**Laboratory 2B:**

Fourier Series and Line Spectrum

In this laboratory you practice with varied exercises the concept of the Fourier series and the corresponding line spectrum, which describes periodic signals in the frequency domain. The exercises cover different aspects as: analytical calculation of Fourier coefficients (on paper), plots and checks with Fourier synthesis in Matlab, measurements and numerical approximation with the FFT function.

# Fourier Series Coefficients: calculate and plot

1. Calculate the complex Fourier coefficients of an odd periodic square with amplitude varying between +A and -A, period T0 and duty cycle 50%.

Please use the ck integral definition for your calculation:



Check what are the numerical values of ck for k within [-5 ; +5] .

Do the values of ck depend on T0 ?

Is it possible to have a simpler expression for all odd, and all even ck coefficients?

Are the non-zero ck coefficients pure real or pure complex numbers? What does this mean compared to the real coefficients ak and bk.

1. Check the values of your ck coefficient with a Fourier synthesis in Matlab.   
     
   Hint: You can use the Matlab code example in List-2 exercise-7 .
2. Prepare a plot of the corresponding double sided and single sided spectrum in Matlab. Use frequency in Hz as unit for the horizontal axis.

Hint: you can define your f vector as f = k\*f0 , where k is the index vector for the ck coefficients.

# Numerical Approximation for Fourier Series with FFT

1. Start a new script in Matlab, and describe a square function as in exercise 1a above using the square() function. Before you start your code read the item (b) below.
2. Adapt the length of the time vector so, that you show exactly one period of the square function, and then calculate the numerical approximation of the ck coefficients using the fft() function. A code extract is given next to help you out:

N = 2^10; % number of points for the FFT

aux = 0:1:N-1; % auxiliary index vector

tstep = 1\*T0/N; % resolution in the time domain

t = tstep\*aux; % time vector

fstep = (1/tstep)/N; % resolution in the frequency domain

f = fstep\*aux; % frequency vector

x\_t = … % here you define your time function

X\_k = (1/N)\*fft(x\_t); % normalised FFT = approx(ck coefficients)

% For the plot taking only DC plus the first 19 harmonics (c0 bis c19)

ck = X\_k(1:20);

% Conversion in case you want to compare to a single sided spectrum

Ak = 2\*abs(ck); % A1-A19 = 2\*abs(c1-c19) but

Ak(1) = Ak(1)/2; % A0 = c0 => do not double DC-component

phik = phase(ck);

1. You can now vary your time function to have different duty cycles (e.g. 50%, 25%, 20%, 10%), and also change to a ramp (use in Matlab the function sawtooth() ) . which changes can you observe in the spectrum?
2. Let us observe now the measurement of an amplitude spectrum with the FFT function of the oscilloscope. Start by measuring the spectrum of a sine function with 2Vpp and f0=1kHz. Generate your sine signal with the Function Generator (FuGe). Which curve do you expect to see in this measurement?
3. Change now the FuGe settings to generate a periodic square with 50% duty cycle (as in your Matlab code of part-a ). Adjust the horizontal scale such that you can observe at least the first 20 harmonics (and maximum 50 harmonics).

Can you imagine why the spectrum displayed by the oscilloscope vary significantly (specially the “noise” part) when you change the horizontal scale? Change the window type from rectangular to Hanning, which changes do you observe in the spectrum?

1. Open, execute and read the Matlab script *sisy\_fft\_ideal\_vs\_non\_ideal.m* . Which of these non-ideal effects do you think can have an influence on your measurement of part (e) ?

In Lab2C we will continue to investigate the FFT, and its usage in non-ideal situations  
(practical cases).