n]$, quantization and quantization error are given by

$$x_q[n] = Q(x[n])$$

$$e_q[n] = x[n] - x_q[n].$$

The quantization function $Q(\cdot)$ maps any real input to a point from discrete set of values. There are numerous quantization functions used in practice. We implement a non-uniform quantizer in the function below.

>> Write a Matlab function `y = quadratic_quant(x,B,a)` with inputs

- `x` – input signal as a vector
- `B` – number of bits used to decide quantization levels (positive integer)
- `a` – positive real number such that $[-a,a]$ forms the range for quantization.

The function should produce an output `y`, which is the quantized version of the input signal. For purpose of this function, assume that the input signal `x` itself is not quantized (though this is not true). We wish to implement a *quadratic non-uniform quantizer* as discussed below.

>> For the range $[0, a)$: To implement the above function, divide the interval $[0,1]$ into $L = 2^{B-1}$ equal sized intervals. Let $0 = r_0 < r_1 \dots < r_{L-1} < r_L = 1$ be edge points of these intervals. Then the quantizer should map values in the interval $[r_i, r_{i+1})$ to its mid-point. Repeat the process symmetrically for the range $[-a, 0)$. Make sure your quantizer has a total of 2^B levels in the interval $[-a, a)$.

As an example, if $a = 1$ and $B = 2$, then the $2^B = 4$ quantizer intervals are $[-1, -0.25)$; $[-0.25, 0)$; $[0, 0.25)$; $[0.25, 1)$ and the quantization follows if $x[i] \in [-1, -0.25)$, $y[i] = -0.625$, if $x[i] \in [0, 0.25)$, $y[i] = 0.125$ and so on.

This is only an example; the code should consider general inputs 'a' and B. For inputs outside the interval $[-a, a)$ you should quantize them to the end points of your quantized set of values.

>> Write a matlab script which does the following tasks:

- Consider the analog signal $x(t) = \sin(2\pi f_0 t)$ with $f_0 = 10 \text{ Hz}$. Sample a 1 second portion of this signal (in the time interval $t \in [0, 1]$) at a sampling frequency of $F_s = 5 \text{ KHz}$ to obtain the discrete-time signal $x[n]$. Use the function above to obtain the quantised signal $x_q[n]$.
- Create a figure with 3x1 subplot grid and plot the sampled signal and quantised signal in the first two subplots (use proper labelling). Use $a = 1$ and $B = 4$.
- Compute and plot the quantization error in the remaining panel of the figure.
- In a different figure, plot histogram of the quantization error using the matlab function `histogram()` with 15 bins (see documentation). Repeat for $B = 3$ and compare the histograms.
- Repeat above processing for $B = 1:8$ (do not repeat the figures). In another figure, plot a graph with B on X-axis and maximum absolute quantization error (over the complete signal duration) on Y-axis and comment on your observations.
- Experimentally measured SQNR is defined as the ratio of signal power to quantization noise power:

$$SQNR = \frac{\sum_n |x[n]|^2}{\sum_n |e_q[n]|^2}$$

In another figure, plot a graph with B on X-axis and Signal to Quantization Noise Ratio (SQNR) on Y-axis and comment on your observations.

- What can you say about accuracy of the above non-uniform quantizer in various regions of the interval $[-a, a)$? How would this compare with say a uniform quantizer which instead considers intervals of the form $[ar_i, ar_{i+1})$?

8.4 Quantization of Audio signals

For this section, use one of the audio file from (previous lab) and write a matlab script.

>> Load .wav file in Matlab. Quantize this signal using your quantization function with $B = 3$

and $a = 1$. Listen to the original signal and the quantized signal using the `sound()` command. How does the sound quality of these two signals compare?

>> Perform quantization for different levels ($B = 1:8$) in a for-loop. Play the quantized signal in the for-loop with a pause of 2 seconds added after every call to the `sound()` command. Note your observations as levels increases and comment on quality of sound by hearing them.

>> How does quantization affect the frequency content of the quantized signal compared to that of the input signal? How does B play a role in this?