Discrete Fourier Transform

442.003 Digital Signal Processing, Laboratory
Winter Term 2023/24
Signal Processing and Speech Communication Laboratory
www.spsc.tugraz.at

Last updated: October 17, 2023

Abstract

This note provides a brief review of the Fourier transform for the analysis of discrete-time signals and systems and a description of practical assignments, which will be performed on a Raspberry Pi. Some tasks are to be performed with the help of MATLAB. It is assumed that the student has enough theoretical background to perform the practical part of the work. The review of some aspects of discrete-time signals and systems provided here is just a quick reminder.

Equipment:

- PC with netbeans IDE & MATLAB installed
- Raspberry Pi
- Oscilloscope Agilent 54622D
- Signal generator Agilent 33120A
- Cables

1 Introduction

In signal processing applications we often face the necessity to combine and separate signals. Sometimes an oscilloscope will clearly show what happens. When you mix signals, however, the oscilloscope is not very useful in many situations, since the sum of signals is often a messy thing to look at.

The idea behind the Fourier transform is pretty simple: one can think of signals as of a sum of sinusoids with different frequencies and phase shifts. This approach can be applied to a variety of signals (see table 1), including continuous-time, discrete-time, periodic, aperiodic, and even N-dimensional signals.

Signal type / Time domain	*Continuous-time*	*Discrete-time*
Periodic signals	**Fourier series**	**Discrete Fourier transform**
Aperiodic signals	**Fourier transform**	**Discrete-time Fourier transform**

Table 1: Fourier analysis framework

2 Overview of Fourier transform family

2.1 Fourier series (FS)

Fourier series are used for periodic continuous-time (analog) signals $x(t)$ with period T_0. There are several ways to write this transform, but the most common is the two-sided, complex exponential representation:

$$X[k] = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t)\, e^{-j\frac{2\pi k}{T_0}t}\, dt, \qquad k = -\infty \ldots \infty.$$

The time signal $x(t)$ can be computed from the Fourier coefficients $X[k]$ by

$$x(t) = \sum_{k=-\infty}^{\infty} X[k]\, e^{jk\omega_0 t} \quad \text{(complex form)},$$

or by

$$x(t) = \frac{A_0}{2} + \sum_{k=1}^{\infty} A_k \cos(k\omega_0 t + \varphi_k) \quad \text{(trigonometric form)},$$

$$\text{using} \quad A_k = 2\,|X[k]|, \quad \text{and} \quad \varphi_k = \arg(X[k]) = \arctan\left(\frac{\Im m\{X[k]\}}{\Re e\{X[k]\}}\right),$$

$$\text{and} \quad \omega_0 = 2\pi/T_0 \quad \text{the fundamental angular frequency.}$$

According to these equations we can represent a periodic signal $x(t)$ by a series of coefficients $X[k]$, or A_k and φ_k. In the trigonometric form A_k and φ_k are the amplitude and the phase of the harmonic signal component with angular frequency $\omega = k\omega_0$. In the complex form both amplitude and phase of the complex harmonic signal component[1] $e^{jk\omega_0 t}$ with angular frequency $\omega = k\omega_0$ are represented by the complex coefficient $X[k]$.

The Fourier series coefficient for $k = 0$, $X[0]$ resp. A_0 represents the mean value of $x(t)$, i.e., the DC component in the signal (a 'harmonic' signal at frequency $\omega = 0$). *Question: What is the phase* $\arg(X[0])$ *resp.* φ_0 *of the DC component?* A_1 resp. $X[1]$ and $X[-1]$ represent harmonic components at the fundamental angular frequency ω_0. Coefficients for $|k| > 1$ represent higher harmonics of the signal $x(t)$. *Questions: What is the relation between* $X[k]$ *and* $X[-k]$*? Why do we sum over* $k = -\infty \ldots \infty$ *in the complex form of the Fourier series representation but only over* $k = 1 \ldots \infty$ *in the trigonometric form?*

[1] Recall that $e^{jk\omega_0 t} = \cos(k\omega_0 t) + j\sin(k\omega_0 t)$.

2.2 Fourier transform (FT)

The FT is used for aperiodic continuous-time (analog) signals $x(t)$. The most common way to express this transform is the two-sided, complex exponential representation:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t} dt.$$

The inverse transform to find the time signal $x(t)$ again, is

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)\, e^{jwt} d\omega.$$

According to these equations any time signal $x(t)$ can be represented by – and perfectly re-constructed from – the function $X(\omega)$ in the spectral domain. $X(\omega)$ is called the *Fourier spectrum* of the signal $x(t)$. The Fourier spectrum is a continuous function of angular frequency $\omega \in [-\infty, \infty]$.

- To calculate the exact FT, you need to know $x(t)$ from $t = -\infty$ to $t = +\infty$. This is of course impossible for real-world applications.
- In most cases of practical interest, $X(\omega)$ will not be periodic (cf. discrete-time Fourier transform, below).

2.3 Discrete-time Fourier transform (DTFT)

The DTFT is used for aperiodic, discrete-time or digital[2] signals $x[n]$. The most common way to express this transform is

$$X(e^{j\theta}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\theta n}.$$

Here, θ is the *normalized angular frequency*, $\theta = 2\pi f/f_s$, with f_s being the sampling rate of $x[n]$. Since the term $e^{-j\theta n}, \forall n$ in the transform is periodic in θ, with period 2π, also the resulting spectrum $X(e^{j\theta})$ is periodic (with period 2π, too). To indicate this periodicity, the argument of the spectrum is commonly written as $e^{j\theta}$ (not θ only). Due to the periodicity, the Fourier spectrum of a discrete-time signal has to be known only for an interval of length 2π, commonly the fundamental interval $\theta \in [-\pi, \pi]$ is used to represent $X(e^{j\theta})$.

The discrete-time signal $x[n]$ can be recovered by

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta})\, e^{j\theta n} d\theta.$$

- This expression says that we can use the function $X(e^{j\theta})$ (Fourier spectrum) to represent the discrete-time signal $x[n]$.
- The spectrum $X(e^{j\theta})$ is a continuous function in θ.
- The DTFT spectrum is *always* periodic with the period of 2π.

[2] A digital signal, as used in computers and DSPs, is a *quantized* discrete-time signal.

2.4 Discrete Fourier Transform (DFT)

The DFT is used for periodic, discrete-time or digital signals $x[n]$. The DFT for a signal with period N is

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi k}{N}n}.$$

The inverse transform (IDFT) is

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi k}{N}n}.$$

These equations say that the coefficients $X[k]$ represent the periodic discrete-time signal $x[n]$. Notice, that only N samples of the time signal $x[n]$ are used to compute $X[k]$ (for $k = -\infty \ldots \infty$), and also only N of the coefficients $X[k]$ are used in the inverse transform.

- A periodic discrete-time signal has a periodic spectrum (just like any other discrete time signal), i.e., $X[k+N] = X[k]$.
- The $X[k]$ terms are evenly spaced samples (at $\theta = k\frac{2\pi}{N}$) of the continuous spectrum $X_1(e^{j\theta})$ of the signal $x_1[n] = x[n] \quad \text{for} \quad 0 \leq n < N; \quad x_1[n] = 0 \quad \text{for} \quad n < 0, n \geq N$ (*Verify this using the formula for the DTFT above!*).

Fast Fourier transform (FFT)

To represent the time signal we need to compute $X[k]$ for N values of k, and for each of these values we must perform N multiplications and $N-1$ additions, so computing the DFT requires $N(2N-1)$ arithmetic operations, i.e., it has a *computational complexity* of $\mathcal{O}(N^2)$ (order of N^2). However, specifically if N is a power of 2, many of the DFT calculations are redundant. By carefully re-arranging the order of multiplications and additions, the computational complexity can be reduced to $\mathcal{O}(N \log_2(N))$. The resulting algorithm is called *fast Fourier transform*. The difference between $\mathcal{O}(N^2)$ and $\mathcal{O}(N \log_2(N))$ can be substantial: If $N = 1024$, and $\log_2(1024) = 10$, the computational complexity of the FFT is only 1% of that of a DFT.

3 Window functions

As noted above the DFT and IDFT are defined for periodic time signals. We may, however, be interested in a digital representation of the spectrum of non-periodic discrete-time signals. For time-limited signals, i.e., for signals that differ from zero only for $0 \leq n < N$, we found that the DFT results in samples of the DTFT. For non-periodic, infinite (or long) time signals we have to restrict the calculation of the DFT to a number of N samples, meaning that we may only use a number of $L \leq N$ signal samples. The procedure to pick only a limited number of samples from a possibly infinitely long signal is called *windowing*.

If we bear in mind, that we now calculate a local (short-term) spectral representation, and that the IDFT will result in a periodic signal, the DFT such can be applied for all discrete-time signals.

3.1 Rectangular window

The simplest way to restrict an infinite time signal to a length of L samples is to multiply the signal with a window function

$$w[n] = \begin{cases} 1 & 0 \leq n < L, \\ 0 & \text{otherwise.} \end{cases}$$

This window function is called *rectangular window* (for obvious reasons).

Considering that we multiply two signals $x[n]$ and $w[n]$ in the time domain, the corresponding operation in the frequency domain is a (circular) convolution of the spectra $X(e^{j\theta})$ and $W(e^{j\theta})$. E.g., the spectral line $X(e^{j\theta}) = 2\pi\delta_{2\pi}(\theta - \theta_0)$ of a harmonic signal $x[n] = e^{j\theta_0 n}$ is transfered to the spectrum of the window function shifted by θ_0 after windowing: $e^{j\theta_0 n}w[n] \circ\!\!-\!\!\bullet 2\pi W(e^{j(\theta-\theta_0)})$. To investigate the effects of windowing we thus should also look at the properties of the Fourier transform of a window function.

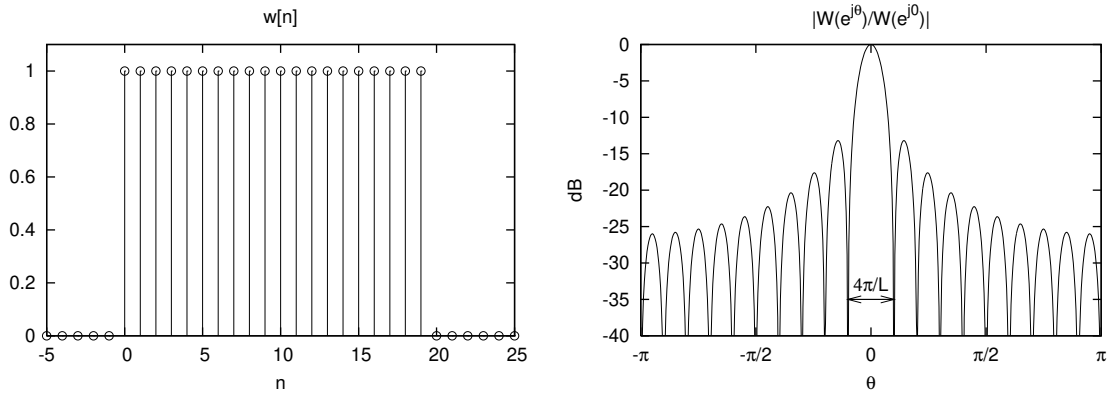

Figure 1: Rectangular window ($L = 20$) and corresponding spectrum

As depicted in fig. 1 the rectangular window results in a good frequency resolution due to the narrow main-lobe. However, the amplitude measurement capabilities are rather poor, due to the high amplitudes of the side-lobes. Except for self-windowing functions (impulses, bursts, etc.) the effects of spectral leakage (transfer of spectral energy due to the side-lobes of the window function Fourier transform) are significant.

The effect of introducing spectral energy at frequencies that are not present in the original signals spectrum by windowing can also be explained in the time domain: By windowing we introduce discontinuities in the periodic signal represented by the DFT coefficients in general. To reduce spectral leakage window functions that decay in amplitude towards the begin and end of the window are used.

3.2 Triangular (Bartlett) window

A simple window function that decays towards the ends is the triangular or Bartlett window:

$$w[n] = \begin{cases} 1 - \dfrac{|n - (L-1)/2|}{(L-1)/2} & 0 \leq n < L, \\ 0 & \text{otherwise.} \end{cases}$$

An example of a triangular window and its spectrum is shown in fig. 2. Notice the lower amplitude of the side-lobes in the spectrum as compared to that of the rectangular window (different scale!). For the triangular window of (approximately) the same length, however, we find a considerably wider main-lobe as for the rectangular window.

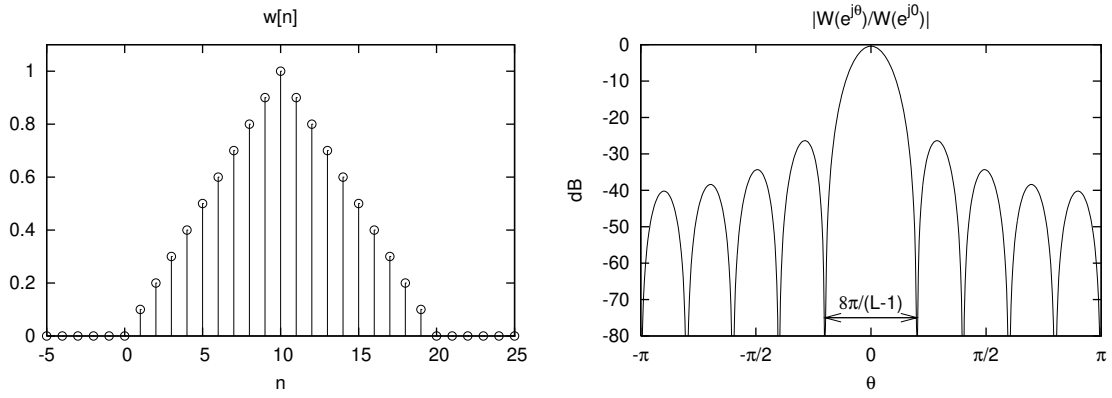

Figure 2: Triangular (Bartlett) window ($L = 21$) and corresponding spectrum

3.3 Generalized cosine windows

The Blackman, Hamming, Hann (aka. Hanning), and rectangular windows are all special cases of the generalized cosine window:

$$w[n] = \begin{cases} A - B\cos(2\pi\frac{n}{L-1}) + C\cos(4\pi\frac{n}{L-1}) & 0 \leq n < L, \\ 0 & \text{otherwise.} \end{cases}$$

where A, B, and C are constants. The window choice is usually task-dependent: the width of the the main-lobe is inversely proportional to height of the side-lobes. Thus, there is a trade-off between frequency resolution and reduction of spectral leakage. The Hamming and Hann windows (fig. 3) are the two-term generalized cosine windows, given by the $A = 0.54$, $B = 0.46$ for Hamming and $A = 0.5$, $B = 0.5$ for Hann windows ($C = 0$ in both cases). The Blackman window is a popular three-term window, given by the $A = 0.42$, $B = 0.5$, $C = 0.08$.

The flat-top window that can be selected for the FFT analysis in the oscilloscope Agilent 54622D is a generalized cosine window with a sum over 5 terms, and $A = 1, B = 1.93, C = 1.29, D = -0.388$, and $E = 0.322$.

3.4 Kaiser window

The Kaiser window is an approximation to the prolate-spheroidal window, for which the ratio of the main-lobe energy to the side-lobe energy is maximized. For the Kaiser window of the particular length L, the parameter α controls the side-lobe height. For the given α, the side-lobe height is fixed with respect to the window length.

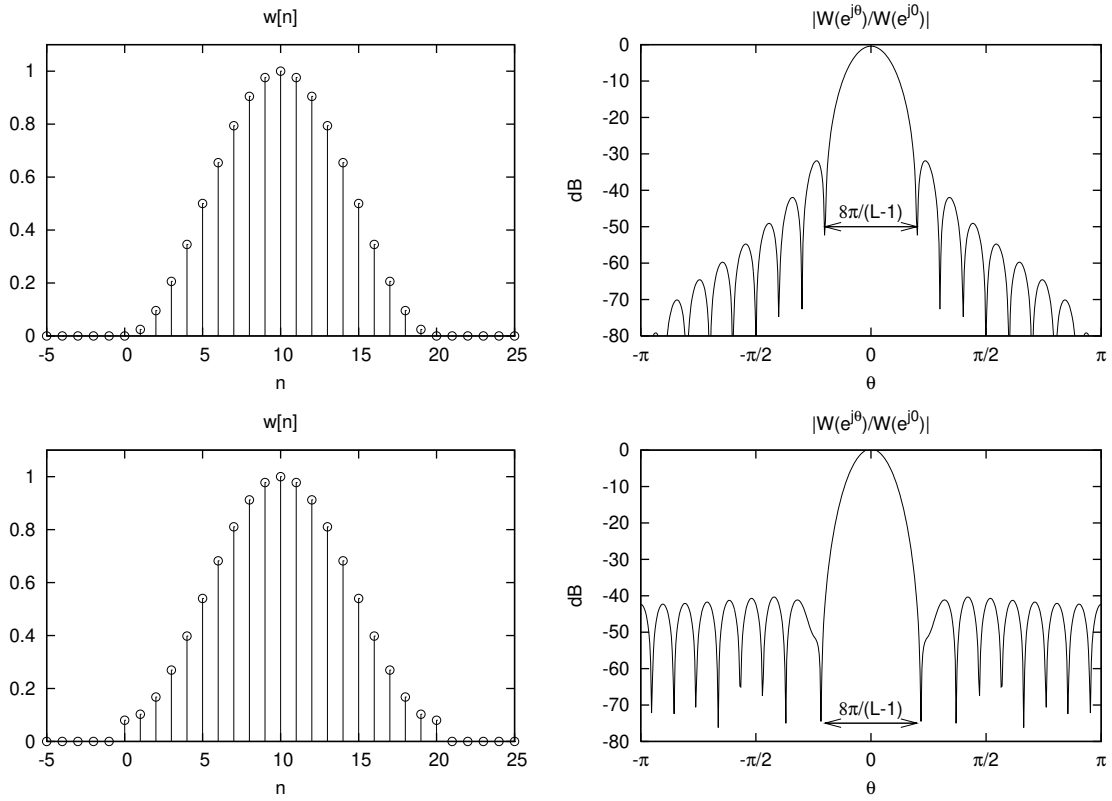

Figure 3: Hann and Hamming windows ($L = 21$) and their spectra

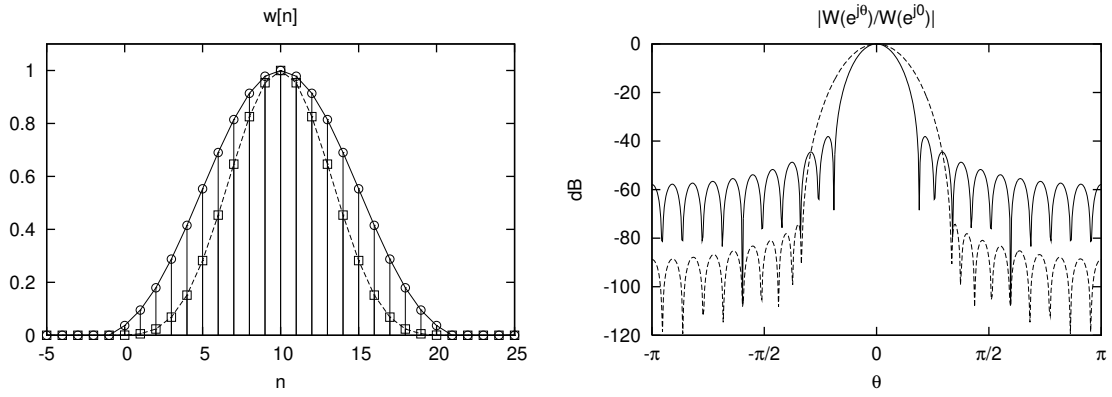

Figure 4: Kaiser windows and spectra for $L = 21$ and $\alpha = 5$ (circles and solid lines), and $\alpha = 10$ (squares and dashed lines)

$$w[n] = \begin{cases} \dfrac{I_0\left(2\alpha\sqrt{\dfrac{n}{L-1} - \left(\dfrac{n}{L-1}\right)^2}\right)}{I_0(\alpha)} & 0 \le n < L, \\ 0 & \text{otherwise.} \end{cases}$$

$I_0(\cdot)$ is a 'modified Bessel functions of the first kind'.

3.5 Chebyshev window (also Dolph-Chebyshev)

The Chebyshev window minimizes the main-lobe width, given a particular side-lobe height. It is characterized by an equi-ripple behavior, that is, its side-lobes have the same height. Examples are given in fig. 5.

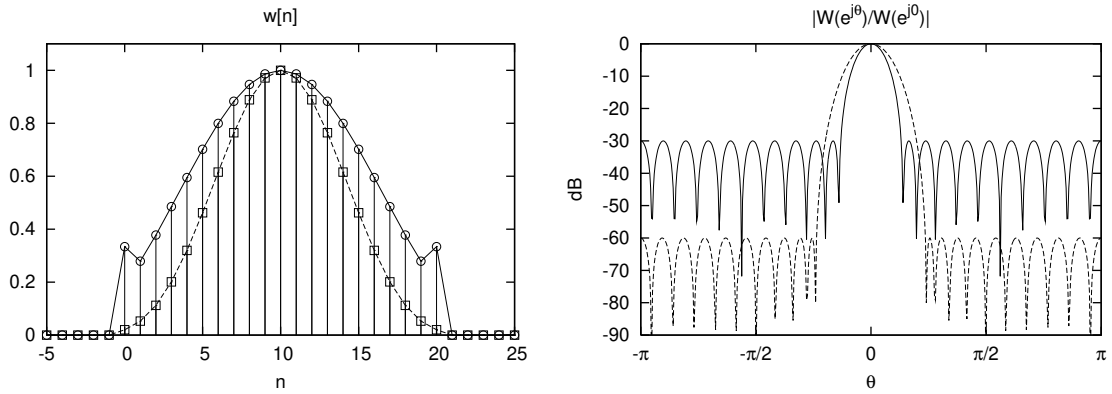

Figure 5: Chebyshev windows and their spectra ($L = 21$) designed for 30dB (circles and solid lines) and 60dB (squares and dashed lines) side-lobe suppression

3.6 Windows usable on the oscilloscope

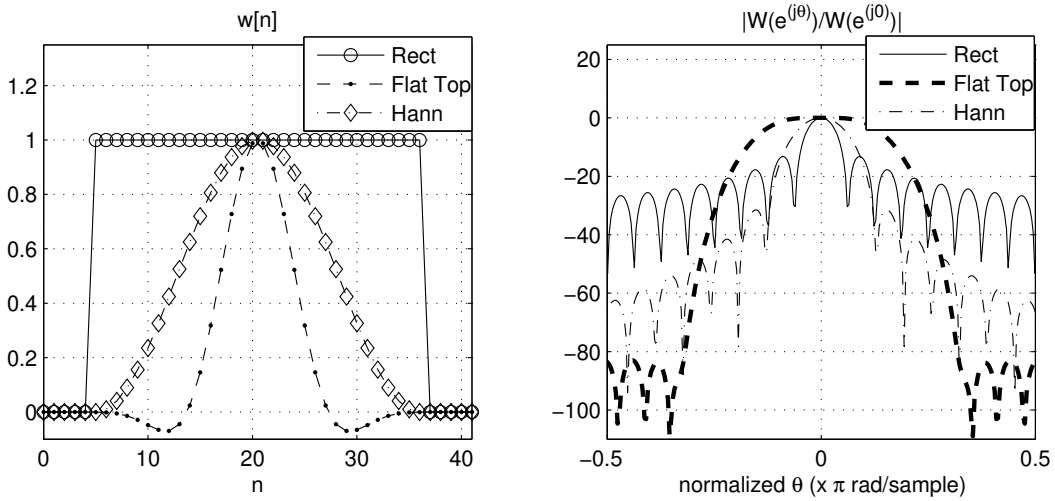

Figure 6: Windows and their spectra on the oscilloscope in the laboratory (Agilent 54622D).

In the practical part, we will make use of the FFT capabilities of the Agilent 54622D oscilloscope. On this device, you can select three different window functions: Rectangular, flat-top and Hann-window. Figure 6 shows a comparison of those windows in time and frequency domain. *Think about application scenarios for each of these three windows!*

In Matlab, you can also use `wintool` to comfortably create and compare different window functions.

4 Applications of the FFT

4.1 Estimation of the power spectral density

A very common task in signal processing is the estimation of the power spectral density (PSD) of a signal $x[n]$ [1]. A first step would be to compute the normalized squared magnitude of the signal's DFT:

$$I[k] = \frac{1}{LU}|X_w[k]|^2$$

where L is the length of the window, U is a normalization constant that depends on the chosen window and $X_w[k]$ is the k-th coefficient of the DFT of the windowed signal $x[n]w[n]$. This estimate of the PSD is called the *periodogram* if a rectangular window is used, otherwise it is called the *modified periodogram*.

Another method to estimate the PSD of $x[n]$ is to compute the average of R periodograms:

$$I[k] = \frac{1}{R}\sum_{r=0}^{R-1} I_r[k], \qquad k = 0, \ldots, N-1$$

This means that the individual $I_r[k]$ are periodograms of blocks of the signal $x[n]$. Depending on the application, these blocks can have overlap or not. The choice of the window function $w[n]$ of course also influences the PSD estimation. If $w[n]$ is the rectangular window then the method of periodogram averaging is called *Bartlett's procedure*. In case of the other window shapes it is called *Welch's procedure*. *Think about advantages and disadvantages of the two methods!*

4.2 Fast convolution

One of the first applications of the FFT algorithm was to implement convolution by multiplication in the frequency domain – faster than by evaluating the convolution in the time domain directly. The linear convolution is defined as

$$y[n] = \sum_{i=0}^{L_h-1} h[i]x[n-i],$$

where $x[n]$ is a length-L_x input signal, $h[n]$ is a length-L_h impulse response of the filter, and $y[n]$ is the output signal. Examination of this equation shows that the output signal $y[n]$ has length $L_y = (L_x + L_h - 1)$, and computational complexity is of order $\mathcal{O}({L_y}^2)$.

The way to compute a fast convolution is to calculate the product $Y[k] = X[k]H[k]$ of FFTs of the time signals $x[n]$ and $h[n]$ and to use an inverse FFT to compute the output signal $y[n]$. For large L_x and/or L_h we profit from the reduced computational complexity of the FFT of $\mathcal{O}(N \log_2(N))$. However, remember that since the DFT – and FFT – represent periodic time signals the direct application of this approach would result in a cyclic convolution, which is often undesirable. The solution is to pad both signals with appropriate number of zeros before computing the FFTs.

5 Practical part

The practical part of the work consists of several tasks that should be performed using the measurement equipment available and the RPi. Detailed explanation of how and what tasks should be accomplished is found below.

Experiment 1:
Fourier analysis of periodic signals

The first task of the laboratory will be accomplished without using the RPi.

1. Switch on signal generator and oscilloscope. Set the signal generator to produce a square waveform with amplitude of 1 V and a frequency of 500 Hz. Connect the signal generator's output to the oscilloscope.

2. Knowing the amplitude and the period of the oscillations, compute (analytically) the coefficients A_k of the Fourier series of the input signal for $k \leq 8$.

3. On the oscilloscope push **Math**→***FFT***→***Settings***. Set ***Source*** to the channel that corresponds to the output of the signal generator. The frequency resolution can be adjusted by selecting the time-base of the input signal. Set an appropriate value. (Try to reach ***SPAN*** = 10 kHz and ***Center*** = 5 kHz.) Select ***More FFT*** and set ***Window***=FLAT TOP. Set up other options (***Scale*** and ***Offset***) to convenient values.

4. Measure the amplitudes of the first 8 harmonic components of the corresponding Fourier representation. Determine the error between the theoretical and the measured values. What is shown on the scope, which information is lost?
   Note: The FFT is plotted in the dB domain. 0 dB corresponds to 1 V RMS amplitude.
   Produce and fill in a table like the following:

Harmonic number	1	2	3	4	5	6	7	8
Frequency (Hz)								
Theoretical amplitude A_t (V)								
Measured amplitude A_m (V)								
Error $E = A_t - A_m$ (V)								

Experiment 2:
Time- and frequency-domain properties of different windows

In this task, MATLAB is used to demonstrate basic properties of windowing functions.

1. Start MATLAB and load the script file `fftdemo.m`. Set a breakpoint in the first line of the program, run it, and discuss it using the single-step debug-mode. Change the frequency from $0.02 f_s$ to $(16/512) f_s$ and discuss the results. In line 20, other window functions can be implemented and discussed, if there is time left.

2. Apply a sine-wave to the oscilloscope, view its FFT spectrum and adjust the parameters of the oscilloscope in order to have the same ratio of the signal- and the FFT-sampling frequencies as in the previous task. Use the rectangular window! Discuss the result and compare it to the MATLAB-figures. To further investigate what happens in the oscilloscope, go step-by-step through the MATLAB script `fft_hp_scope.m`.

3. Set the waveform of the signal generator to sine, the amplitude to 1.414 V (corresponding to 0 dBV RMS on the scope), and the frequency to 1 kHz. On the oscilloscope, set the time-base to $10\frac{\text{msec}}{\text{div}}$ to get a frequency-resolution of 10 Hz. Select the rectangular window. Change the frequency between 1 kHz and 1.02 kHz in steps of 1 Hz and measure the amplitudes. Plot the measured amplitudes vs. frequency and discuss the result! Repeat for the other windows and identify which window provides the most accurate measurement of the amplitude.
   Note: To make your measurements more accurate, change the FFT scaling factor to $1\frac{\text{dB}}{\text{div}}$ and also make use of cursors and **Quick Meas** capabilities

4. Generate an AM modulated sine-wave using the signal generator (Press **Shift** + **AM** to activate and de-activate AM). Set the carrier frequency (**Freq**) to 1 kHz, the amplitude to 0.5 Vpp, and the waveform to sine. For the modulating signal set the frequency to 100 Hz (**Shift** + **Freq**), modulation depth to 10 % (**Shift** + **Level**), and the waveform to sine (is default). Discuss how the spectrum of the resulting signal should look like!

5. On the oscilloscope, activate the input signal and set the time-base of the input signal to $2\frac{\text{msec}}{\text{div}}$. As the result you will see the modulated sinusoid with two periods of the modulated waveform. Set the trigger to the external signal, which will cause a stable picture. (Set ***Mode*** to ***Normal***, if you don't obtain the stable picture.)

6. Press **Math**→***FFT*** →***Settings***. Make sure that ***Source*** is set to the correct channel (the output of the signal generator). Set ***Span***= 2 kHz and ***Center***= 1 kHz. Press ***More FFT*** and select ***Window***="Rectangular". What can be seen? What is the frequency-resolution of the FFT used in the oscilloscope? (Consider the window-duration, which corresponds to the visible portion of the time-domain signal.)

7. Using the cursors, mark the frequency of the carrier and the frequency of the upper side-band at 1.1 kHz. Can you distinguish the carrier and the sideband signals? How can the frequency-resolution be enhanced? Which frequency-resolution is required to distinguish the signals, without taking spectral leakage into account? Explain your answer.

8. Answer the following questions: How are the sampling period T_s, sampling frequency f_s, the time-duration of the FFT-window T_w, and the frequency-spacing Δf (frequency-resolution) between FFT-lines related to one another? Use the parameter N to specify the number of FFT-points used.

9. Change the time-base to increase the spectral-resolution. Denote for which resolution the sidebands become visible first.

Experiment 3:
PSD estimation

Implementation of the periodogram and Welch's method on the RPi.

1. Set up the signal generator to produce a 500 Hz sawtooth oscillation with 0.5 Vpp (peak to peak) amplitude. Connect the RPi to the signal generator and the oscilloscope. On the oscilloscope press **Math** and enable FFT function following the known procedure.

2. In netbeans, load the project `Exp3` and open `blockprocessing.h`. Look for the constant `FRAMELENGTH` which is identical to the chosen FFT size (DFT length) N.

3. Make sure that your code uses an FFT size of 128 samples. Do not run the program yet.

4. Open `Fouriertransform.h`. Scroll this file to the beginning until you see the definitions of two important parameters: `const int AVERAGING` is used to specify the number of periodograms to average and `const float ADDNOISE` is used to add a certain amount of noise (simulated within the program) to the input signal to test for the robustness of the estimator. Make sure that `AVERAGING` is 1 (such that no averaging is performed) and that no noise is added. If necessary, rebuild the project.

5. Explore the power spectral density using Matlab. Copy the periodogram values from the command window to Matlab and plot it. Note which harmonics can be identified (the sampling frequency is 32 kHz, you can click in the graph area and see the sample index on the lower left). For visualization purposes, often the decibel scale (`10*log10()`) is chosen.

6. Add some noise by increasing `ADDNOISE` to a value of 0.005 and rebuilding the program. Again note which harmonics can be identified. Now increase the FFT-size (equivalent to the buffer size FRAMELENGTH) to 256 and 512. Describe the effect of the longer FFT on the PSD estimate!

7. Now use periodogram averaging by setting `AVERAGING` to 10 and with the FFT-size again reduced to 128. Rebuild and run the program. Note again which harmonics can be identified. Also try the longer FFT sizes with averaging enabled and also higher numbers for the number of periodograms to average. What are the advantages and disadvantages of the Welch method?

8. For your report, capture some of the outputs for the PSD estimation that you find to be explanatory.

A Credits

This document was authored and/or adapted by Dmitriy Shutin, Klaus Witrisal, Erhard Rank, Marián Képesi, Paul Meissner, Christian Knoll and Josef Kulmer.

References

[1] Oppenheim, A. V., Schafer, R. W., and Buck, J. R., "Discrete-Time Signal Processing", 2nd edition, Prentice-Hall, Inc., 1999.

[2] Feldbauer, C.: "Real-Time Block Processing Environment," Laboratory Handout, http://www.spsc.tugraz.at/courses/dsplab/intro/blockprocessing.pdf, 2005.