Radar and Remote Sensing

Laboratories

Ferro Demetrio	207872
Minetto Alex	211419

Contents

Lab 1 - Evaluation of SNR and EIRP from Radar Range Equation

Task 1 - SNR Evaluation

A radar system is characterized by the following parameters: Peak power $P_t = 1.5$ MW, antenna gain $G = 45$ dB, radar losses $L = 6$ dB, noise figure $F = 3$ dB, radar bandwidth $B = 5$ MHz.
The radar minimum and maximum detection ranges are $R_{min} = 25$ Km and $R_{max} = 165$ Km.
Using MATLAB, plot the minimum signal to noise ratio versus detection range in the following conditions (in two different figures):

1. $f_0 = 5$ GHz and $f_0 = 10$ GHz with $\sigma = 0.1\ m^2$;
2. $f_0 = 5$ GHz assuming three different radar cross section: $\sigma = 0.1\ m^2$, $\sigma = 1\ m^2$, $\sigma = 10\ m^2$;

```
% Radar Parameters for SNR evaluation from Radar Ranging Equation

Rmin=25e3;           % Minimum Distance for detection
Rmax=165e3;          % Maximum Distance for detection

% noise is expressed as KbFTB

R=Rmin:100:Rmax;     % Variable for x-axes to varying distance from radar's antenna
Pt =1.5e6;           % Transmitting Power
G=10^(45/10);        % Transmitting antenna Gain
L=10^(6/10);         % System Loss
F=10^(3/10);         % Noise Figure
B=5e6;               % Bandwidth of pulse
Kb=1.38e-23;         % Boltzman Constant

% Case 1

c=299792458;         % Speed of Light
freq = 5e9;          % Frequency of transmission [first]
freq2= 10e9;         % Frequency of transmission [second]
lambda = c/freq;
lambda2= c/freq2;

s=0.1;               % Target cross section
n=1;                 % Number of pulses
T0=16+273;           % Kelvin Temperature of system
```

From the radar ranging equation for the computation of received power we can evaluate SNR as follows:

$$P_r = \frac{P_t G^2 \lambda^2 \sigma n}{(4\pi)^3 R^4 L_s} \tag{1}$$

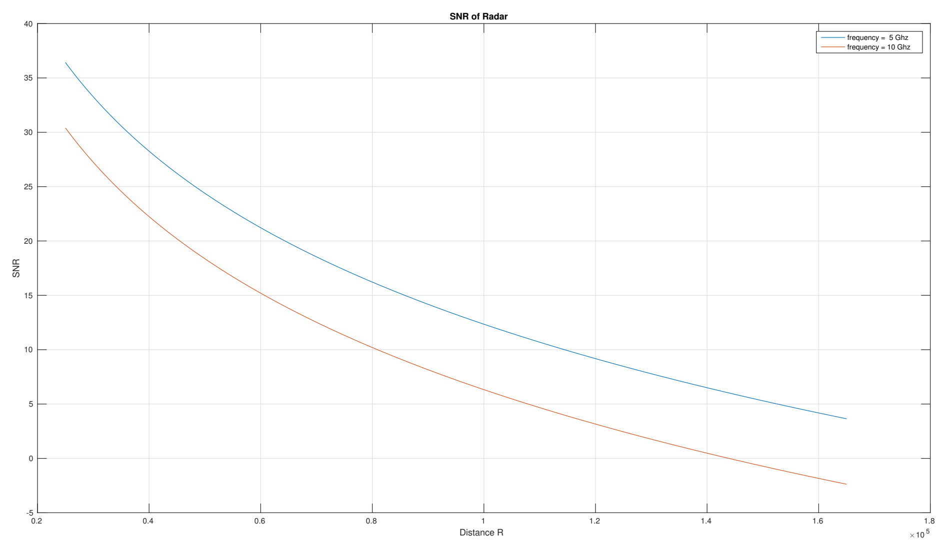
$$P_n = K_b F T B \tag{2}$$

$$SNR = \frac{P_r}{P_n} = \frac{P_r}{K_b F T B} = \frac{P_t G^2 \lambda^2 \sigma n}{(4\pi)^3 R^4 L_s K_b F T B} \tag{3}$$

We reported the formulas in MATLAB and then plotted the behaviour of SNR with respect to the distance between radar and target.

```
% Double SNR evaluation for f1 and f2

SNR=(Pt*G^2*lambda^2*s*n)./(((4*pi)^3)*(Kb*T0*B*F)*L*(R.^4));
SNR2=(Pt*G^2*lambda2^2*s*n)./(((4*pi)^3)*(Kb*T0*B*F)*L*(R.^4));

figure('Name','SNR related to distance');
plot(R,10*log10(SNR),'b',R,10*log10(SNR2),'r')
hold on;
grid on;
title('SNR of Radar');
ylabel('SNR')
xlabel('Distance R')
legend('frequency =  5 Ghz','frequency = 10 Ghz');
```

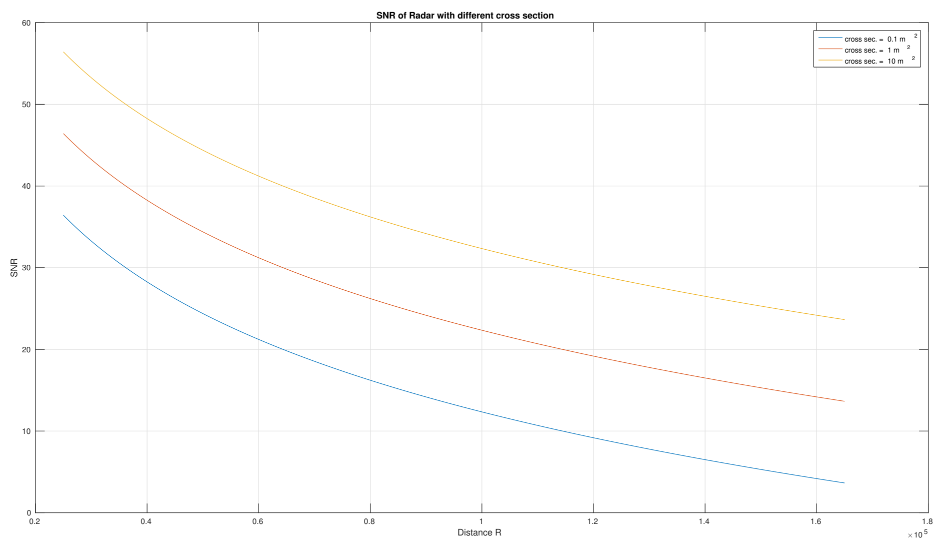
As we expected the value of SNR decreases along the distance, basically because free space attenuation degrades the output signal proportionally to the fourth power of distance R. This is a simple simulation which shows how geometrical distance can affect propagation phenomena, any other kind of attenuation is not taken in account except for the system loss that is moreover constant along the link because is a proper property of the system.



As we can notice in the plot reported above, higher frequencies suffer noise in a stronger way respect with the lower ones. This is immediate from the equation point of view because frequency is invertially proportional to λ, so, if we adopt an higher frequency we reduce the wavelength and the respective SNR value.

The second part of the exercise proposes us to evaluate the same quantity for three different cross section values. We simply report the MATLAB code that is almost the same as before. We have to precise that cross section σ is a quantity that represents the capacity of target to reflect the incoming signal, it is linked to the surface and to the volume of the object but does not represent strictly this geometrical feature.

```
sig1=[0.1 1 10];            % Vector of different cross sections
color=[1 0 0.5 1 0.5];

figure('Name','SNR with different Cross Section');
for i=1:3
   SNR=(Pt*G^2*lambda^2*sig1(i)*n)./(((4*pi)^3)*Kb*T0*B*F*L*(R.^4));
   plot(R,10*log10(SNR),'color',[color(i+1),color(i+2),color(i)])
   grid on;
   hold on;
end
title('SNR of Radar with different cross section');
ylabel('SNR')
xlabel('Distance R')
legend('cross sec. =  0.1 m^2','cross sec. =  1 m^2','cross sec. =  10 m^2');
```



As we can see in the plot, radar cross section that is in some way the "equivalent reflective surface" of the target, has a strong impact on the SNR evaluation, signals that meet target with bigger cross section are less sensitive with respect to the noise power.

For the target with the smallest cross-section, the SNR falls under 10 dB, so it could be more difficult to individuate original transmitted pulse without some other solution to make it stronger than the noise (like for example data integration or some modulation methods). Even in this simulation, equations confirm our data behaviour, received power P_r is proportional to the target cross section, it is the effective equivalent area that reaches the power density emitted by radar's antenna.

Task 2 - EIRP Evaluation

A surveillance radar is characterized by the following parameters: minimum $SNR = 15$ dB, radar losses L = 6dB and noise figure $F = 3$ dB. The radar scan time is $T_{sc} = 2.5$ s. The solid angle extent of the atmospheric volume to be surveilled is $\Theta = 3$ *srad*.
The minimum and maximum detection ranges are $R_{min} = 25$ Km and $R_{max} = 165$ Km.
Using MATLAB plot the EIRP versus detection range in the following conditions:

1. $f_0 = 5$ GHz and $f_0 = 10$ GHz with $\sigma = 0.1\ m^2$;
2. $f_0 = 5$ GHz assuming three different radar cross section: $\sigma = 0.1\ m^2$, $\sigma = 1\ m^2$, $\sigma = 10\ m^2$;

Remembering that EIRP (Equivalent Isotropic Radiated Power) is defined as the product $P_{avg} \cdot G$ where P_{avg} is the averaged power characterizing the transmitted waveform (pulsed waveform with pulse length τ and Pulse Repetition Interval T_i) and G is the radar antenna gain which, in turn, can be defined in terms of the antennas half power beam-widths θ_{3dB} and ϕ_{3dB}, as it follows using *Top-Hat Approximation*:

$$G = \frac{4\pi}{\theta_{3dB} \cdot \phi_{3dB}} \tag{4}$$

For the radar's characterization, we use almost the same code as before. In order to implement the same expression of radar equation we have to convert quantities expressed in dB in linear values.

```
%% Radar Specifics for SNR evaluation

Rmin=25e3;         %Minimum Distance for detection
Rmax=165e3;        %Maximum Distance for detection

% noise is expressed as KbFTB

R=Rmin:100:Rmax;    %Variable for x-axes to varying distance from radar
Pt =1.5e6;          %Transmission Power
G=10^(45/10);       %Transmitting antenna Gain
L=10^(6/10);        %System Loss
F=10^(3/10);        %Noise Figure
B=5e6;              %Bandwidth of pulse
Kb=1.38e-23;        %Boltzman Costant

%%case one
c=299792458;        %Speed of Light
freq = 5e9;         %Frequency of transmission [first]
freq2= 10e9;        %Frequency of transmission [second]
lambda = c/freq;
lambda2= c/freq2;

s=0.1;              %Target cross section
n=1;                %number of pulses
T0=16+273;          %Kelvin Temperature of system
```

By using theoretical hints we substitute G of radar ranging equation with the proper one evaluated in relation of beam-width solid angle. Inverting SNR equation we can solve the problem specification of maximum SNR.

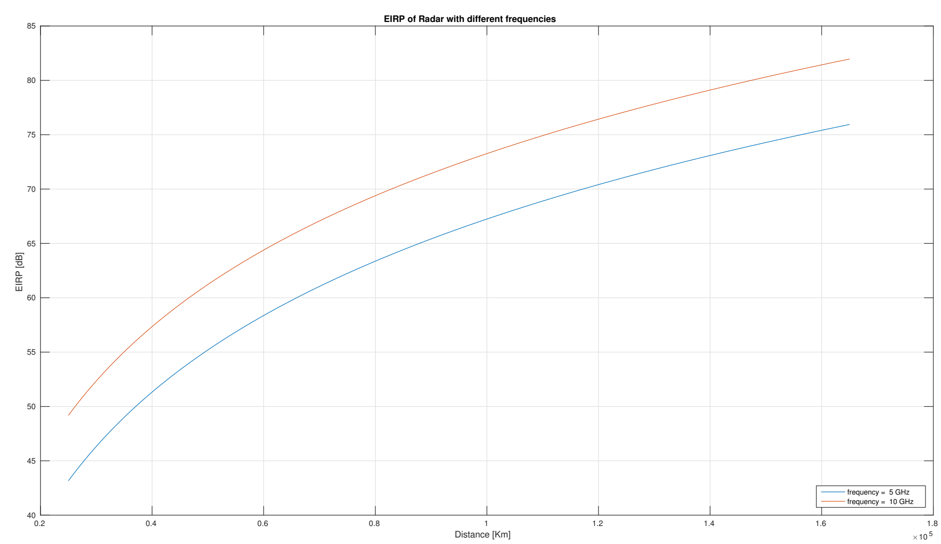
$$EIRP = P_{avg} \cdot G = \frac{P_t 4\pi}{T_s \sigma} \tag{5}$$

$$P_t = \frac{(4\pi)^3 R^4 L_s K_b F T B \cdot (SNR)}{G^2 \lambda^2 \sigma n} \tag{6}$$

$$EIRP = \frac{(4\pi)^3 R^4 L_s K_b F T B \cdot (SNR) \cdot 4\pi}{(4\pi/\Omega)^2 \lambda^2 \sigma n} = \frac{(4\pi)^2 R^4 L_s K_b F T B \cdot (SNR) \cdot \Omega}{G^2 \lambda^2 \sigma n T_s} \tag{7}$$

We can implement this derivation in MATLAB and then we evaluate the behaviour of EIRP in relation to distance from the target.

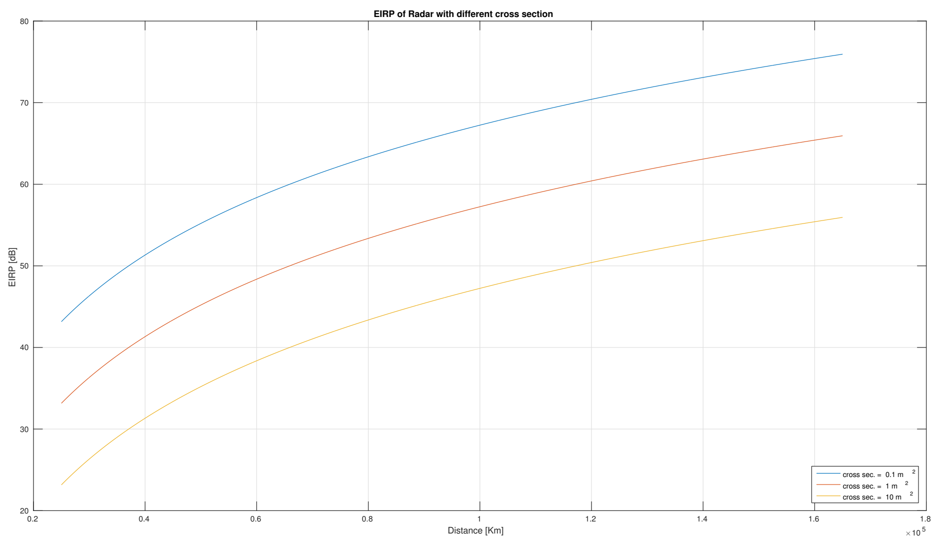
```
EIRP=(((4*pi)^2)*Kb*T0*F*L*(R.^4)*omega*SNR)./((lambda^2)*s*n*Tscan); % Linear format NO dB
EIRP2=(((4*pi)^2)*Kb*T0*F*L*(R.^4)*omega*SNR)./((lambda2^2)*s*n*Tscan);

plot(R,10*log10(EIRP),R,10*log10(EIRP2),'r'); %Plot in dB form
grid on;
title('EIRP of Radar with different frequencies');
ylabel('EIRP [dB]')
xlabel('Distance [Km]')
legend('frequency =  5 GHz','frequency =  10 GHz','Location','SouthEast');
```



We can proceed evaluating the distribution of EIRP function related with target cross section. Data are the same as the first exercise and so we can keep a similar code that is reported below for completion.

```
%% Subplot for different cross section case two

sig1=[0.1 1 10];
color=[1 0 0.5 1 0.5];

figure();
for i=1:3
   EIRP=(((4*pi)^2)*Kb*T0*F*L*(R.^4)*omega*SNR)./((lambda^2)*sig1(i)*n*Tscan);
   plot(R,10*log10(EIRP),'color',[color(i+1),color(i+2),color(i)])
   grid on;
   hold on;
end
title('EIRP of Radar with different cross section');
ylabel('EIRP [dB]')
xlabel('Distance [Km]')
legend('cross sec. =  0.1 m^2','cross sec. =  1 m^2','cross sec. =  10 m^2','Location','SouthEast
    ');
```

Here it is reported the ouput plot of the previous code section. We can notice that the behaviour of EIRP with respect to SNR is exactly inverse, that shows us the correct inverse proportionality between the two quantities.



Since smaller object reflect usually a reduced amount of power, EIRP value has to be higher in order to keep constant the SNR with low value of Cross Scattering Section.

Laboratory 2 - Refractive Index and Obstacle Diffraction

Evaluate the possibility to create a communication link between the two extreme points.

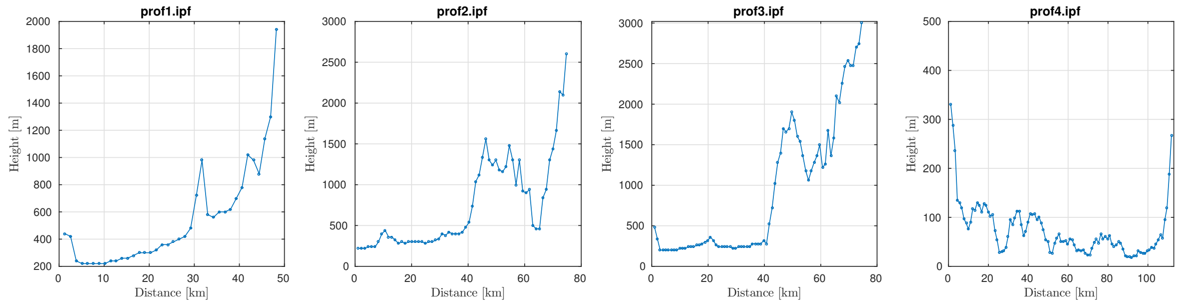
1. Download from the personal web page one of the ASCII files containing the terrain profile (profx.jpf where x=1, $\cdots$, 4) and plot it. These profiles are extracted from the Piedmont Numerical Terrain Modelling.
2. Download a radiosounding profile (from Cuneo-Levaldigi if available or Milano-Linate site).
3. Compute the spatial-averaged K value using the refractivity gradient evaluated from radiosounding data in the height layers defined by the orographic profile, (R_e = Earths radius = 6378 Km)
4. Evaluate the distance of the worst obstacle from the line of sight, considering a refractivity gradient dN/dh = -157 N/km
5. Plot the orographic profile above the modified Earth, taking into account also the two following cases:
   (a) a value $K = 4/3$
   (b) the averaged K_{mean} value evaluated using radiosounding
6. Evaluate the distance of the worst obstacle from the line of sight considering 5a and 5b cases.
7. Compare these distances with the first Fresnel radius dimension (at the point where the worst obstacle is), considering a working frequency of 10 GHz and evaluate the attenuation due to diffraction effects, considering the three cases (dN/dh = -157 N/km, 5a case, 5b case).

Task 1 - **Orography Acquisition**

We are just commenting the first two tasks in order to show the MATLAB's code for the data processing and the relative plots. We report below the plots of terrain profiles from Piedmont model.

```
for i=1:4
x=load(strcat('prof',int2str(i),'.ipf')); % data loading from profx.ipf files
figure(1)
subplot(2,2,i)
%plot(x), hold on
plot(x(:,1),x(:,2),'-b.'), grid on
title(strcat('Orography ',int2str(i)));
xlabel('Distance [km]');
ylabel('Height [m]');
end
```

The code before is useful to get geographic profiles from proper files and plot them in order to see the orographic element's shape.



Task 2 - Orographic profiles Download

Downloading Radiosound profile for empirical evaluation of refractive index, we have to extract specific information useful for the Beam and Dutton Formulae. As we can see in the extract of code we need the atmospheric pressure, relative altitude, temperature and relative humidity, so we extract this informations from the downloaded file.

Task 3 - Refractivity Index Computation

And then we compute the atmospheric refractivity by using Beem and Dutton Formulae. Through the following cycle we evaluate the K_{mean} for each terrain profile loaded before. K_{mean} represents an approximation of the gradient of refractive index n with respect to the altitude, once converted in N units, it will be a multiplicative coefficient useful to modify earth's radius in our simulation model.

```
%% Points 2-3

mRSP=load('cuneo-levaldigi.txt');

vHPA=mRSP(:,1);              % Atmospheric Pressure
vHGT=mRSP(:,2);              % Height
vTEMP=mRSP(:,3);             % Temperature
vRELH=mRSP(:,5);             % Relative Humidity
vWVPP=(vRELH/100*6.122).*(exp((17.67*vTEMP)./(243.5+vTEMP)));
              % Water Vapour partial pressure e(h)

% Atmoshperic Refractivity N(h) with Been and Dutton Formulae
N=77.6*(vHPA./(vTEMP+273.15))-6*(vWVPP./(vTEMP+273.15))+3.73e5*(vWVPP./(vTEMP+273.15).^2);

Re=6378e3;           % Earth's radius
dNoverdH=0;          % Gradient variable for following calculations
mean_k=zeros(1,4);       % Vector of k coefficients

for j=1:4
% Interval where to compute kmean
[vIndStart,nValue1]=find(vHGT > nTxHeight(j));
[vIndStop,nValue2]=find(vHGT > nRxHeight(j));
Nlev=vIndStart(1):vIndStop(end)-1;

for i=Nlev
    dNoverdH=dNoverdH+(N(i+1)-N(i))/(vHGT(i+1)-vHGT(i))/length(Nlev);
end
```

```

mean_k(j)=1/(1+(Re*1e-6*dNoverdH));

fprintf('The mean dN/dH of k-order is: %2.2d\n',mean_k(j));
end
```

The output of this extract of code is the following:

```
The mean dN/dH of k-order is: 1.07e+00
The mean dN/dH of k-order is: 1.14e+00
The mean dN/dH of k-order is: 1.23e+00
The mean dN/dH of k-order is: 1.33e+00
```

As we can notice in the output lines, the mean value of the refractive index gradient decrease with the altitude. Lower values refer to lower profiles scenario while $K_{mean} = 1.33$, which represent a quite strong modification of earth's curvature, is relative to near-sea-level profile. This is a typical trend for the refractive index due to the variations in temperature and humidity through the atmosphere that usually decreases with the height.

Task 4 - Distance from LOS

Evaluating distances (or better height) considering a refractivity gradient $dN/dh = -157N/Km$ means to consider the propagation bending with a curvature that is exactly the same of the earth's surface (practically we don't have any modifications on the line of sight height), so it's the same issue to compute this amount considering $K_{mean} = 1$.
So we can evaluate the behaviour of bending with the variation of K as it is request at point 6.
We report here the portion of code used to evaluate the 4th point of this assignment, the output of the code is instead reported below in task 6.

```
% Refractivity index
dNdH=-157;
%ourK=1/(1+Re*1e-6*dNdH);
%mean_k=1;
x=load(strcat('prof',int2str(j),'.ipf'));
[AbsLOS(:,1), nDifferencePrec(:,1)]=Distance_from_obstacle_to_LOS(ourK,1);
```

Task 5 - Modified Earth Profiles

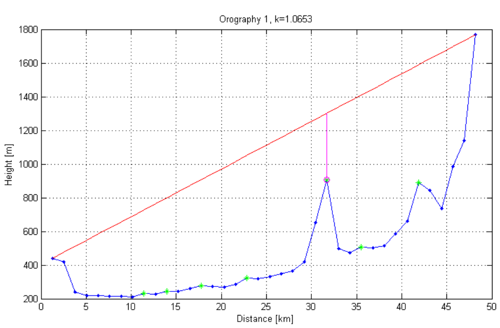
We need to plot the terrain profiles with the proper bending induced by refractive index. This is a useful representation that allows us to keep the propagation line horizontal and change curvature of the ground in order to appreciate the real distance of obstacles from line of sight. We have to compute the variation in height for each point of the profile and then we can modify the original orography with the modified earth radius approach.

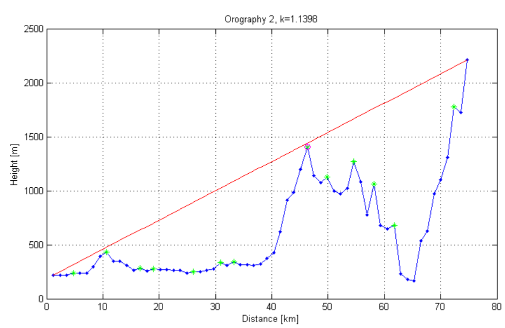
```
%% Point 5
for i=1:4
x=load(strcat('prof',int2str(i),'.ipf'));

% try with k=1
k=1e0;
```

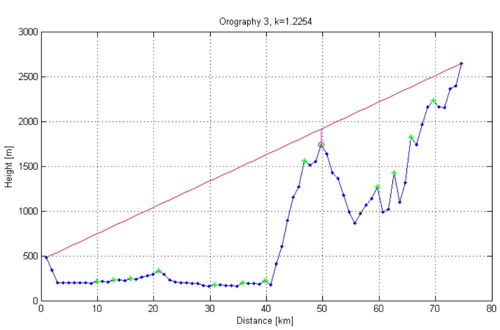
```
vVarh=(x(:,1).*1e3).^2./(2*k*Re);
vHmod=x(:,2)-vVarh;
figure(2)
subplot(2,2,i);
% plot(x(:,1),vHmod,'g.-'), hold on, grid on
 plot(x(:,1),vHmod,'g.-'), hold on, grid on

% try with k computed in the 3rd point
k=mean_k(i);
figure(2)
vVarh=(x(:,1).*1e3).^2./(2*k*Re);
vHmod=x(:,2)-vVarh;
plot(x(:,1),vHmod,'r.-'), hold on, grid on

% now with k=4/3
k=4/3;
figure(2)
vVarh=(x(:,1).*1e3).^2./(2*k*Re);
vHmod=x(:,2)-vVarh;
plot(x(:,1),vHmod,'b.-'), hold on, grid on
title('Orography with respect to k.');
legend('k=1',sprintf('k=%d',mean_k(i)),'k=4/3','Location','NorthWest');
end
```

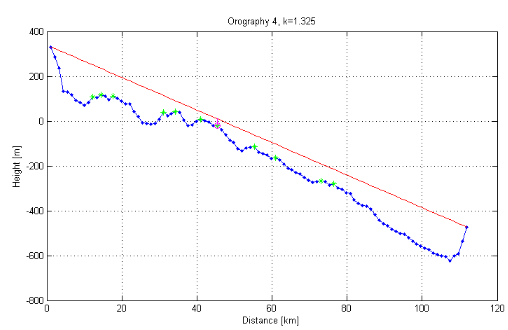
In the following pages there are four plots for each value of $\frac{dN}{dh}$, one for each terrain profiles given before.

Profiles evaluated with K_{mean}:

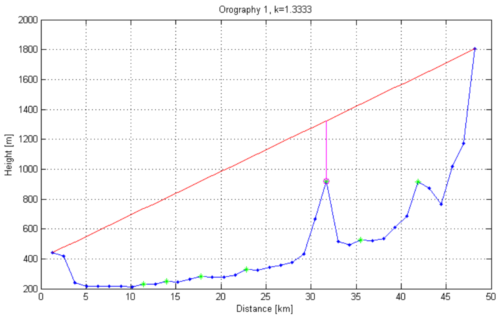


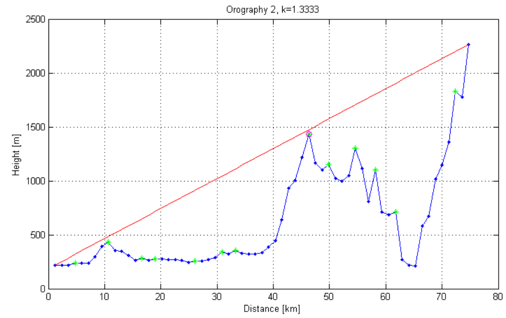


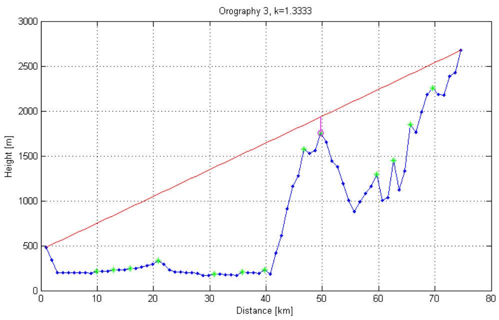


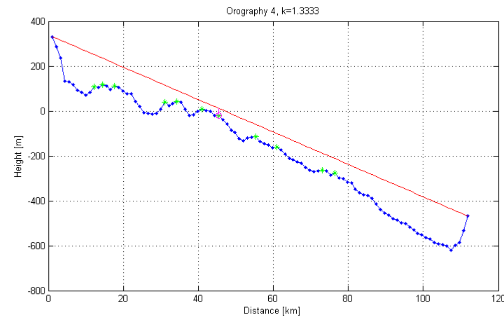


Profiles evaluated with $K = \frac{4}{3}$:

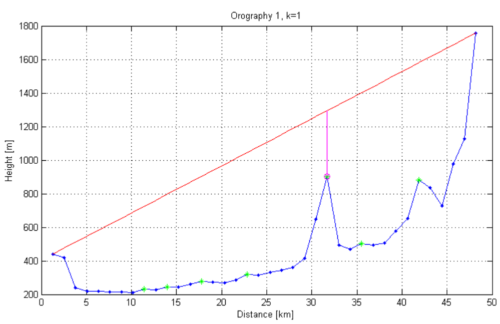


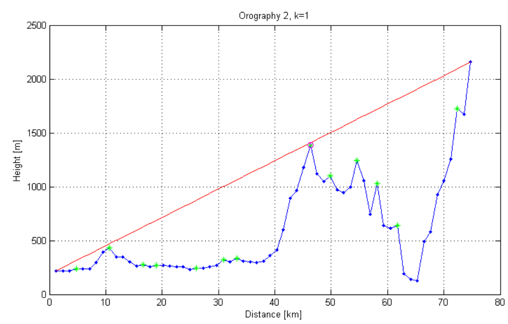


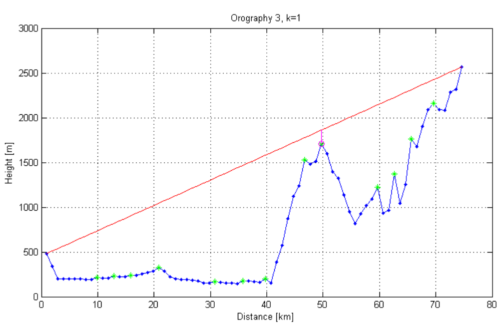


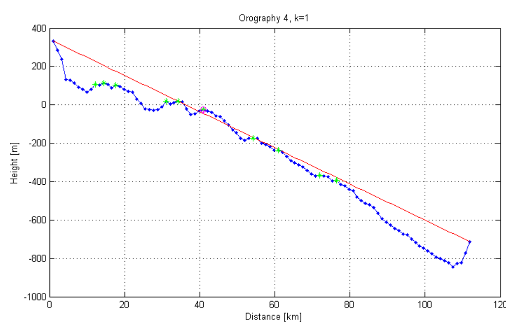


Profiles evaluated with $K = 1$:

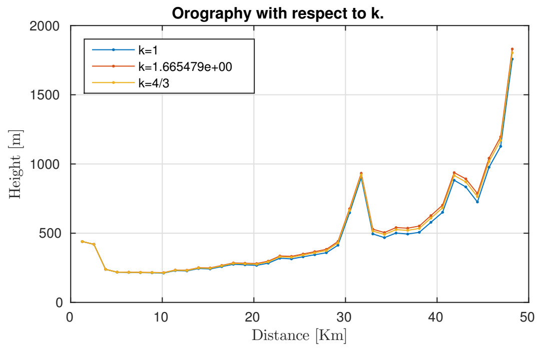


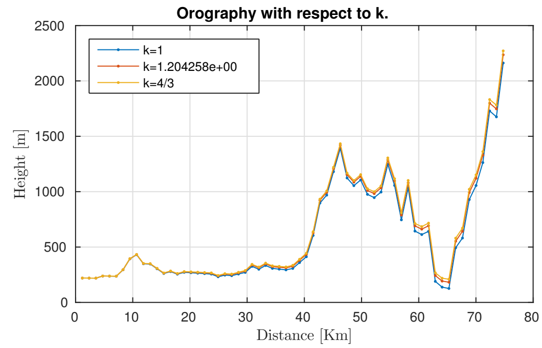


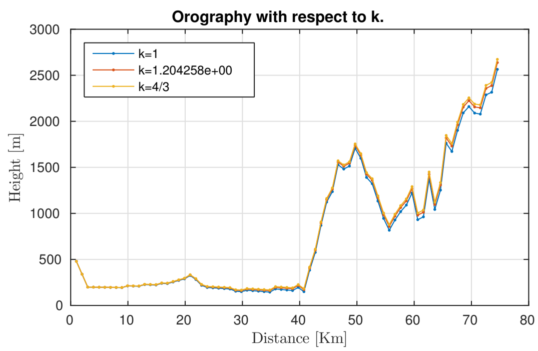


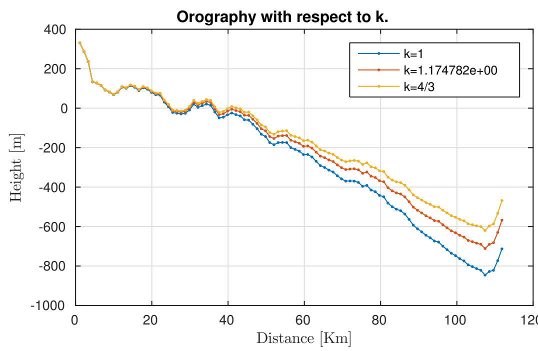


In the next plot it is possible to see the superposition of different modified earth profiles in order to make the difference more visible.
We can see that lower altitudes are more curved than higher ones properly for what we said before about refractive index and its height-decreasing trend (tipically of about $-20N/Km$).









Task 6 - Modified Earth Distance from LOS

Proceeding from Task 4, we evaluate the distance of the worst obstacles from the lines of sight of each transreceiving system. Here is reported the full matlab code for the accumulation in a vector of various data collected about this point.

For this Task we have developed a function that evaluates the specific distance of obstacles from line of sight. The code of the function is reported after the code's lines of computation. The function accepts as parameters the index of reference figure (relative to ground's profile) and the K_{mean} coefficient for earth's curvature.

```
%% Point 6 evaluate the distance of the worst obstacla from the LOS
[AbsLOS(:,2), nDifferencePrec(:,2)]=Distance_from_obstacle_to_LOS(1,3);
[AbsLOS(:,3), nDifferencePrec(:,3)]=Distance_from_obstacle_to_LOS(mean_k,4);
[AbsLOS(:,4), nDifferencePrec(:,4)]=Distance_from_obstacle_to_LOS(4/3,5);

%% Code for Distance_from_obstacle_to_LOS(n,m)
function [AbsLOS, nDifferencePrec]=Distance_from_obstacle_to_LOS(k,indfigure)
Re=6378e3;
AbsLOS=zeros(1,4);
nDifferencePrec=zeros(1,4)+1e10;

for ii=1:4

x=load(strcat('prof',int2str(ii),'.ipf'));

if(length(k)>1)
```

```
vk=k(ii);
vVarh=(x(:,1).*1e3).^2./(2*vk*Re);
x(:,2)=x(:,2)-vVarh;
elseif(k==0)
vk=k;
else
vk=k;
vVarh=(x(:,1).*1e3).^2./(2*vk*Re);
x(:,2)=x(:,2)-vVarh;
end

% Compute LOS
nDTx=x(1,1)*1e3;
nDRx=x(end,1)*1e3-nDTx;
nHTx=x(1,2);
nHRx=x(end,2);
vHighLOS=linspace(nHTx, nHRx, length(x(:,1)));
AbsLOS(ii)=sqrt((nDRx)^2+(nHTx-nHRx)^2);


% Find Possible Peaks
[vPeaks,vIndexPeaks]=findpeaks(x(:,2));
vDifference=vHighLOS-x(:,2)';

% Find Highest Obstacle
nIndPeaks=1;
nIndObst=1;

for nInd=2:length(vDifference)-1
    if nIndPeaks<=length(vIndexPeaks)
        if(nInd==vIndexPeaks(nIndPeaks))
            if(vDifference(nInd)<nDifferencePrec(ii))
                nDifferencePrec(ii)=vDifference(nInd);
                nIndObst=nInd;
            end
            nIndPeaks=nIndPeaks+1;
        end
    end
end

% Plot profile
figure(indfigure)
subplot(2,2,ii)
plot(x(:,1),x(:,2),'-b.'); hold on, grid on

% Plot LOS
plot(x(:,1),vHighLOS,'r'); hold on

% Plot Peaks
for j=1:length(vIndexPeaks)
    plot(x(vIndexPeaks(j),1),x(vIndexPeaks(j),2),'g*');
end
```

```
% Plot Highest Peaks
for kk=1:length(nIndObst)
    plot(x(nIndObst(kk),1),x(nIndObst(kk),2),'mO'); hold on
    plot([x(nIndObst(kk),1) x(nIndObst(kk),1)],[x(nIndObst(kk),2) nDifferencePrec(ii)+x(nIndObst(
    kk),2)],'m'), hold on
    plot([x(nIndObst(kk),1) x(nIndObst(kk),1)],[x(nIndObst(kk),2) nDifferencePrec(ii)+x(nIndObst(
    kk),2)],'m'), hold on
end

% Titles
%if(k>0)
title(['Orography ' num2str(ii) ', k=' num2str(vk)])
%else
%title(['Orography ' num2str(ii)])
%end
xlabel('Distance [km]');
ylabel('Height [m]');
end
end
```

This implementation provides also plots of LOS over several modified terrain profiles and in the overlying plot, it points out the highest peaks of each profile. In order to avoid a very long report, plots are reported in the previous task as superposition of different requested figures.

Task 7 - First Fresnel Radius Computation

The following code implements the computation of First Fresnel Radius for the different profiles:

```
%% Point 7 compute the first fresnel radius

freq=1e10;
c0=physconst('LightSpeed');
lambda=c0/freq;

Rf=zeros(1,4);

for i=1:size(nDifferencePrec,2)
    for j=1:size(nDifferencePrec,1)
    Rf=sqrt(lambda*AbsLOS./4);
    fprintf('Distance: %3.2f \tFresnel radius:%3.2f\n',nDifferencePrec(j,i),Rf(j,i));
    end
    fprintf('\n');
end
```

We report below the final outuput which include the distance of the worst obstacle for each orography and the respective fresnel radius in order to evaluate immediately how much the obstacle can degrade propagation performance.

```
Distance: 432.97  Fresnel radius:18.76
Distance: 87.10   Fresnel radius:23.49
Distance: 248.65  Fresnel radius:23.48
```

```
Distance: 193.32  Fresnel radius:28.81
Distance: 393.61  Fresnel radius:18.76
Distance: 19.02   Fresnel radius:23.49
Distance: 153.81  Fresnel radius:23.48
Distance: -20.42  Fresnel radius:28.81

Distance: 396.02  Fresnel radius:18.76
Distance: 31.36   Fresnel radius:23.49
Distance: 171.26  Fresnel radius:23.48
Distance: 30.36   Fresnel radius:28.81

Distance: 403.45  Fresnel radius:18.76
Distance: 44.19   Fresnel radius:23.49
Distance: 177.52  Fresnel radius:23.48
Distance: 31.45   Fresnel radius:28.81
```

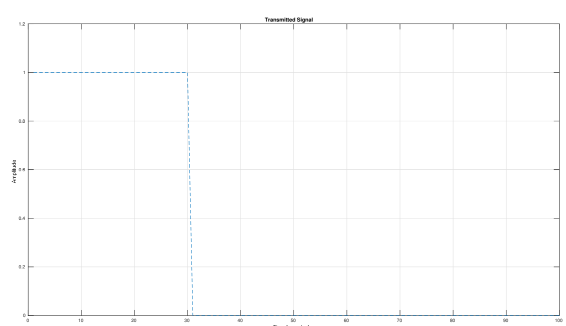
As we can see from the outputs, the worst obstacle in the first scenario is quite far from line of sight and since the Fresnel radius is small it cannot interfere in the propagation.

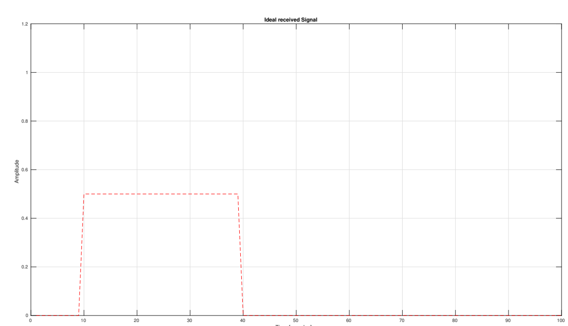
Laboratory 3 - Detection of echo return by Signal Integration

Task 1 - Generation of Transmitted Signal

The transmitted signal x_n is a rectangular pulse with amplitude A and fixed length τ. Generate the transmitted signal in Matlab for $A = 1$ and $\tau = 30$ (samples). The total length of the signal is 100 samples. Plot it.

```
% Transmitted signal
A=1;
t=30;
tot=100;
X=[A*ones(1,t) zeros(1,tot-t)];
figure('name','Transmitted Signal');
plot(X,'--','linewidth',3)
grid on;
axis([0 100 0 1.2])

%% Received Signal
B=0.5;
tr=30;
delay=9;
N=randn(1,100);

Y=[zeros(1,delay) B*ones(1,tr) zeros(1,tot-delay-tr)];
figure('name','Received Theoretical Signal');
plot(Y,'--r','linewidth',3)
title('Ideal Received Signal')
xlabel('Time [samples]')
ylabel('Amplitude')
grid on;
axis([0 100 0 1.2]);
```



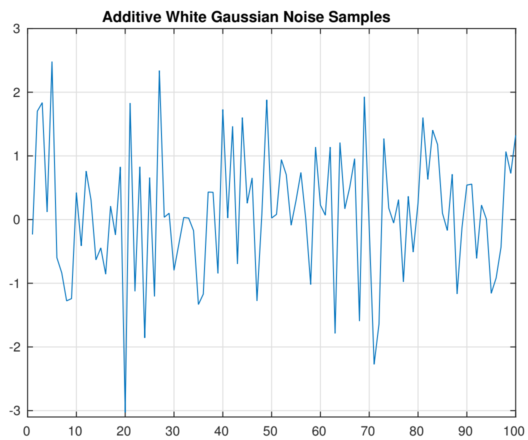


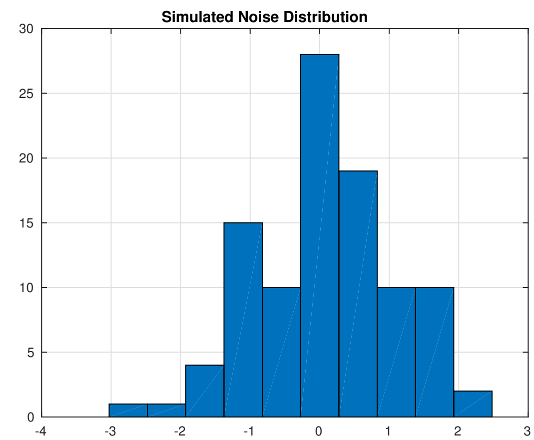
We plot both the ideal TX and RX signals in order to evaluate easily what are the shape modifications due to noise in the second part of this exercise. For the ideal RX signal we have just applied an attenuation of amplitude that could be modelled as a typical free space loss.

The signal received y_n is a rectangular pulse with amplitude $B = 0.5$, length $\tau = 30$ samples, delay of $\Delta = 9$ samples, plus additional noise. Noise signal amplitude N_n should be modeled as Gaussian distributed random variable with zero mean and a unitary standard deviation.

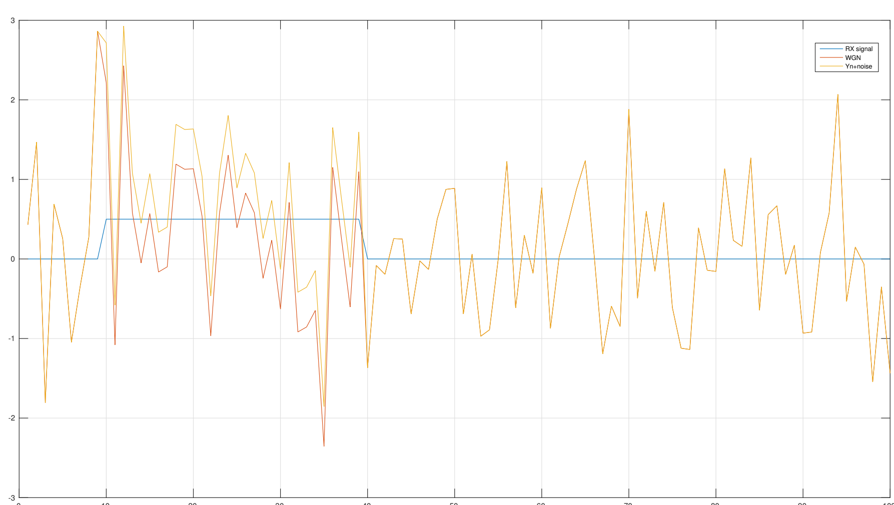
Check that the noise is correctly generated plotting an histogram showing the distribution of noise samples. Generate the overall received signal (the total length of the signal is 100 samples). Plot all together the transmitted signal (x_n), the noise (N_n) and the received signal ($y_n = x_{n-\Delta} + N_n$).

```
figure('name','Received Noisy Signal');
plot(Y,'k--','linewidth',2)
hold on
title('Noise vs Ideal Received Signal')
xlabel('Time [samples]')
ylabel('Amplitude')
plot(N,'r','linewidth',1)
```





Thanks to the plots we can notice that noise amplitude is considerably higher than signal amplitude. At the receiver, original signal is completely hide by the noise and it seems to be unreachable. On the right we have verified noise's samples distribution which appears exactly Gaussian.



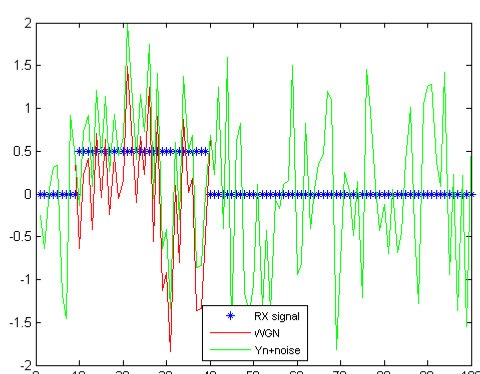
As requested, the following plot shows a superposition of the different elements in signal transmission. We have to underline that the plot does not correspond to the same previous realizations because it is the product of a new random instance of WGN.

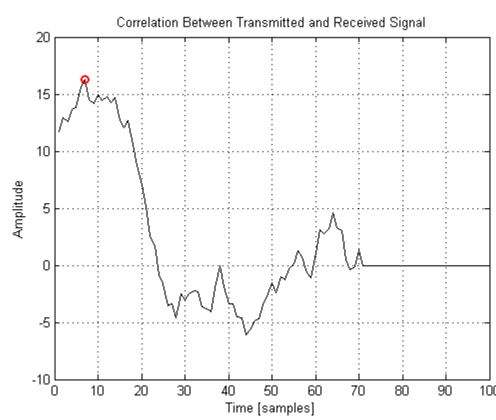
Task 6 - Convolutional Signal Identification

Generate the Matlab code for evaluating the convolution between the received signal ($y_n = x_{n-\Delta} + N_n$) and the transmitted one (x_n). Plot the entire convolution ($C_n(x, y)$) and identify the correlation lag for which the convolution shows a maxima. It should be delayed by $\Delta = 9$ samples. Make several trials (each time the noise generated must be different), and check that the maximum correlation lag is not always found for $k = 9$.
In order to get the suggested result we cant use standard MATLAB function for the convolution, the convolution techniques from signal theory find the peak of autocorrelation after a whole signal length plus the relative delay. In this case we are looking for delay identification and so we can rewrite a proper convolution code which shows us a good result for this application.

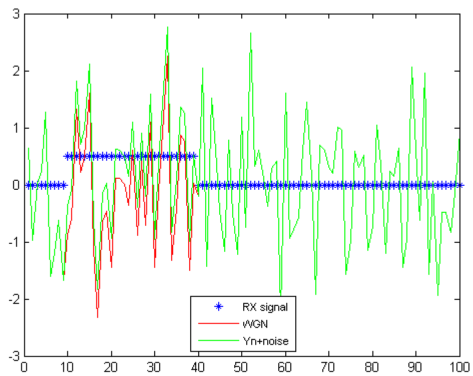
```
%% Correlation of signal
figure('name','Correlation between transmitted and received signal');
Yn=Y+N;
%Cxy=xcorr(X,Yn);
Cxy=zeros(1,tot);
for s=1:tot-t
   for i=1:t
    Cxy(s)=Cxy(s)+X(i)*Yn(i+(s));
   end
end
plot(Cxy,'k')
%[pks,locs] = findpeaks(Cxy);
hold on;
grid on;
%plot(locs,pks,'ro')
[Ymax,Xmax]=max(Cxy)
plot(Xmax,Ymax,'ro');
```

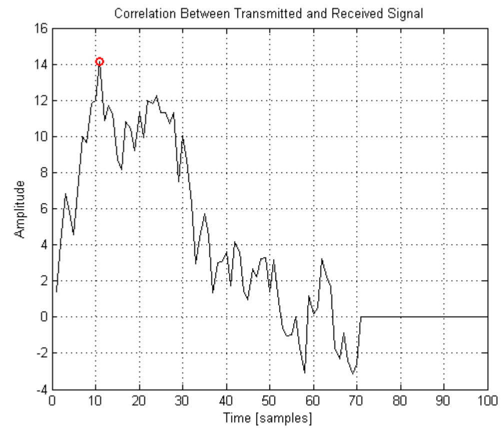
We are speaking about correlation as a convolutions synonym, the original signal and the received one are different for sure because delay and noise has changed the rectangular shape.





So, the one on the right is the plot of correlation between original signal and received one, it is a typical operation done by receiver in order to identify the delay of the signal, we have to notice that if we have a delay equal to τ_{rx} the range of object detected by the system is the half of that value: $\frac{\tau_{rx}}{2}$





We can observe in the plots that maximum of correlation is not always correct, the variation is due basically to the amplitude of the noise that as we said before, masks signal behaviour in a stronger way.
In order to obtain a more robust value for correlation's peak we have to evaluate the effects of data integration. This technique allows us to collect power information of a certain amount of pulses and obtain a clean shape of the received signal by knowing that noise has a Gaussian zero-mean distribution. If the target is stationary, B is always the same.
For each received pulse (for each generated $y_n = x_{n-\Delta} + N_n$ signal) only the noise is changing.

Task 3 - Data Integration

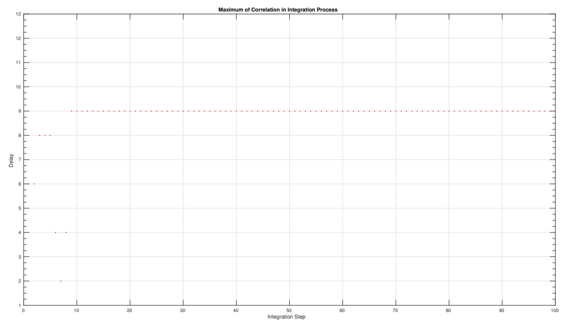
Generate N different received signals $r_{n,i} = y_n + N_{n,i}$ ($i = 1, \cdots, N$ where $N = 1 : 1 : N_{MAX}$ and $N_{MAX} = 100$) where y_n has a constant amplitude B = 0.5. For each N (from 1 to N_{MAX}), sum together all the signals (integrated signal). Compute the convolution between the integrated signal and the transmitted one (x_n) and evaluate (for each N) the correlation lag for which the correlation is maxima. Like in the figure here below, plot the lag for which max correlation is found in function of the number of signals N summed together (plot D).
This value should not always be 9, but it is stabilized after having integrated a certain number of received signals (in the picture here below, summing 35 received signals, the maximum of correlation is found for the correlation lag = 9 and from this point it is stabilized).

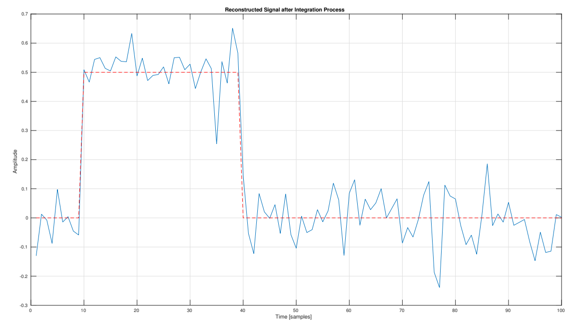
```
%% Data integration
int_samples=100;
Yint=zeros(1,100);
figure('name','Evolution of Maxima for integrated signal');
h = waitbar(0,'Please wait...');
for n=1:int_samples
    N=1*randn(1,100);
    Yn=Y+N;
    Yint=Yint+Yn;
    %intConv=conv(X,Yint);
    intConv=zeros(1,tot);
```

```
    waitbar(n / int_samples)
    for s=1:tot-t
        for i=1:t
        intConv(s)=intConv(s)+X(i)*Yint(i+(s));
        end
    end
    [Ymax,Xmax]=max(intConv);
    plot(n,Xmax,'r.')
    title('Maximum of Correlation in Integration Process')
    xlabel('Integration Step')
    ylabel('Delay')
    axis([0 int_samples 5 12]);
    grid on
    hold on;
end
close(h)
```

At the end of the simulation and of the signal integration process, we can plot the equivalent extracted signal and notice that thanks to the elaboration we have reach correctly the original signal in bad noise condition.

```
figure('name','Integrated signal and theretical received signal');
plot(Yint/int_samples)
hold on
 title('Reconstructed Signal after Integration Process')
    xlabel('Time [samples]')
    ylabel('Amplitude')
grid on
plot(Y,'--r');
```

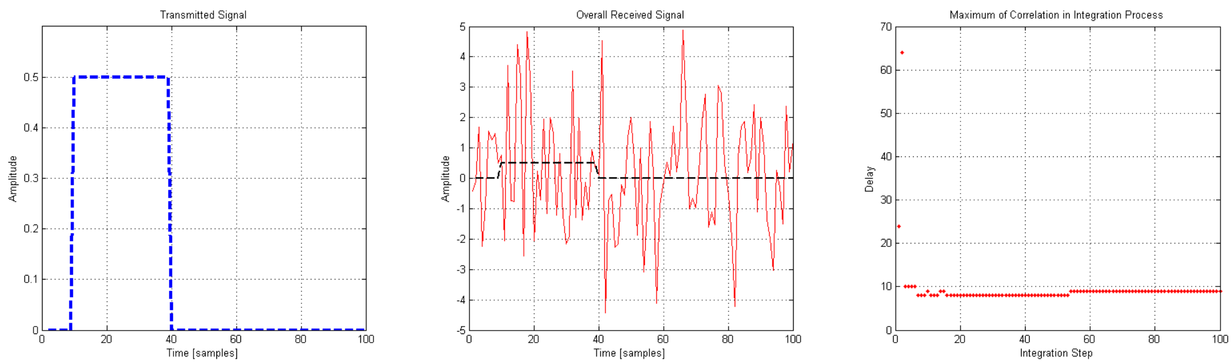


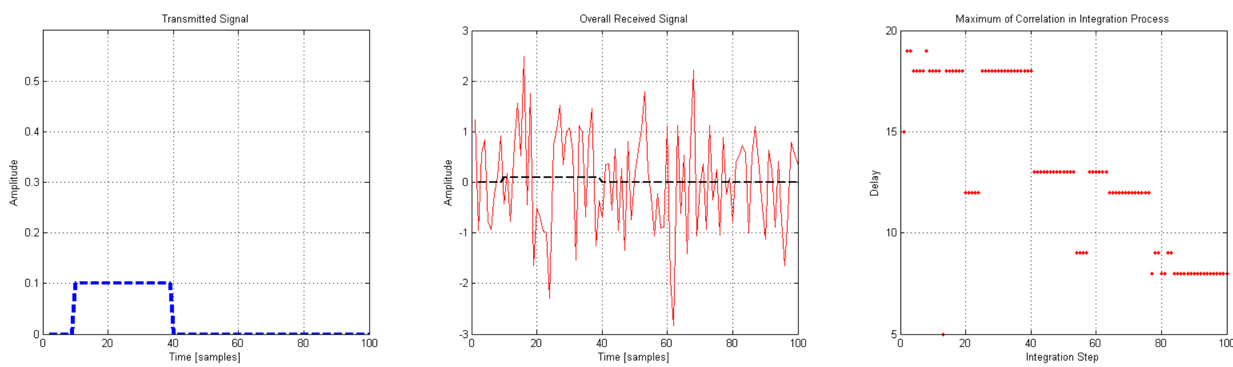


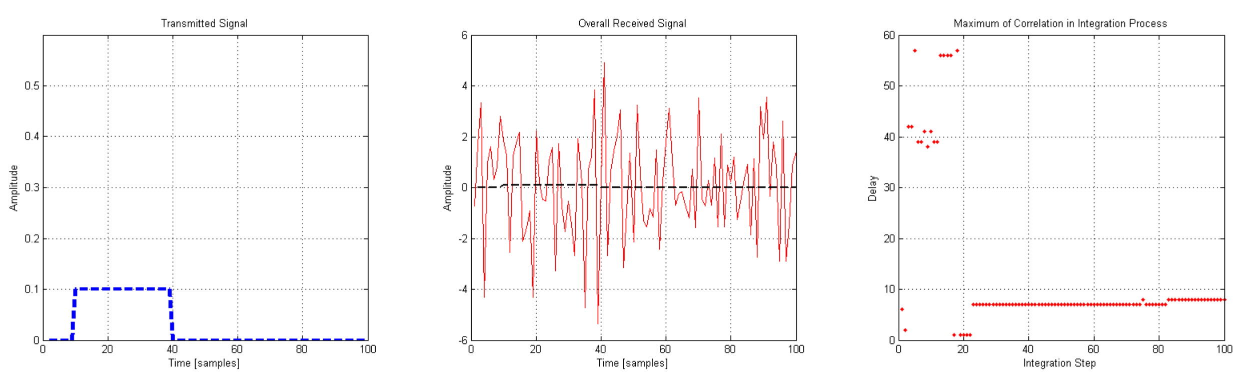
Integration process represent a good way to find signal typically correlated through different measurements, reducing the noise disturb over the samples thanks to its own distribution.
We have to remind that different noise distributions do not behave as for the Gaussian one.

Task 4 - Change of Parameters

Repeat Task 3 changing the following parameters:
$B = 0.5$ and noise (Gaussian distribution) with $E[n] = 0$ and $\sigma = 2$ (plot E)
$B = 0.1$ and noise (Gaussian distribution) with $E[n] = 0$ and $\sigma = 1$ (plot F)
$B = 0.1$ and noise (Gaussian distribution) with $E[n] = 0$ and $\sigma = 2$ (plot G)
If the lag of the correlation peak is not set, it is possible to increment the number of signals to be integrated (sum over more N_{MAX} signals).







Laboratory 4 - Radar Meteorology Application

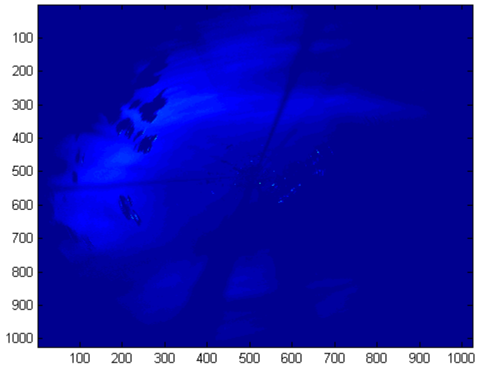
Task 1 - Hourly Cumulative Rain Map

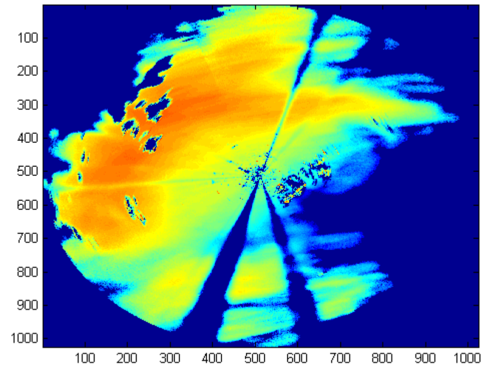
Starting from the raw radar maps (which represent received power [in DN] evaluated on a time interval of 1 minute), compute the hourly cumulated rain maps (containing R in $[mm]$). In the report plot only one of that (the most representative in terms of rain quantity), using the matlab command imagesc. Pay attention that, from each DN contained inside each radar map, you should extract $ZdbZ$ and its corresponding R value, which is the rainfall rate for that minute in $[mm/h]$. Then you should compute the cumulated rain during the overall hour.

In the following code we have computed the hourly cumulated rain maps from the radar data:

```
%% Cumulative Hourly Rain Map

rdpath='C:/Users/Alex/Desktop/radar_data/';
filelist=dir(rdpath);
R_mat=zeros(1024,1024);
t=3;
h = waitbar(0,'Wait please');
for hour=1:23
    waitbar(hour/24)
    R=0;
    R_mat=zeros(1024,1024);
for i=t+60*(hour-1):(60*hour)+t-1
    DN=imread(strcat(rdpath,filelist(i).name),'png');
    ZdBz=((double(DN)./2.55)-100)+91.4;
    Zmm_m=10.^(ZdBz./10);
    R_mat=R_mat+(double(Zmm_m)./316).^(2/3);
end
    subplot(4,6,hour);
    %figure('Name','Cumulated Map','NumberTitle','off');
    %imagesc((R_mat./60));
    %figure('Name','Cumulated Map [Log]','NumberTitle','off');
    imagesc(log(R_mat/60));
end
close(h);
```

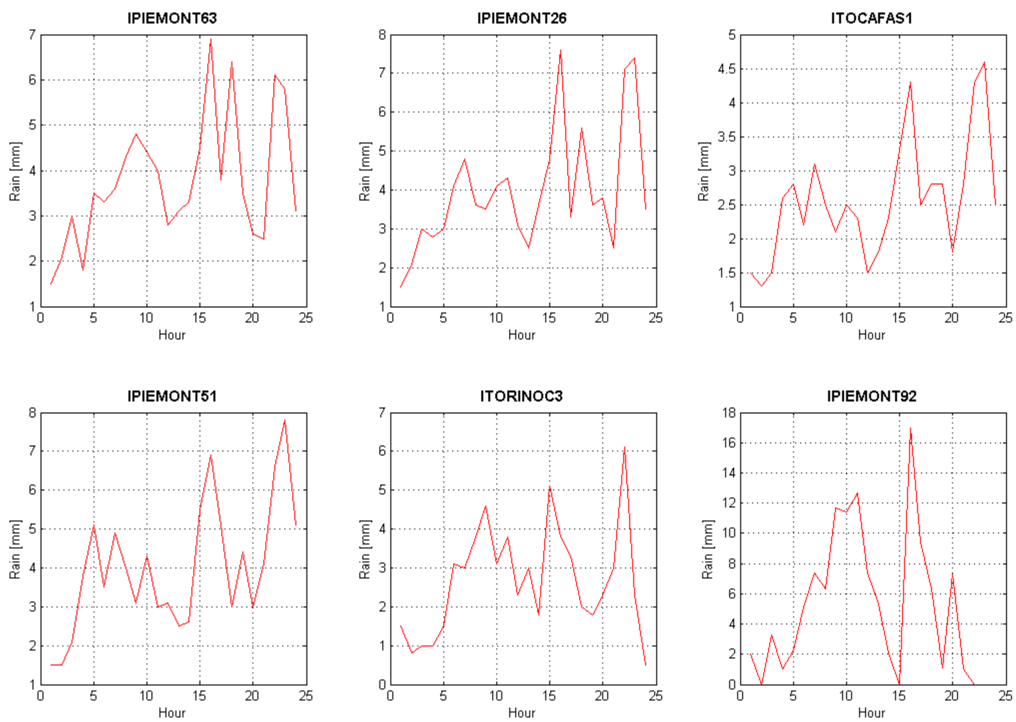
As we can notice in the code, the value of each pixel is converted following the proper equation in order to get, at the end of the process, the Rainfall Rate R. By summing each sample we must take in account that each value converted in R is evaluated in mm/h so after the sum and the linear conversion we need to divide by 60 (minutes in a hour).Here we report the most significant Cumulative Rain map with standard linear visualization and the equivalent log scale plot that is more comprehensible (low difference between values of R are badly represented in colour gradient so we need to compress image's dynamic with logarithmic scale).





Task 2 - Hourly Cumulative Behaviour from Terrestrial Gauges

Choose the rain gauges from the /gaugedata/raingauges-2012-11-28.txt, extract and plot (in two different matlab figures) the time series of hourly cumulated rain data (mm) for the overall day.




```
T = readtable('raingauges_28-11-2012.txt','HeaderLines',1,'Delimiter','\t');
G_mat=zeros(7,24);
t=1;
for g=1:6
   G_Mat(g,:)=T.Var5(t:t+23);
   subplot(2,3,g)
   plot(G_Mat(g,:),'r')
    grid on;
```

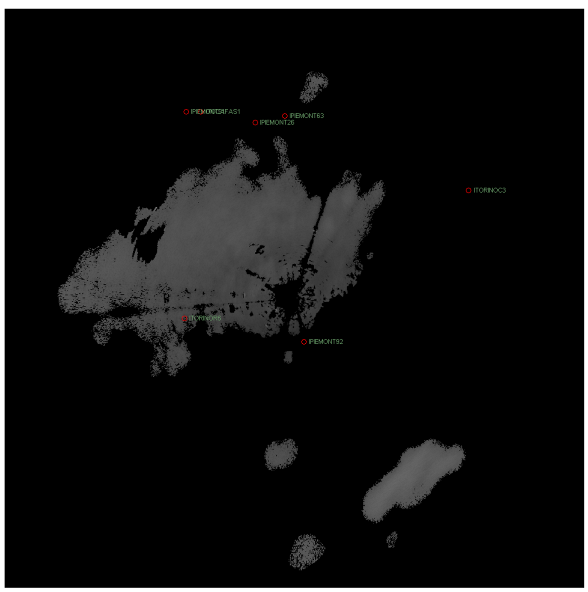
```
    title(gauge(g).name,'FontWeight','bold','FontSize',11,'FontName','Arial')
    xlabel('Hour','FontName','Arial')
    ylabel('Rain [mm]','FontName','Arial')
   t=t+24;
end
```

In order to visualize an exhaustive number of realizations of the process, weve reported above the value collected by six different gauges. We can observe there is some correlation in our graphs, for sure because the observed area is not so wide to return much uncorrelated profiles of water precipitation.

Task 3 - Radar Accuracy Analysis

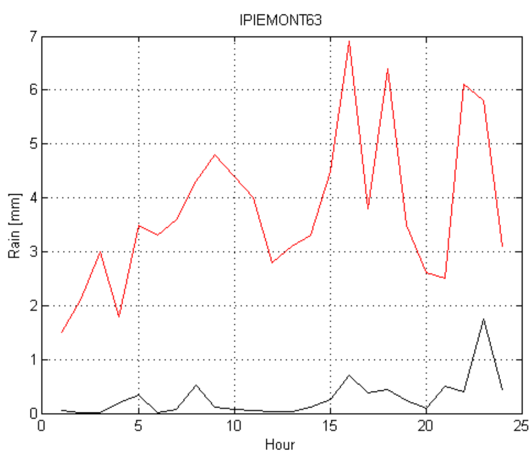
Superimpose in the same plot the hourly cumulated rain observations taken processing radar data in the pixel where the rain gauge is.

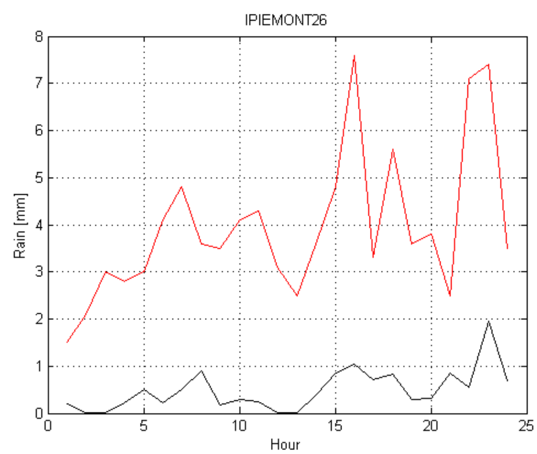
We report below, the map of gauges locations in order to evaluate their position in relation to meteorological conditions and using the code reported we evaluate the cumulated rain for specific point of gauges location checking how much precise the radar sensing is.




```
R=0;
R_mat=zeros(7,24);
t=3;
h = waitbar(0,'Wait please');
for gau=1:2
for hour=1:24
    waitbar(hour/24,h)
    R=0;
for i=t+60*(hour-1):(60*hour)+t-1
```

```
    %h = waitbar(i,'summing');
    DN=imread(strcat(rdpath,filelist(i).name),'png');
    ZdBz=(double(DN(gauge(gau).x_cord,gauge(gau).y_cord))./2.55-100)+91.4;
    Zmm_m=10.^(ZdBz./10);
    R=R+(double(Zmm_m)./316).^(2/3);
end
    R_mat(gau,hour)=R/60;
end

end
close(h);
figure('Name','Rain by Gauges')
for i=1:2

    %subplot(3,3,i)
    figure('Name',gauge(i).name)
    plot((R_mat(i,:)),'k')
    hold on;
    plot(G_Mat(i,:),'r')
    grid on;
    title(gauge(i).name)
    xlabel('Hour')
    ylabel('Rain [mm]')
end
```





As we can see, the result is not very precise. There are several reason for the consistent difference from radar values and the gauges ones and one of this is proper of precipitation nature. Not all the water falling at 1 km of altitude falls exactly in the same position it has seen before and above all most of that water doesn't fall at all because it can evaporate until to touch the ground.

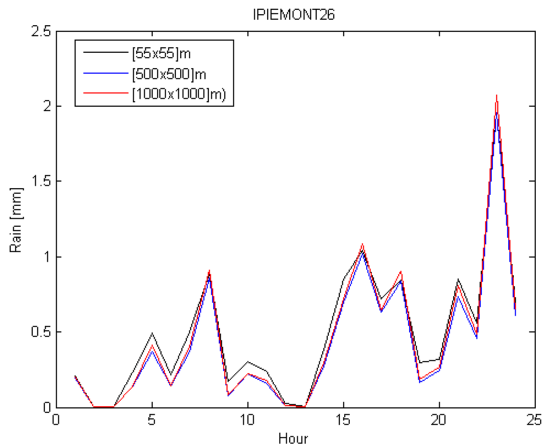
Task 4 - Spatial Average on Radar Data

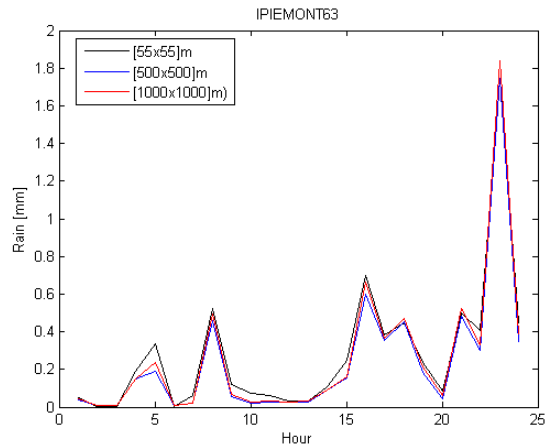
Evaluate the effects of spatial-averaging the radar data considering, for each point where each rain gauge is, the hourly cumulated rain averaged on an area of 9x9 (500m x 500m) and 17x17 (1km x

1km) pixels around the rain gauge. Superimpose the corresponding spatial-averaged hourly cumulated rain time series over each previous plot.

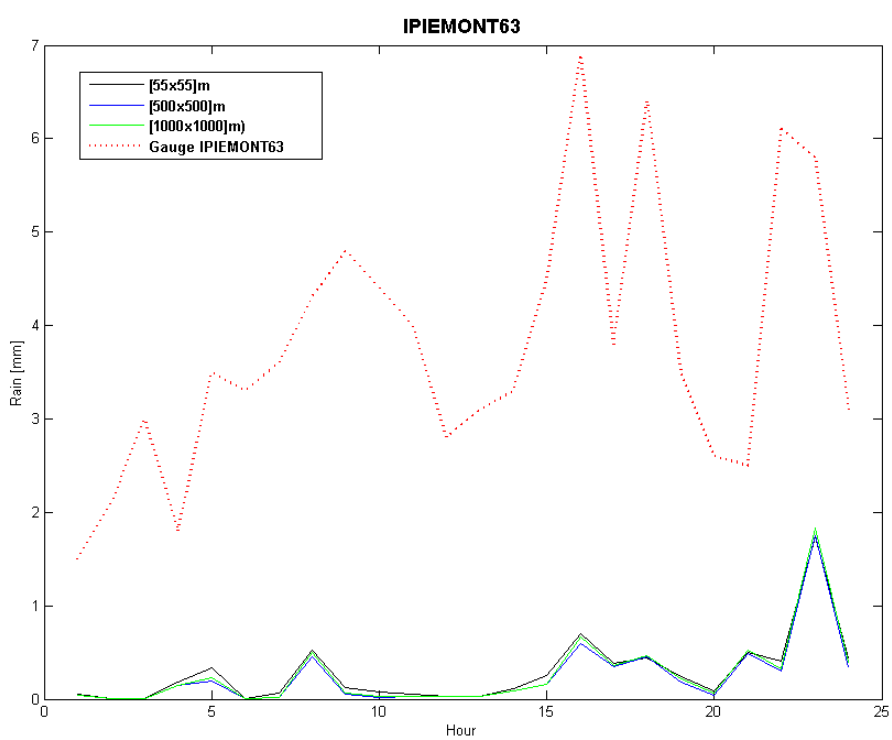
Using the same algorithm as previous task, we extract from each picture generated by radar, the mean value of an area's amount of rain felt. We do this in order to improve the accuracy of radar measurement and so we report the plot where each different mean is superposed on original single-pixel extraction of the two previous gauges.

```
% Mean Rain on surface area 17x17
R=0;
R_mat2=zeros(7,24);
t=3;
h = waitbar(0,'Wait please');
for gau=1:3
for hour=1:24
    waitbar(hour/24,h)
    R=0;
    cum=0;
for i=t+60*(hour-1):(60*hour)+t-1
    %h = waitbar(i,'summing');
    mean_rain=0;
    DN=imread(strcat(rdpath,filelist(i).name),'png');
    for row=-8:8
        for col=-8:8
    cum=cum+(double(DN(gauge(gau).x_cord+row,gauge(gau).y_cord)+col));
        end
    end
    cum=cum/289;
    ZdBz=(cum./2.55-100)+91.4;
    Zmm_m=10.^(ZdBz./10);
    R=R+(double(Zmm_m)./316).^(2/3);
end
    R_mat2(gau,hour)=R/60;
end
end
```





We can easily observe that averaging the spatial distribution changes in non-considerable way the radar data distribution and this is because the radar capacity of detection is under-estimated in relation to real rain-fall rate. A solution on this non-matching should be the adoption of a different conversion equation; we have to remind that conversion of radar echoes, due to rain in effectively valid rain fall rate on earth surface, is not a closed form procedure but it is empirically derived.
In the following plot we can see the effect of space averaging on real accuracy of gauge.



While we are shure that what is collected by gauges is really water fallen to the ground, it is not so predictable that amount detected by radar echoes is strictly linked to rain drops. Radar is calibrated in order to get information about perturbation but rain is typically complex weather characterized by very differentiated distribution of drops dimensions.