ECE710 Space Time Coding For Wireless Communication

Zhirong Li
Electrical & Computer Engineering Department
University of Waterloo, Waterloo, ON, Canada
z32li@engmail.uwaterloo.ca

Abstract

This manuscript is intended for providing solutions to mobile fading channel simulations. The first part gives problem formulation and requirements of the simulators. Section 2 will give a detail description of Time-Domain and Frequency-Domain method to simulate mobile fading channels. Section 3 will present the main steps in the simulation and the choices of key parameters. Section 4 shows the simulation and verification results based on theory. In the final part, a conclusion will be drawn.

Index Terms

Wireless, Fading Channel, Time Domain Method, Frequency Domain Method, Power Spectrum Desensity,

I. Simulation Requirements

LAND mobile fading channel simulator- In this problem, I will create a computer program to generate realizations of a Rayleigh fading process based on the frequency domain method and time domain method described in the class. In order to test the simulation program, generate a realization of a fading process where the maximum Doppler rate is 100Hz. Then the following results will be presented:

a) Plot the power spectral density of the fading signal.
b) Plot a single realization of the amplitude and phase process using a sampling rate of 10 kHz for a time span of 100 msec.
c) Calculate the average rate (in crossings per second) at which the fading amplitude crosses the level 10 rms X (10 dB below its rms level) in the positive going direction. Compare it with its theoretical value given in the class.
d) Calculate what fraction of the time the amplitude of your fading process is more than 10dB below its rms value. That is if X is the amplitude of the complex random process, what fraction of the time is

$$10\log\left(\frac{X^2}{X_{rms}^2}\right) \quad < \quad -10$$

Then measure this quantity based on the simulator and compare it with the theoretical value obtained in the class.

I will also create a computer program to generate realizations of a Rayleigh fading process based on the time domain method. Start by creating a third order filter as described in the class. Use the same parameters as given in Frequency Domain Method. Then present the following results:

a) Plot its autocorrelation function.
b) Repeat b, c, d from Frequency Domain method requirements.

II. Previous Work

A. Frequency Domain Method

Firstly we should define the intended realization of the mobile fading channel's observation window in time domain. i.e., represented as $x(t), t \in (0,T)$. By reproducing $x(t)$ every T seconds to generate a period signal which has a fourier series representation of the repeated signal $\tilde{x}(t) = \sum_{n=-\infty}^{n=\infty} x(t-nT)$ and PSD.

The fourier transform of $\tilde{x}(t)$ is given by

$$\tilde{x}(t) \quad = \quad \sum_k X_k e^{jk\omega_o t} \tag{1}$$

and power spectrum density function (PSD) is given by

$$S_{\tilde{x}\tilde{x}}(f) \quad = \quad \sum_k \sigma_k^2 \delta(f - kf_0) \tag{2}$$

Your name is with xyz Department…

where $\sigma_k^2 = E\left\lfloor |X_k|^2 \right\rfloor$.

However the σ_k^2 can be chosen to shape aribitrary shape of desired PSD. In Modeling of Wireless Channels slice 21, we get the PSD of the Rayleigh fading channel is

$$S_{xx}(f) = \frac{P_x}{\pi}\frac{1}{f_d}\frac{1}{\sqrt{1-(f/f_d)^2}}, |f| \le f_d \tag{3}$$

where f_d is the maximum doppler spread.

By selecting σ_k^2 taking form of $\frac{1}{\sqrt{f_d^2-(kf_0)^2}}$ to match the bathtub shape and let $\sum_k \sigma_k^2 = 1$ to satisfy the unit power constraints.

B. Time Domain Method

Time domain method is supposed to create desired channel simulator by carefully designing a filter. We generate white Gaussian process with zero mean and unit variance to pass through the filter, the autocorrelation of the output will be denoted as

$$\Phi_{XX}(\tau) = \Phi_{NN}(\tau) * h(\tau) * h(-\tau) = h(\tau) * h(-\tau) \tag{4}$$

The rayleigh fading channel's autocorrelation function is well known to be $J_0(2\pi f_d \tau)$, the central point of time domain method is to carefully design a filter whose autocorrelation function takes form as $J_0(2\pi f_d \tau)$ as much similar as possible. If we use 3 order IIR filter for simulation, it can be constructed as one 1-order and one 2-order filter's concatenation.

$$H_1(s) = \frac{\omega_0}{s+\omega_0}, H_2(s) = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2}$$

and

$$H_3(s) = \frac{\omega_0^3}{s^3 + (2\xi\omega_0 + \omega_0)s^2 + (\omega_0^2 + 2\xi\omega_0^2)s + \omega_0^3} \tag{5}$$

The continous time filter can be mapped to discrete time filter by this transform

$$H_3(z) = H_3(s)|_{s=2f_s\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

For 3-order transfer function

$$H_3(z) = \frac{X(z)}{N(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}$$

The coefficients are listed as follows

$$b_0 = b_3 = \omega_0^3, b_1 = b_2 = 3\omega_0^3$$

$$a_0 = 8f_s^3 + \lambda_1 4 f_s^2 + \lambda_2 2 f_s + \omega_0^3$$

$$a_1 = -24 f_s^3 - \lambda_1 4 f_s^2 + \lambda_2 2 f_s + 3\omega_0^3$$

$$a_2 = 24 f_s^3 - \lambda_1 4 f_s^2 - \lambda_2 2 f_s + 3\omega_0^3$$

$$a_3 = -8 f_s^3 + \lambda_1 4 f_s^2 - \lambda_2 2 f_s + \omega_0^3$$

$$\lambda_1 = \omega_0(1+2\xi), \lambda_2 = \omega_0^2(1+2\xi)$$

The 3-order filter can be realized as 1-order and 2-order system's concatenation.

$$\begin{aligned} h_3(t) &= \int_{-\infty}^{\infty} e^{-\alpha(t-\tau)} u(t-\tau) e^{-\alpha\tau}\sin(\beta\tau)u(\tau)d\tau \\ &= -\frac{e^{-\alpha t}u(t)(-1+\cos\beta t)}{\beta} \end{aligned} \tag{6}$$

Thus corresponding autocorrelation function would take form as (Verified by Maple 9.5)

$$\begin{aligned} h_3(t) * h_3(-t) &= \int_{-\infty}^{\infty} h_3(t+\tau)h_3(\tau)d\tau \\ &= \frac{e^{(-\alpha|t|)}(2\beta^2 + \cos(|t|\beta)\beta^2 + 2\alpha^2 + 3\beta\sin(|t|\beta)\alpha - 2\cos(|t|\beta)\alpha^2}{4\alpha(4\alpha^2+\beta^2)(\alpha^2+\beta^2)} \end{aligned} \tag{7}$$

The autocorrelation in Fig.1 is drawn by taken $\alpha = 2\pi\xi f_0, \beta = 2\pi f_0\sqrt{1-\xi^2}$, here $\xi = 0.175, f_0 = 10$Hz. The discrete autocorrelation function is calculated by inverse Z transform of $H_3(z)$ to get $h(n)$, then compute $h(n) * h(-n)$.

III. Methodology

A. Frequency Domain Method

a) The first step of frequency domain method is to determine the number of the frequency component to represent the bathtub-shaped PSD. Herein we have $f_d = 100$ Hz, $f_s = 10$ kHz, $T = 100$ ms, $f_0 = 10$ Hz. Thus $M = \left\lfloor \frac{f_d}{f_0} \right\rfloor = 10$. We need to generate $2M + 1 = 21$ iid. complex Gaussian random variables with

$$E[X_k] = 0, E[|X_k|^2] = \sigma_k^2 = \frac{\beta}{\sqrt{f_d^2 - (kf_0)^2}} = \frac{\beta}{\sqrt{(Mf_0)^2 - (kf_0)^2}} \tag{8}$$

The slope at $(M-1)f_0$ can be denoted as $\frac{(M-1)\beta}{f_0^2(2M-1)^{2/3}}$, thus the truncated value at Mf_0 should be

$$\sigma_M^2 = \frac{\beta}{f_0\sqrt{2M-1}} + \frac{(M-1)\beta}{f_0(2M-1)^{3/2}} = \frac{\beta(3M-2)}{f_0(2M-1)^{3/2}} \tag{9}$$

Thus

$$\sum_{k=-M}^{k=M} \sigma_k^2 = 2\left(\sum_{k=1}^{M-1} \frac{\beta}{\sqrt{(Mf_0)^2 - (kf_0)^2}} + \frac{\beta(3M-2)}{f_0(2M-1)^{3/2}}\right) + \frac{\beta}{Mf_0}$$

In order to unify the output power, we should choose $\beta = 3.221615$. We can use Matlab's build-in function of *randn* to generate zero mean, unit variance complex Gaussian distributed random variable. Here we generate $2M + 1 = 21$ iid. Gaussian distributed random variables with variance σ_k^2 respectively.

b) The sencond step is to sum up the complex random variables to form a random process for time interval $(0, T)$.

$$X(t) = \sum_{k=-M}^{k=M} X_k e^{j2\pi k \frac{t}{T}} \tag{10}$$

Then we can sample at $f_s = 10$kHz to get the desired $T/T_s = f_s/f_0 = 1000$ samples.

$$X[i] = X(iT_s) = \sum_{k=-M}^{k=M} X_k e^{j2\pi \frac{ikT_s}{T}} \tag{11}$$

B. Time Domain Method

a) Here we have $f_d = 100$Hz, $f_s = 10$kHz. The first step of time domain method is to determine the ARMA$(3, 3)$ system function. Substitute $\omega_0 = 2\pi f_d/1.2$ and $\xi = 0.175$ to the equations representing coefficients of numerator and denominator.

b) Inverse Z transform of $H_3(z)$ to get the time domain filter coefficients $h(n)$. We define n_0 as $\frac{|h(n_0)|}{\max_n |h(n)|} \le 0.01$, we will truncate infinite length IIR filter to FIR. By substituting ω_0 ξ into the coefficients representation formular we get:

$$+0.1328287684e{-}1{*}I{*}(.9895688730{-}0.5104903619e{-}1{*}I)^n{-}0.1585969046e{-}1{*}(.9895688730{+}0.5104903619e{-}1{*}I)^n$$

$$-0.1328287684e - 1 * I * (.9895688730 + 0.5104903619e - 1 * I)^n$$

, by checking Fig.3., the desired n_0 would around 230.

c) Normalize $H(z)$ by let $\beta_2\sqrt{\sum_{n=0}^{n=1000} h^2(n)} = 1$ to have unit power or unify the autocorrelation sequence $h(n) * h(-n)$ at $n = 0$, we get $\beta_2 = 23.2838$.

d) Creat an input sequence with zero mean unit variance complex Gaussian distribution, the length is n_0 and run the input to the filter to get rid of any transients by using $[\text{YTrans}, \text{Zi}] = \text{filter}(B, A, \text{XTrans})$.

e) Generate the input sequence with number of 100ms $\times$ 10kHz =1000 zero mean unit variance complex Gaussian variables, run through the filter and take the output as a single realization of the fading process. Note that we will use matlab built-in filter function in way like: $[Y, \text{Zf}] = \text{filter}(\beta_2 B, A, X, \text{Zi})$.

C. Test of Fading Simulators

Theoretically speaking, the average number of times per second that the magnitude $\alpha(t)$ of the fading process crosses some threshold ρ/a_{rms} in positive going direction satisfy the following equation:

$$N_R = \sqrt{2\pi} f_d \rho e^{-\rho^2}, a_{rms} = \sqrt{E\left[a^2\right]} \tag{12}$$

The average period of time for which the received signal remains below a specified level ρ/a_{rms} should be

$$\lambda = \frac{e^{\rho^2} - 1}{\sqrt{2\pi}\rho f_d} \tag{13}$$

In this simulation, the $a_{rms} = 1$Watt, ρ would be $1/\sqrt{10}$, thus $N_R = 71.72$ and $\lambda = 0.13268\%$.

In my simulations, we sample at f_s thus only get discrete sampling points. We define positive going accross predefined threshold occurs iff. $\alpha(i) <$ Threshold, $\alpha(i+1) \geq$ Threshold. The average fade duration is calculated as consecutive sampling points whose $\alpha(i) <$ Threshold.

IV. Results

A. Simulation Results

Figure 2 shows the frequency domain method with $X_k$changing simutaneously at every time instant and time domain without eliminating filter delay's impact on the output.We can see that for frequency domain method, the frequency component is Gaussian distributed with corresponding variance, the values changes very quickly, actually I just draw a 1000 point sampling of a rayleigh distributed random process. Thus we only need generate X_K once for the frequency domain method. For the time domain method, we need to eliminate any transients.

The filter coefficients for its first 1000-taps after inverse Z transform come out to be shown in following figure 3:

It tells us we can estimate the filter if realized by FIR, the filter tap may as well deem as 230-tap FIR filter if define n_0 as $\frac{|h(n_0)|}{\max_n |h(n)|} \leq 0.01$. In my simulation, I choose n_0 to be 600.

The output of frequency domain method and time domain method's autocorrelation and corresponding PSD are drawn in the following figure 4.

From above figure, we can clearly see both method get a similar performance.

The matlab code for these two simulations are listed as follows:

```
function FadingCH_simulator
clear all;
close all;
format long;
fd=100; % Doppler Rate
fs=10000; % Sampling Rate
T=0.1; % Observation Interval
f0=10; %  Base Frequency
beta=1; % Normalized Factor
M=floor(fd/f0); % Rays in Positive Axis
num_sample=floor(T*fs); % Number of samples in the observation interval.
jj=sqrt(-1);
X=zeros(1,2*M+1); %Gaussian distributed frequency component
EX=zeros(1,2*M+1); % Expectations of sigma_k^2
Y=zeros(1,num_sample+1); %Output of a single realization of rayleigh fading channel
%Expectations of sigma_k^2
EX(M+1:2*M)=1./sqrt(fd^2-((0:M-1)*f0).^2);
EX(2*M+1)=(3*M-2)/(f0*(2*M-1)^(1.5));
EX(1:M)=EX(2*M+1:-1:M+2);
beta=1/sum(EX);
EX=beta*EX;
%Generate Gaussian Random Variables with variance of sigma_k^2.
X=sqrt(1/2)*(randn(1,2*M+1)+jj*randn(1,2*M+1));
X=X.*sqrt(EX); % Normalize to desired variance
for i=1:num_sample+1
        Y(i)=sum(X.*exp(jj*2*pi*(i-1)*f0/fs*(-M:1:M)));
end
RMS_Y=20*log(sqrt(sum(Y.*conj(Y))/size(Y,2)));
omega0=2*pi*fd/1.2; %Filter Parameters
xi=0.175; %Filter Parameters
b0=omega0^3;
b3=omega0^3;
b1=3*omega0^3;
b2=3*omega0^3;
```

```
lambda1=omega0*(1+2*xi);
lambda2=omega0^2*(1+2*xi);
a0=8*fs^3+lambda1*4*fs^2+lambda2*2*fs+omega0^3;
a1=-24*fs^3-lambda1*4*fs^2+lambda2*2*fs+3*omega0^3;
a2=24*fs^3-lambda1*4*fs^2-lambda2*2*fs+3*omega0^3;
a3=-8*fs^3+lambda1*4*fs^2-lambda2*2*fs+omega0^3;
B=[b0 b1 b2 b3];
A=[a0 a1 a2 a3];
TransientXIn=sqrt(1/2)*(randn(1,250)+jj*randn(1,250));
[TransientYOut,Zi]=filter(B,A,TransientXIn);
XIn=sqrt(1/2)*(randn(1,num_sample+1)+jj*randn(1,num_sample+1));
[YOut,Zf]=filter(B,A,XIn,Zi);
RMS_YOut=20*log(sqrt(sum(YOut.*conj(YOut))/size(YOut,2))); %RMS Calculation
YCount=0;
YOutCount=0;
for i=1:num_sample
    if (20*log(abs(Y(i)))<RMS_Y-10)&(20*log(abs(Y(i+1)))>RMS_Y-10)
        YCount=YCount+1;
    end

    if (20*log(abs(YOut(i)))<RMS_YOut-10)&(20*log(abs(YOut(i+1)))>RMS_YOut-10)
        YOutCount=YOutCount+1;
    end
end
YPeriod=0;
YOutPeriod=0;
i=1;
while (i<=(num_sample+1))
    if (20*log(abs(Y(i)))<RMS_Y-10)
        j=i;
        while(i<=(num_sample+1))&(20*log(abs(Y(i)))<RMS_Y-10)
            i=i+1;
        end
        YPeriod=YPeriod+i-j;
    else
        i=i+1;
    end
end
i=1;
while (i<=(num_sample+1))
    if (20*log(abs(YOut(i)))<RMS_YOut-10)
        j=i;
        while (i<=(num_sample+1))&(20*log(abs(YOut(i)))<RMS_YOut-10)
            i=i+1;
        end
        YOutPeriod=YOutPeriod+i-j;
    else
        i=i+1;
    end
end
figure(1);
grid on;
plot((0:num_sample)/num_sample*0.1,20*log(abs(Y)),'-');
xlabel(['Time [sec] Level Cross Rate ' num2str(YCount) '/1001 times'
'Average Fade Duration ' num2str(YPeriod/1001*100) '%']);
ylabel('Magnitude [dB]');
title('Frequency Domain Method: A single realization of fading channel amplitude');
figure(11);
```

```
grid on;
plot((0:num_sample)/num_sample*0.1,20*log(abs(YOut)),'-');
xlabel(['Time [sec] Level Cross Rate ' num2str(YOutCount) '/1001 times'
 'Average Fade Duration 10dB below RMS ' num2str(YOutPeriod/1001*100) '%']);
ylabel('Magnitude [dB]');
title('Time Domain Method: A single realization of fading channel amplitude');
```

V. Conclusions

In my fading channel simulator, both frequency domain method and time domain method work and output corresponding single realization of rayleigh fading channel. The autocorrelation and corresponding PSD are drawn in Fig.4. The verification purpose of cross level rate and average fade duration is calculated and further verifications needed to be done.

References

[1] Prof. Murat Uysal, “ECE710 Space-time Coding for Wireless Communication handouts”, Spring, 2004

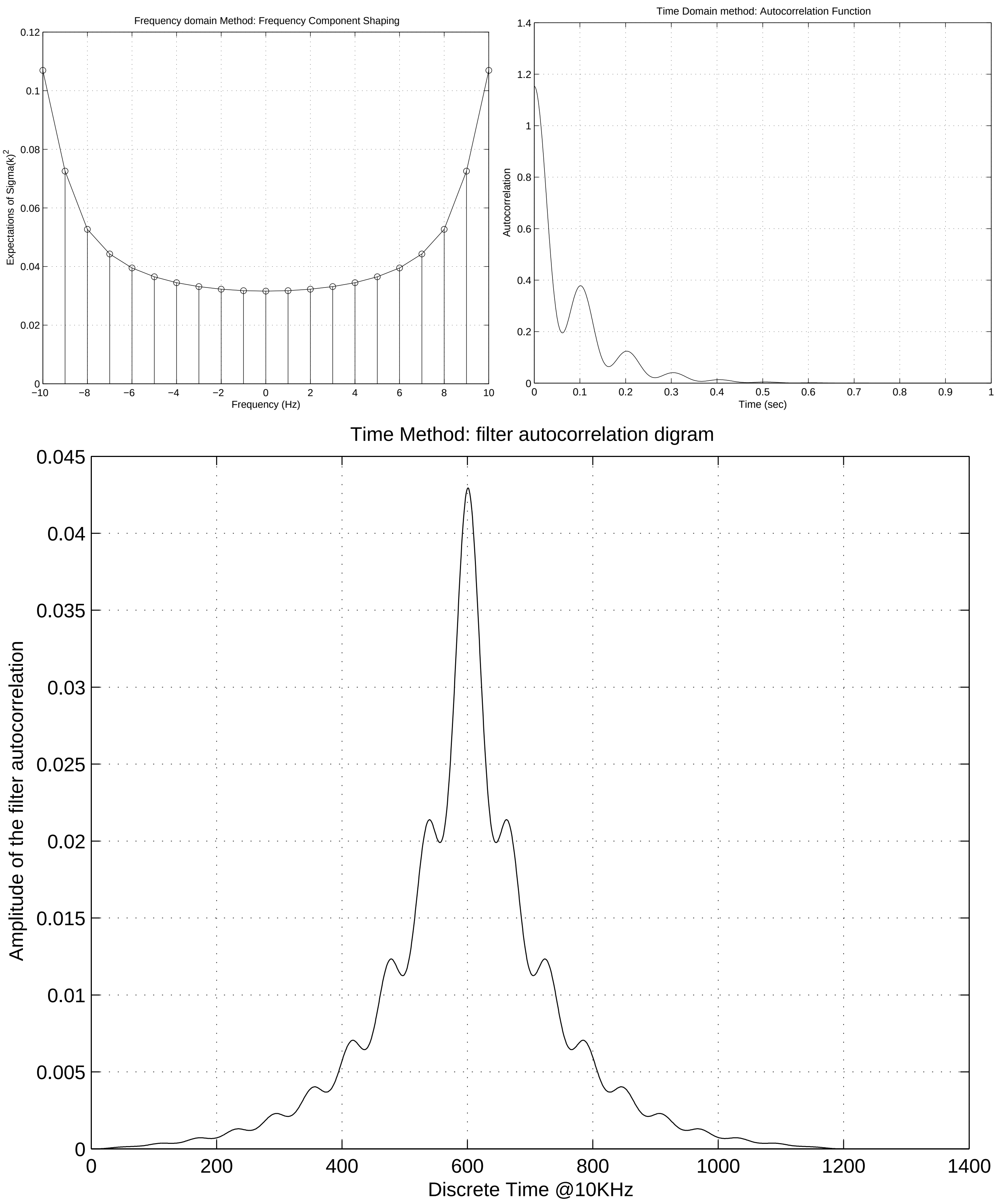

Fig. 1. PSD of Periodic Signals AutoCorrelation of 3-order Filter (Continous Time and Discrete Time)

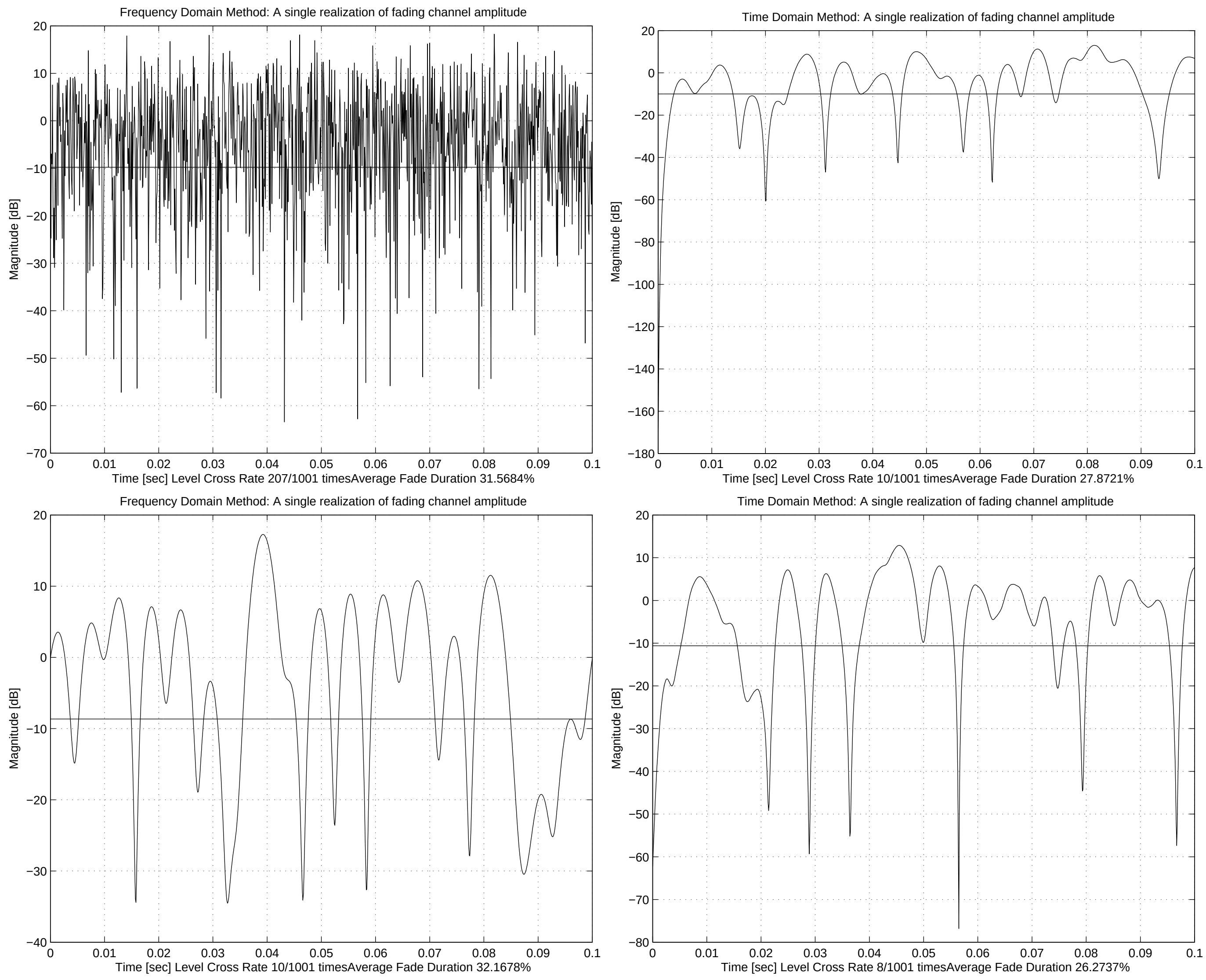

Fig. 2. Frequency Domain Method With X_kChanging with time Time Domain Method without eliminating initial transients.
Frequency Domain Method with X_k generating once and Time Domain Method with elliminating Transients.

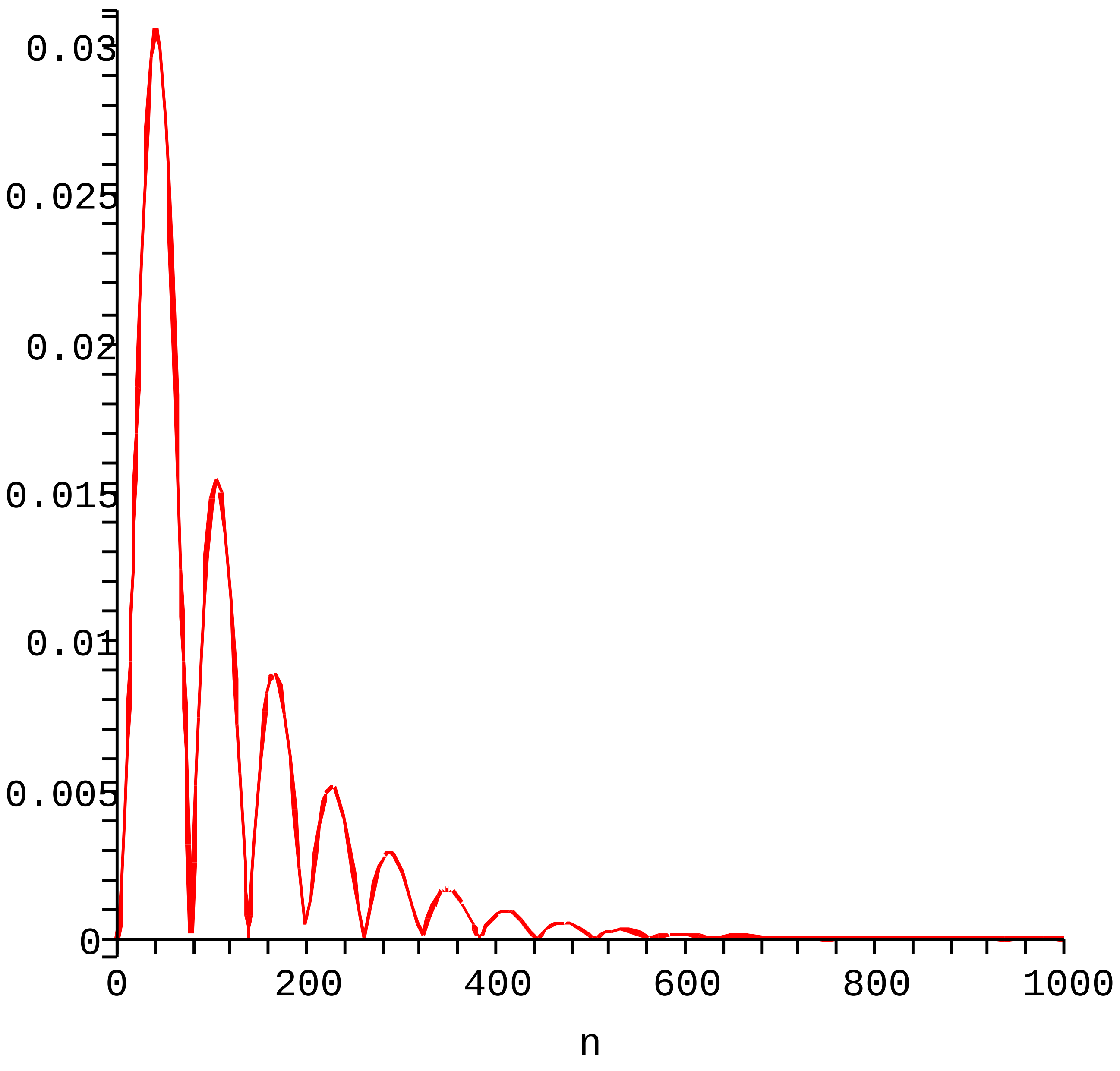

Fig. 3. Filter Coefficients for First 1000 tap (ABS Value)

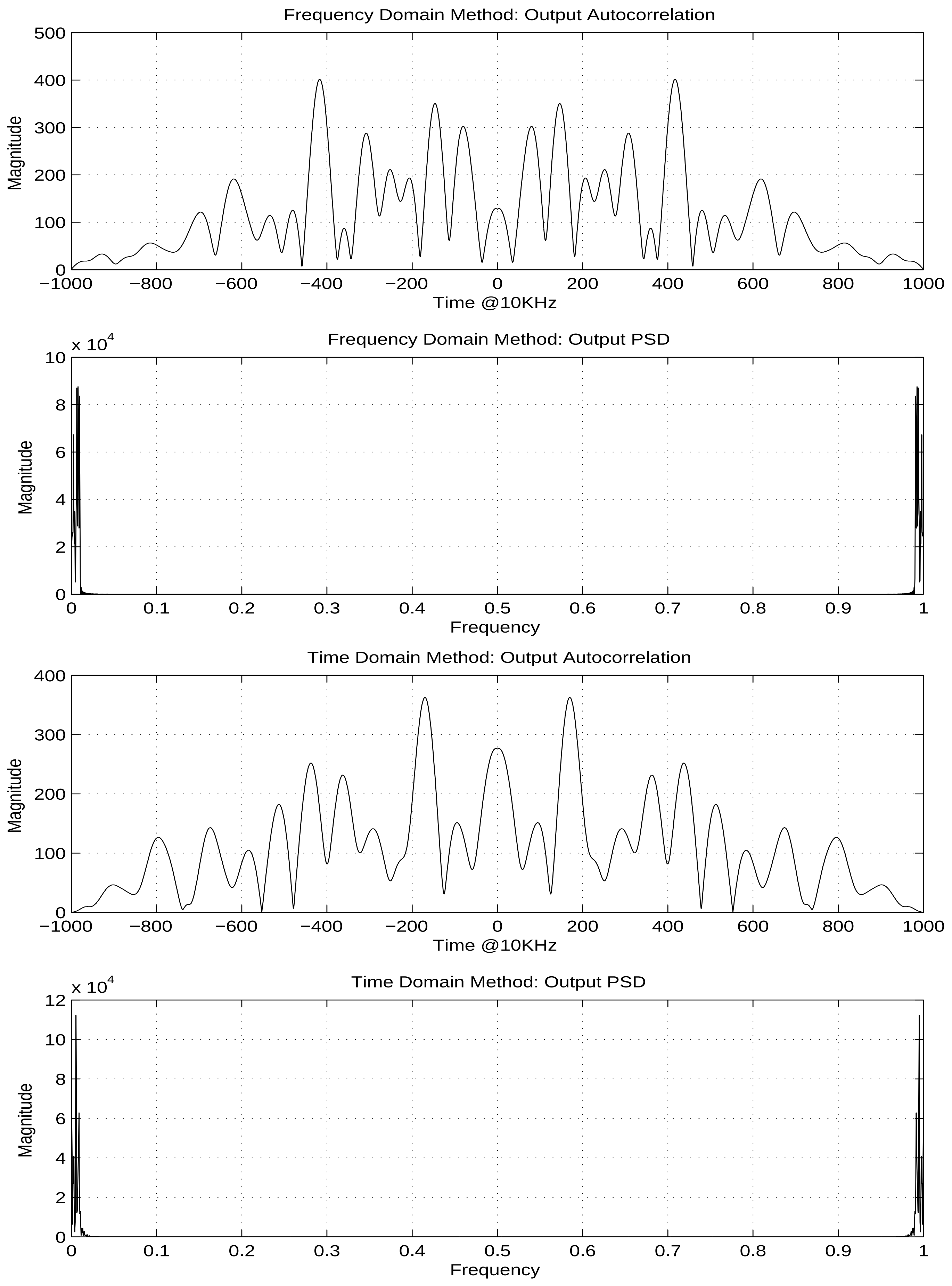

Fig. 4. Frequency and Time Domain Method: Ouput's autocorrelation and PSD