LULEÅ UNIVERSITY OF TECHNOLOGY
A. Enmark / 2016

F7003R Optics and Radar Based Observations, 7.5 ECTS

Problems Part 1 (4 points + 2 bonus points)
Signal Processing for radar applications

Goal

On successful completion of the assignment the student shall

- know some important functions for radar applications (rectangular pulse, harmonic function, delta-function, constant, sinc) and their Fourier transforms
- know some important transform properties
- know the impact of sampling (aliasing, time resolution) and truncation (leakage, frequency resolution) on the DFT output
- be able to use MATLAB for basic signal processing computations

Content

The assignment consists of two parts:
Part A: MATLAB tutorial (0 points)
Part B: Signal processing for radar (4points + 2 bonus points)

Students *shall* answer Part B questions **1-12** to be approved (4 points)
Students may attempt to answer Part B question **13,** but this is *not obligatory* (2 bonus points)

If you are unfamiliar with MATLAB start to go through Part A (MATLAB tutorial) as soon as possible.
You are also recommended to do the preparations part for Part B as soon as possible, Fourier transforms will be used extensively in the course.

Part A: MATLAB Tutorial

The aim of Part A is to get familiar with MATLAB. MATLAB is a software package that is extensively used within the space community for system simulation, control applications, data processing, signal processing etc. MATLAB will be used in assignments during this course, and if you are not already familiar with the tool, you need to go through a basic tutorial

MATLAB is a trademark of The MathWorks, Inc. and links to MATLAB tutorials are found on their webpage:

http://www.mathworks.se/academia/student_center/tutorials/launchpad.html

The pdf from University of Dundee is recommended, but you may choose another tutorial. The Dundee tutorial is found on:

http://www.maths.dundee.ac.uk/ftp/na-reports/MatlabNotes.pdf

Part B: Signal Processing (total 4 points + 2 bonus points)

You are going to investigate the Fourier transform, the discrete Fourier transform and detection in noise, using small MATLAB examples.

The Fourier transform is one of the basic mathematical tools used in radar signals and systems design, analysis and signal processing. Many processing algorithms are operations performed in the frequency domain, using the Fourier transform of a sampled signal, instead of the time domain signal.

Parts of receivers and detectors are often implemented as digital systems. It is therefore important that you are familiar with the Fourier transform, including the discrete Fourier transform. The code given in some of the examples in this assignment is actually very similar to the code of many real radar systems, even if they are implemented in systems with better performance than a desktop computer with MATLAB.

The assignment does not present a thorough mathematical background in Fourier transforms (it is assumed that you have a basic course in Fourier transform from your bachelor degree). It will instead introduce some important issues (such as aliasing, leakage, phase wrapping) that may be present in discrete data sets.

When you are implementing a discrete time radar receiver in real life, you will be working with sampled, truncated and quantized *number sequences*. In the small examples in this assignment we have disregarded the quantization.

Task

Go through the preparation part as soon as possible. Then go through the tutorial part. It is a step by step instruction "***tutorial***" with some comments and ***questions***. The small code examples may be used as command line commands, but using scripts (.m-files) is recommended. ***Answer the questions and discuss the results.*** Hand in the answers and discussions. Questions and discussion topics that shall be reported in the hand in of Problem 1 are marked with

bold, red, italic

The obligatory parts render 4 points and the optional part render 2 bonus points

1 Preparations (0.5 point)

1a) Sketch the magnitude of the Fourier transform or Fourier series for the functions below (for real valued transforms you shall sketch the function itself, not only the magnitude).
1b) Discuss the character of spectrum of real valued even functions and functions that are not even

***Fourier transforms*:**
A constant (DC-level)

Dirac’s delta function
A harmonic function with amplitude A and frequency f and phase zero (cosine)
A harmonic function with amplitude A and frequency f and phase $\pi/2$ (sine)
A narrow rectangular function
A broad rectangular function
Sinc-function (broad and narrow)
White noise
A real valued function (general characteristics)
A real valued even function (in general).

Fourier series:
Rectangular periodic function
Triangular periodic function
General look of periodic function that is “softer” than the two functions above
Pulse train with large distance between pulses
Pulse train with small distance between pulses

2. Sampling (0.25 point)

We will start by studying harmonic functions, on the form $x(t) = A\cos(2\pi ft + \varphi)$, using MATLAB. For simplicity we use unit amplitude (A=1) and zero phase (φ=0) in our first examples. The frequency f is set to f=10Hz, giving $x(t) = \cos(2\pi 10t)$. In MATLAB code:

```
dt=0.01;          %Sampling interval 0.01 s
t=0:dt:1000*dt;
f=10;
xs=cos(2*pi*f*t);
```

The MATLAB vector **xs** is a discrete, sampled and truncated, version of the continuous function *x(t)* . In our case we have sampled *x(t)* for *t* =0 s, 0.01 s, 0.02 s,… 9.98 s,9.99 s.

Plot the function. Try to increase the sampling interval Δt to $\Delta t > 1/(2f)$ (i.e sampling frequency below the so called Nyqvist frequency) and plot the two functions in the same graph

```
dt1=0.01;         %Sampling interval 0.01 s
t1=0:dt1:2000*dt;
f=10;
xs1=cos(2*pi*f*t1);

dt2=99*dt;        %Sampling interval 0.99 s
t2=0:dt2:2000*dt;
xs2=cos(2*pi*f*t2);

dt3=199*dt;       %Sampling interval 1.99 s
t3=0:dt3:2000*dt;
xs3=cos(2*pi*f*t3);
```

```
close all
plot(t1,xs1);
hold on
plot(t2,xs2,'r');
plot(t3,xs3,'k');
```

You should label the axis and give the plot a title (these commands are not repeated in the rest of the examples)

```
legend('dt=0.01s','dt=0.99s', 'dt=1.99s')
xlabel('t [s]')
ylabel('x(t)')
title('Aliasing')
set(gcf,'Color','w')
```

2) Try to determine the two aliasing frequencies? How do they relate to the sampling frequency and the frequency of the harmonic functions? Try to set up an expression.

3. The Discrete Fourier Transform (DFT) (0 point)

The frequency content of a continuous function given as a closed expression may be determined by using the Fourier Transform. When we want to study the frequency content of a time series (sampled and truncated function) **`xs`**, we can use the discrete Fourier transform (DFT). This will give us a discrete approximation, **`Xs`** of the continuous Fourier transform, $X(u)=\mathcal{F}\{x(t)\}$, where u is the frequency and $\mathcal{F}\{\cdot\}$ denotes the Fourier transform. The sampling and the truncation will restrict the frequency resolution in the approximations and in some cases it will introduce some "artefacts" (compared to the continuous Fourier transform). In our first example the approximation parameters are chosen to minimize artefacts, but later in the tutorial we will investigate *leakage.*

The Fourier transform gives a double sided diagram, so if you have a harmonic function of only one frequency, the amplitude diagram will show two impulse functions, one on the positive side and one on the negative side of the frequency axis.. Depending on how the DFT is defined in the software package you are using, you will either get a double sided diagram with one "impulse" on the positive frequency axis combined with one on a negative frequency axis (like for the continuous transform), *or* a shifted version of this (like in MATLAB) where both "impulses" are on the positive part of the axis. If you want to present a single sided or symmetric double sided plot you must shift the function.

The time domain functions we are working with in this tutorial are real valued, but the DFT output will in general be complex valued. The time domain function is sampled with N samples, at t=0, $1\Delta t$ $2\Delta t$, $3\Delta t$,… (N-1) Δt and it is truncated to $T= N\cdot\Delta t$. For simplicity we assume N to be odd in the examples below. If N is even the scaling of axes must be changed accordingly. The DFT of a *real valued time series* will give out a *complex valued vector*, representing a sampled version of the DTFT, or may also be interpreted as an approximation of the Fourier transform $X(u)$ of the original function $x(t)$, sampled at the frequencies $u=0,\ \pm 1\cdot\Delta u,\ \pm 2\cdot\Delta u,\ \pm 3\cdot\Delta u,\ \pm 4\cdot\Delta u\ \ldots\ \pm\ (N/2)\cdot\Delta u,$

where N is the length of the vector **xs,** and $\Delta u=1/T$ is the frequency resolution . The values for negative frequencies are the complex conjugate of the values for positive frequencies (and are in MATLAB shifted to the positive axis), so for real valued signals all information is contained in half of the vector. Note that the sampled function and the DFT do not hold any information on the sampling interval or the frequency resolution; the DFT is an operation on a number sequence, giving back another number sequence. When interpreting the numbers the axis must be scaled accordingly, by the sampling interval:

```
dt=0.01;            %Sampling interval 0.01 s=> time
                    %resolution
t=0:dt:2000*dt;     %Prepare a scale for time axis
N=length(t);        %Number of samples
T=dt*N;             %Time duration of sampled and truncated
                    %function [s]
du=1/T;             %Frequency resolution [Hz]
u=0:du:(N-1)*du;    %Prepare a scale for frequency axis
```

4. The amplitude diagram (0.25 points)

For a complex number $z=\mathrm{Re}(z)+j\mathrm{Im}(z)$, the magnitude is $\sqrt{\mathrm{Re}(z)^2+\mathrm{Im}(z)^2}$, and the phase is $\arctan(\mathrm{Im}(z)/\mathrm{Re}(z))$. This means that we can get the amplitude and phase diagrams from the complex vector output of the DFT. The real and imaginary parts of a complex function can be calculated in MATLAB, using the `real` and `imag` operations, and to get the magnitude and phase `abs` and `angle` can be used.

The DFT is performed by the MATLABs `fft` (Fast Fourier Transform) command. The FFT is an efficient algorithm for calculation of the DFT.

When MATLABs `fft` command is used to approximate the Fourier transform, the result must be scaled with N to get the correct amplitude. For simplicity (to avoid so called leakage, see later in tutorial) we set the frequency of the harmonic function to an integer multiple of the frequency resolution

```
f=200*du;
xs=cos(2*pi*f*t);
Xs=fft(xs)/N;              %Do the FFT and scale
stem(u,abs(Xs));           %Show amplitude diagram
xlabel('u [Hz]')
ylabel('Amplitude')
```

Study the amplitude diagram of the Fourier transform of the two functions

$$x_1(t)=A_1\cos(2\pi f_1 t)$$
$$x_2(t)=A_2\cos(2\pi f_2 t)$$

where A_1=4, A_2=5, f_1=100*Δu and f_2=300* Δu. If you want a single sided amplitude diagram you can always plot half of the spectrum with double magnitude using

```
cut=[1: ceil(N/2)];
```

```
stem(u(cut),2*abs(Xs(cut)));
```

Study the amplitude diagram for the sum of the two functions $x_1(t)$ and $x_2(t)$.

4) Discuss the power of the sum of the two functions. How is the double sided diagram related to the power of the signal?

5. Leakage (0.25 points)

So far the sampled versions of our functions have been truncated so that the harmonic functions have frequencies that are integer multiples of $\Delta u=1/T$. The frequency Δu will have exactly one period over the truncation window T and harmonic (and periodic) functions with frequencies $u=k\,\Delta u$, $k \in \mathbb{Z}$ (an integer multiple of Δu), will always have an integer number of periods over T.

What happens if we have a periodic function with a fundamental frequency that is not a multiple of Δu? This will be the case if we keep the size of our truncated function and change the period of the function OR keep the period and change the truncation size a little. Both leads to a truncated version of the function that does not have *integer number* of periods over the truncation window!

To illustrate this change the frequency slightly from f=200* Δu to f=C* Δu; try for example C=200.1, 200.3, 200.5, 200.7 and 200.9.

5a) What happens?

The phenomenon you see is called *leakage*. In this course we will not go into the details of this phenomenon. For now you should only note, that leakage may be present when you study the outcome of a DFT. You will see leakage in some of the following exercises. When you have leakage, it is often more convenient to use the `plot` command in MATLAB instead of `stem`. Figure 1 illustrates the phenomenon.

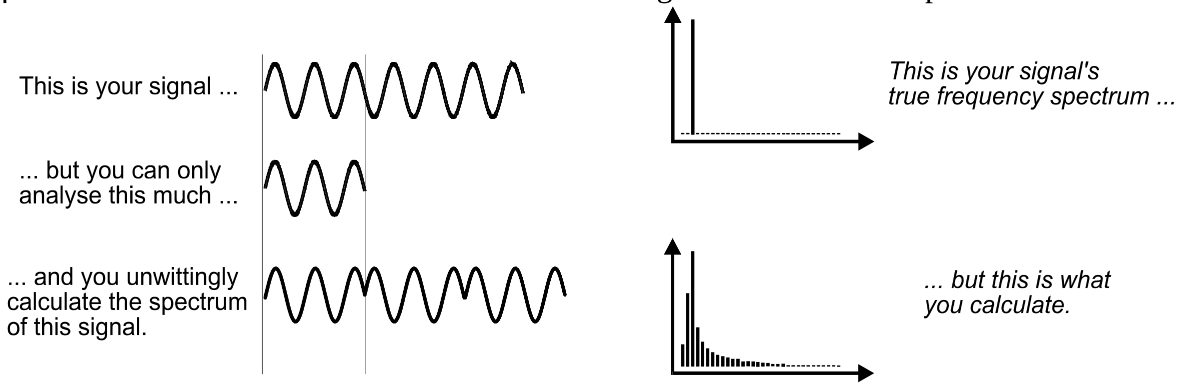
This is your signal ...
... but you can only analyse this much ...
... and you unwittingly calculate the spectrum of this signal.
This is your signal's true frequency spectrum ...
... but this is what you calculate.

Figure 1. Leakage from truncation

5b) If you are measuring the return magnitude of a radar signal, what role will leakage play? Use your plots to backup your answer. (Hint: Look at "straddle loss")

6. The phase diagram (0 points)

We will now investigate the phase diagram. The following MATLAB commands may be used to plot the diagram:

```
xs=A1*cos(2*pi*f1*t);
Xs=fft(xs)/N;
plot(u(cut), angle(Xs(cut)));
```

Since you are using a numerical approximation, the phase frequencies not present in the signal will look noisy, and may drown the information for frequencies present in signal.
Zoom on the frequency 3 Hz. The phase for this frequency should be very close to zero, but this is not obvious when you look at the phase diagram!

To avoid including noisy phase values for non-existing or low power frequencies you can set the transform value to zero for frequencies with a calculated amplitude close to zero, and then study the phase.

```
index=find(abs(Xs)<1e-5)
Xs(index)=0;
```

When we use the command `angle` in MATLAB, we will get a phase between +/- π, i.e the phase plot will include phase jumps (discontinuities). Figure 2 illustrate the typical look of the phase map of 2D phase data with phase jumps.

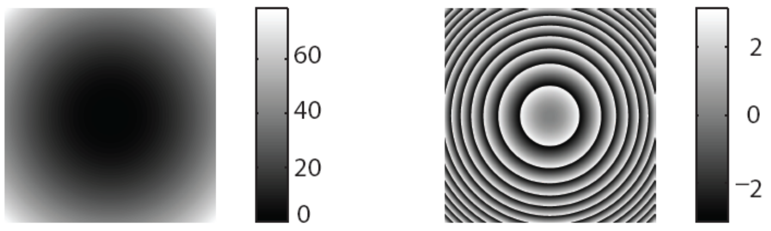

Fig. 2 Original phase map (*left*) and the corresponding phase map retrieved by a software library function from the complex numbers (*right*).

In many applications you can assume that the phase is changing slowly and you therefore use so called *unwrapping* to avoid jumps in the phase plot. In MATLAB the `unwrap` command may be used. If this command is used, MATLAB will increase the angle when a phase jump from 2π to 0 is found, 3 · 2π will then for example become 6π and not zero. Study the phase of the signal from above:

```
plot(u(cut),unwrap(angle(Xs(cut)))
```

This plot reveals that the phase computed for *u*=3Hz is not really zero but very close to zero due to numerical issues..

For simple 1-D functions like the one above, unwrapping is straight forward, but for more complicated functions it may be a challenge. A straightforward method for unwrapping 2D data is illustrated in Figure 3: Phase jumps along rows are removed and followed by removing remaining phase jumps along columns. When jumps are found, remove them by adding or subtracting 2π to the rest of the row or column. Finally piston is removed and the result is the same as the original phase. The method is very simple and limited to smooth functions..

3.770	1.885	3.770
1.885	0.000	1.885
3.770	1.885	3.770

a)

-2.513	1.885	-2.513
1.885	0.000	1.885
-2.513	1.885	-2.513

b)

-2.513	1.885	-2.513
-4.398	0.000	-4.398
-2.513	1.885	-2.513

c)

-2.513	-4.398	-2.513
-4.398	-6.283	-4.398
-2.513	-4.398	-2.513

d)

Fig. 3 The matrix in a) shows a small 3x3 phase map, with the phase given in radians. The phase retrieved from the complex values are shown in b). The corner values are negative due to phase jumps. Phase jumps along columns are removed in c) and along rows in d). If then the piston is removed, the result will be identical to a).

Unwrapping is always a part of software for imaging radar applications, but unwrapping algorithms used in these applications are often complicated.. More sophisticated phase unwrapping methods are presented in D. C. Ghiglia and M. D. Pritt, *Two-Dimensional Phase Unwrapping*. New York: Wiley, 1998.

Using the methods above study the amplitude and phase diagram for the Fourier transform of the functions:

$$x_1(t) = A_1 \cos(2\pi f_1 t + \pi / 4)$$
$$x_2(t) = A_2 \cos(2\pi f_2 t + \pi / 2)$$
$$x_3(t) = -A_1 \cos(2\pi f_1 t)$$
$$x_4(t) = A_1 \sin(2\pi f_1 t)$$

Comment on the results

7. Inverse DFT (IDFT) (0 points)

You can go back the opposite way, doing an Inverse Discrete Fourier Transform (IDFT), and compare the result with your original function. We then use MATLABs command `ifft.` It may sometimes be advantageous to take a sampled signal, perform the DFT, do some operation in the frequency domain and then go back to the time domain with an IDFT (instead of performing the operation on the time domain signals directly). If you go only one way (like in the examples above) you need to scale the function with *N*, dividing when you go from time=> frequency and multiplying when you go from frequency=>time (this is in MATLAB, it may be the opposite in other numerical libraries), but if you go both ways this is unnecessary.

Compare the signal transformed forward and backwards with the original:

```
xs=cos(2*pi*f1*t);
Xs=fft(xs);
xs2=ifft(Xs);
```

```
plot(t,xs,'r')
hold
plot(t,xs2,'b:')
```

You can also define a function in the frequency domain and transform it to the time domain.
We first try to make a function that is constant for all *t*, $x(t)=K$. This is done by setting up a vector **Xs** with all elements zero, except the first element (representing zero frequency) which we set to K :

```
K=  %Choose a value
Xs=zeros(size(xs));
Xs(1)=K;
xs2=ifft(Xs)*N;
plot(t,xs2,'k')
```

Next example produces a harmonic function with zero phase by changing the amplitude elements corresponding to the frequency of the function. Remember that you need two points and half the amplitude on every point. Let's set the frequency $10*\Delta u$ and the amplitude 6:

```
Xs=zeros(size(xs));
Xs(1+10)=3*N;
Xs(N-9)=3*N;
subplot(2,1,1)
stem(u,abs(Xs)/N);
xs2=ifft(Xs);
subplot(2,1,2)
plot(t,xs2,'k')
grid
```

Check that you get the correct frequency and amplitude by inspecting the spatial domain function. If you check period time it should be $1/(10\Delta u)$ and since the phase is zero it should be a cosine-function.

8. Rectangular function (0.25 points)
The Fourier transform of a rectangular function is a sinc-function, $\text{sinc}(x) = \sin(\pi x)/(\pi x)$). We shall now study the DFT of a rectangular function.

You sample the rectangular function $x(t) = \begin{cases} 1 & 0 \le t < 3\text{s} \\ 0 & 3s \le t \end{cases}$
for $t<=6$ with a sampling distance Δt=1s and get the sequence of samples

```
xs=[1 1 1 0 0 0];
```

8) Study the complex values of the DFT and the magnitude of the DFT. How are the two halves of the transform related?

The frequency resolution in this example is not good. It is not so easy to see if that the DFT is an approximation of a sinc-function. To increase the frequency resolution you can use *zeropadding*.

9. Zeropadding (0.5 point)

The highest frequency you can represent is half the sampling frequency $f_{samp}=1/\Delta t$ (Nyqvist criterion or sampling theorem) and the frequency resolution is determined by $\Delta u=1/T$.

If you decrease Δt (for the same truncation interval, T), N must be increased; by sampling denser you can study higher frequencies, but the frequency resolution is the same.

If you are satisfied with the frequency range, but want to increase the frequency resolution of your approximation (i.e. smaller Δu), you can keep Δt but you then need to take more samples (increase N). This may be done by adding more zeros to the function, thereby expanding T and decreasing Δu. This is called *zeropadding.*

Zeropad the function in the code above. Set

```
xs=[1 1 1 0 0 0 0 0 0 0 0]
```

and compare the amplitude spectrum with the previous one. Note that since N has increased the axis scaling must be adjusted accordingly.

MATLAB has an inbuilt option that performs zeropadding before doing the `fft`

```
M=50;
du2=1/(M*dt);
%Note: scaling with the original number of elements
Xs2=fft(xs,M)/N;  %Zeropads to M elements
u2=0:du2:(M-1)*du2;
stem(u,abs(Xs),'r', 'filled');
hold on
stem(u2,abs(Xs2),'k');
```

Can you now see the magnitude plot of a sinc-like function? The resulting function, the Dirichlet function, is sometimes called the “asinc”, the “aliased sinc”, or the “periodic sinc”, and has a “sinc-like” appearance.

9a) How much lower is the peak side lobe compared to the main lobe?
9b) At what frequencies (normalized to sampling frequency fs=1) is the amplitude 50% lower
9c) Express a and b as 10log(amplitude)

10. The Dirac's delta function δ(t) (0.25 points)

If you are not familiar with Dirac's delta function δ(t), then start by reading about this "function" (distribution).

The Delta function is extremely sharp (loosely speaking) and should therefore contain very high frequency components. Look for the Fourier transform of δ(t) in a transform table and note that the spectrum for the Dirac function spans over all frequencies and has constant amplitude. In MATLAB we can test the transform pair $\delta(t) \sim \Delta(u)$ using a unit pulse, i.e a single sample with unit amplitude:

```
d=[1];   % N=1
D=fft(d,M);
plot(u2,abs(D));
```

We can also try to produce a unit pulse using the inverse transform

```
d2=ifft(ones(1,M));
stem(d2)
```

10. What is the Fourier transform $\mathcal{F}(u)$ of a function x(t)=K, a constant level in the time domain (DC-level)? This is an example on the duality of transform pairs.

11. Sinc function (0.25 points)

Using the duality property, what would the transform of a sinc-function look like ?
Make a sampled and truncated sinc

```
dt=0.2;N=200;
t=-(N/2)*dt:dt:N/2*dt;
T=N*dt;
du=1/T;
u=0:du:(N-1)*du;
cut=[1:N/2+1];
xs=sin(pi*t)./pi./t;
%sin(x)/x is not defined for x=0, but will be 1 for lim
x→0
xs(N/2+1)=1;
```

Study the DFT.

11. What happens with the spectrum if you make the sinc-function broader?
Broader function in time domain → ??? in frequency domain
Narrower function in time domain → ??? in frequency domain

12. Detection in noise, auto-correlation and cross-correlation. (1 point)

Auto-correlation, cross-correlation and power spectrum (spectral density) will be further explained during lectures. It may therefore be advisable to postpone this part until the topics have been covered by lectures (if you do not have a background knowledge within the field)

Gaussian, white noise is not deterministic (like a harmonic function). We therefore need to use a statistical approach to characterize the noise. We will study the properties of noise by generating a number of realizations of the noise and then we will make some statistical operations on the datasets.

We start by generating one realization of the noise and then we will compute the discrete auto correlation function and the power spectrum. The power spectrum is computed in two ways; using the DFT of the realization or using the DFT of the auto correlation function. The two should be identical, but is not due to the numerical approximation (zoom and compare).

```
N=2001;
stdev=0.1 %sqrt(Watts)=> Amplitude in Volts over R= 1 Ohm
noise=randn(1,N)*stdev;
Noise=abs(fft(noise)).^2/N;      %Note the scaling
%Compute auto correlation
rn=(convn(noise,fliplr(noise),'same'));
%Compute spectral density
S=fft(rn)/N;                     %Note the scaling
mean(noise)
std(noise).^2
max(rn)
close all
subplot(3,1,1)
plot(noise,'b.-')
xlabel('t [s] ')
ylabel('Noise amplitude [V] ')
subplot(3,1,2)
plot(rn,'r--')
subplot(3,1,3)
plot(abs(S),'r--')
hold
plot(abs(Noise),'b.-')
mean(Noise)
mean(abs(S))
```

12a) What will happen if we take the mean of many realizations? Explain

```
stdev=0.1; n=0;N1=0;r=0;
for k=1:200
    noise=randn(1,N)*stdev;
    Noise=abs(fft(noise)).^2/N;
    N1=N1+Noise;
    n=n+noise;
    rn=convn(noise,fliplr(noise),'same');
    r=r+rn;
    k
end;
n=n/k;
r=r/k;
```

```
N1=N1/k;
S=fft(r)/N;

close all
subplot(3,1,1)
plot(noise,'blue')
hold
plot(n,'r')
xlabel('t [s] ')
ylabel('Noise amplitude [V] ')
subplot(3,1,2)
plot(r,'k')
subplot(3,1,3)
%Since MATLAB assumes the origin and our function has the origin in
the % %middle, S is complex and we need to take the magnitude (abs)
plot(abs(Noise),'b')
hold
plot(abs(S),'k--')
plot(abs(N1),'r:')
```

Compute and study the cross correlation between different realizations of the noise in the same way.

```
n1=randn(1,N)*std;
n2=randn(1,N)*std;
r12=convn(n1,fliplr(n2),’same’);
```

Use the file ***pulseNoise2013.m*** (on Canvas) to simulate integration of many pulses of a rectangular signal with a DC level and white Gaussian noise. The code also simulates auto-correlation with an ideal signal (rectangular pulse without DC level and noise).

12b) Try to identify the auto-correlation and cross-correlation components in the power spectrum and in the autocorrelation function.

12c) Explain why the auto-correlation function and power spectrum differs depending on where in the signal path the integration is performed.

OPTIONAL, RENDERING BONUS POINTS:
13. Spectrum (2 bonus points)

13a) Construct a simple signal (add for example some basic functions like rectangles, triangles or similar). Assume some signal amplitude units (of your choice) and some sampling distance. Numerically calculate the signal power from I) the autocorrelation function and from II) the power spectrum. Compare with the real value of the power

b) Try to shift the power spectrum of a simple signal (of your choice) by manipulating the auto-correlation function before performing the FFT.

c) Will you experience leakage. If so, when will this happen and how is it shown?

d) Use I) the auto-correlation function and II) the shifted spectrum to calculate the Doppler shift numerically from the sampled signals. Compare with the injected shift.

Please note that using "copy-paste" techniques will result in report rejection. Plagiarism will be reported to LTUs lawyer according to the Swedish national legislation.