**EE-5283-002**

**Spread Spectrum Communication**

**&**

**GPS (Global Positioning System)**

**Project: 2**

**Submitted By**

**Name:**  Nimish Joshi

**Banner Id:**  @01482152

* **Introduction of Acquisition:**

Acquisition refers to the period of time when conditional responses first appears and increase with the frequency.

Acquisition of Global Positioning System Signal composed of two dimensional search in a time /frequency space to synchronize a receiver generated reference signal with the received signal in coed phase and frequency. Side of the region can be searched by user uncertainty in code phase and frequency offset. Main aim of this part is to explore and use current signal detection concepts to quantify the time to achieve acquisition of the direct sequence code. Additionally it also mitigating the effect of parallel processioning shrinking acquisition time in the presence of initial (receiver) position, time, velocity, and frequency errors and/or uncertainties.

A two dimension uncertainty region characterizes uncertainties in career frequency and code phase. Frequency uncertainty are due to unknown Doppler shift and local oscillator drift, on the other side the uncertainties in shown clock time and receiver position dictate the span of code phase to be searched.

* **Part:1**

Considering the following Process.

* Taking signal that generated in previous project.
* Signal with shifted code phase and Doppler sinusoidal Modulation.
* Setting the code phase to 556 as given information
* Frequency f=1KHZ and
* Adding Additive White Gaussian Noise(AWGN)

After these inclusion we have the generated code as below:

**\*Code\***

% Program\*

%

% Project\_gold sequece.m

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Spreading code initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IPOINT = 5;

n=1023;

user = 1; % number of users

stage = 10; % number of stages

ptap1 = [3 10]; % position of taps for 1st

ptap2 = [2 3 6 8 9 10]; % position of taps for 2nd

regi1 = [1 1 1 1 1 1 1 1 1 1]; % initial value of register for 1st

regi2 = [1 1 1 1 1 1 1 1 1 1]; % initial value of register for 2nd

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Generation of the gold code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = goldseq(m1,m2,user);

code= code\*2-1;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end of gold sequence generation\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Perform hard decision coding

new\_code=code;

new\_code =repmat(new\_code,4,1); % Oversample signal(4sample per chip)

% Now input through the impulse response

new\_code = [new\_code,new\_code,new\_code,new\_code,new\_code];

new\_code=new\_code\*2-1;

%Pass the signal through correlator and doppler effect

ts=0:2.5e-7:5e-3;

ts=0:2.44e-7:5e-3;

f=1000; % frequency=1KHZ %

dop=exp(2\*pi\*i\*ts\*f); %Doppler Sinusoidal

for n=1:length(new\_code)

sampled\_code(n)=new\_code(n).\*dop(n);

end

code\_new=[sampled\_code(556:556+4092-1)]; %setting code phase to 556 %

code\_new=[code\_new code\_new code\_new code\_new code\_new];

figure(1)

plot(code\_new);

title('setting code phase to 556');

whitebg('w')

grid on;

% Now adding AWGN noise to the code %

code2=awgn(code\_new,-40);

figure(2)

plot(code2);

title('code with AWGN noise');

whitebg('c');

grid on;

code5=new\_code(1:4\*4092); % multiplying code with 4092 for 4 sample per chip

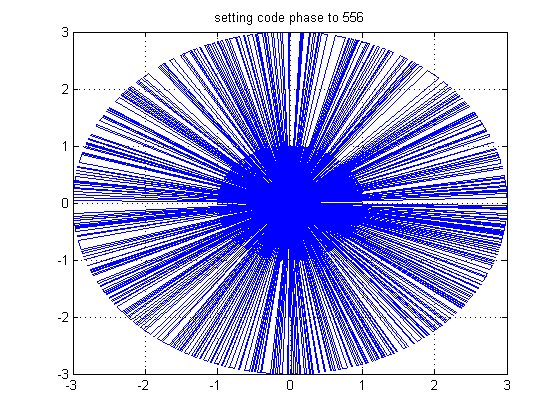
figure(3)

plot(code5);

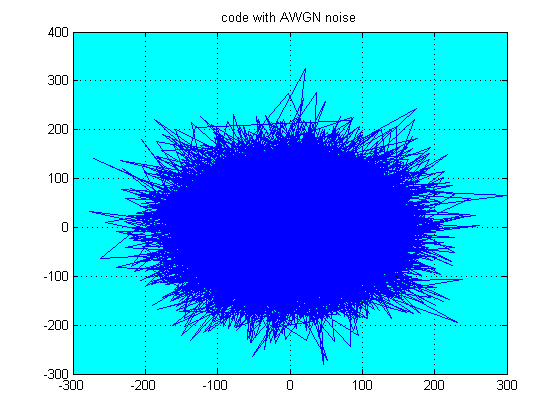
whitebg('w');

grid on;

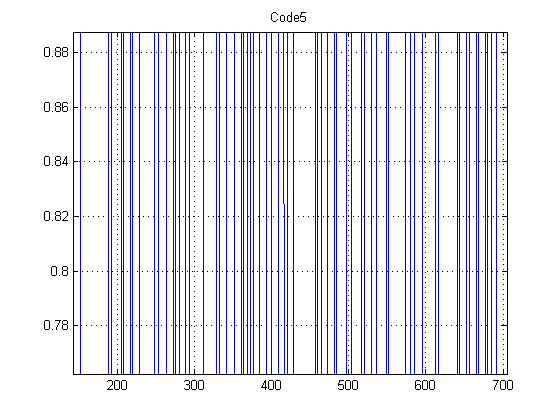
**\*Related Graphs\***

****

**Fig1: Doppler Sinusoidal code with code phase 556**



**Fig:2 Inserting AWGN noise.**

****

**Fig: 3: Code5**

* **Part:2**

Consider following steps in the final code

* Adding two for loop
* Multiplying signal to complex sinusoids.
* Using frequencies[-2kHZ, -1.5kHZ, -1kHZ, 0kHZ, 1kHZ, 1.5kHZ, 2kHZ

**\*CODE\***

% adding different frequencies and for loop below %

f=[-2000,-1500,-1000,-500,0,500,1000,1500,2000];

for m=1:9;

code\_dop(m,:)=exp(-2\*pi\*i\*ts\*f(m));

code1(m,:)=(code2(1:16368).\*code\_dop(m,1:16368))\*2-1;

for count2 = 1:4092

num1(m,:)=(code5.\*code1(m,:));

counter1=0;

for count = 1:8184

counter1=counter1+num1(m,count);

end

counter2=0;

for count = 8185:16368

counter2=counter2+num1(m,count);

end

counter(m,count2)=counter1\*conj(counter1)+counter2\*conj(counter2);

code5=new\_code(count2+1:count2+4\*4092);

end

end

figure(4)

plot(num1(m,:));

title('code after multiplying with complex sinusoidal to wipe off dopler modulation');

grid on;

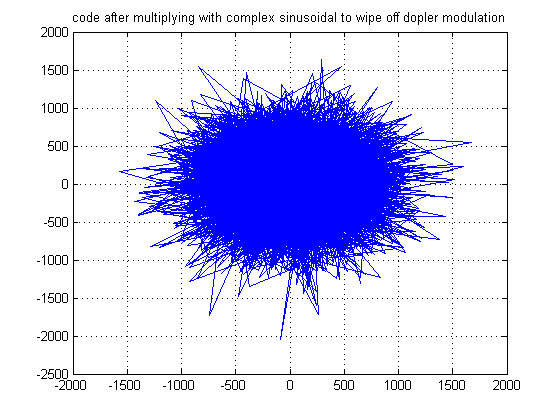
figure(5)

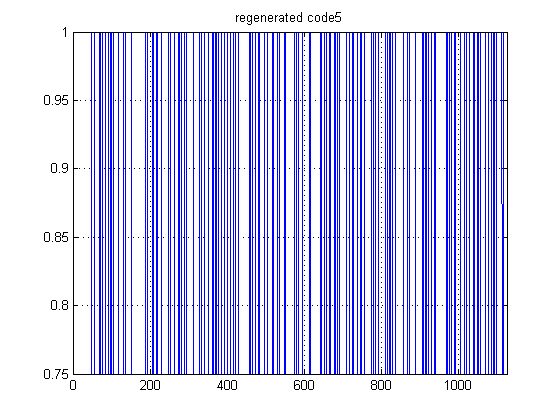
plot(code5);

title('regenerated code5');

grid on;

**\*Related Graphs\***





* **Part 3 & 4:**
* **Using Two Correlator Design Approach:**

Considering the following steps in these parts;

* Adding another loop;
* Using parallel set of correlator
* Using matched filter correlator
* Peak, when noise is added
* Peak when noise is’ 0’

**\*CODE\*:**

% Program\*

%

% Project\_gold sequece.m

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Spreading code initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

clc;

close all;

clear all;

IPOINT = 5;

n=1023;

user = 1; % number of users

stage = 10; % number of stages

ptap1 = [3 10]; % position of taps for 1st

ptap2 = [2 3 6 8 9 10]; % position of taps for 2nd

regi1 = [1 1 1 1 1 1 1 1 1 1]; % initial value of register for 1st

regi2 = [1 1 1 1 1 1 1 1 1 1]; % initial value of register for 2nd

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Generation of the gold code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = goldseq(m1,m2,user);

code= code\*2-1;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end of gold sequence generation\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Perform hard decision coding

new\_code=code;

new\_code =repmat(new\_code,4,1); % Oversample signal(4sample per chip)

% Now input through the impulse response

new\_code = [new\_code,new\_code,new\_code,new\_code,new\_code];

new\_code=new\_code\*2-1;

%Pass the signal through correlator and doppler effect

ts=0:2.5e-7:5e-3;

ts=0:2.44e-7:5e-3;

f=1000; % frequency=1KHZ %

dop=exp(2\*pi\*i\*ts\*f); %Doppler Sinusoidal

for n=1:length(new\_code)

sampled\_code(n)=new\_code(n).\*dop(n);

end

code\_new=[sampled\_code(556:556+4092-1)]; %setting code phase to 556 %

code\_new=[code\_new code\_new code\_new code\_new code\_new];

% figure(1)

% plot(code\_new);

% title('setting code phase to 556');

% whitebg('w')

% grid on;

% Now adding AWGN noise to the code %

code2=awgn(code\_new,0);

% figure(2)

% plot(code2);

% title('code with AWGN noise');

% whitebg('c');

% grid on;

code5=new\_code(1:4\*4092); % multiplying code with 4092 for 4 sample per chip

% figure(3)

% plot(code5);

% title('Code5');

% whitebg('w');

% grid on;

% adding different frequencies and for loop below %

f=[-2000,-1500,-1000,-500,0,500,1000,1500,2000];

for m=1:9;

code\_dop(m,:)=exp(-2\*pi\*i\*ts\*f(m));

code1(m,:)=(code2(1:16368).\*code\_dop(m,1:16368))\*2-1;

for count2 = 1:4092

num1(m,:)=(code5.\*code1(m,:));

counter1=0;

for count = 1:8184

counter1=counter1+num1(m,count);

end

counter2=0;

for count = 8185:16368

counter2=counter2+num1(m,count);

end

counter(m,count2)=counter1\*conj(counter1)+counter2\*conj(counter2);

code5=new\_code(count2+1:count2+4\*4092);

end

end

% figure(4)

% plot(num1(m,:));

% title('code after multiplying with complex sinusoidal to wipe off dopler modulation');

% grid on;

% figure(5)

% plot(code5);

% title('regenerated code5');

% grid on;

figure;

n = 1:4092;

f =-2000:500:2000;

[N,F]=meshgrid(n,f);

surf(N,F,abs(counter))

title('Parallel filter Peak');

%

%

%

%

% for f=-2000:500:2000

% dop=exp(2\*pi\*1i\*ts\*f);

% for n=1:556

% sampled\_code(n)=new\_code(n)\*dop(n);

%

% end

% end

%

% new\_code2=sampled\_code(128:512);

% new\_code3=awgn(new\_code2,10);

% parallel\_corr=abs(crosscorr(new\_code(1:256),new\_code2(1:256)));

% figure(1)

% %plot the 3D graph

% plot3(n,f,parallel\_corr)

%

% %matched filter

% ts=0:2.5e-7:5e-3;

% for f=-2000:500:2000

% dop=exp(2\*pi\*1i\*ts\*f);

% for n=1:556

% sampled\_code(n)=new\_code(n)\*dop(n);

%

% end

% end

% new\_code2=sampled\_code(128:512);

% new\_code3=awgn(new\_code2,10);

% matched\_corr=compconv2(new\_code(1:256),new\_code2(1:256),new\_code2(1:256));

% figure(2)

% %plot the 3D graph

% plot3(n,f,matched\_corr)

f=[-2000,-1500,-1000,-500,0,500,1000,1500,2000];

for m=1:9;

code\_dop(m,:)=exp(-2\*pi\*i\*ts\*f(m));

code1(m,:)=(code2(1:16368).\*code\_dop(m,1:16368))\*2-1;

code5=fliplr(new\_code(1:4092));

num1=conv(code1(m,:),code5);

sum1=num1(1:4092)+num1(4093:8184);

change2=num1(8185:12276)+num1(12277:16368);

count7(m,:)=abs((sum1.\*conj(sum1))+(change2.\*conj(change2)));

end

figure(7)

plot(code1(m,:));

figure(8)

plot(num1);

figure(9)

plot(sum1);

figure(10)

plot(change2);

figure(11)

plot(count7(m,:));

countfinal=fliplr(count7);

figure;

n = 1:4092;

f =-2000:500:2000;

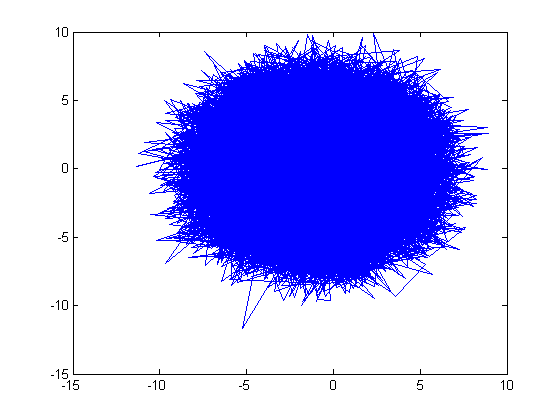
[N,F]=meshgrid(n,f);

surf(N,F,abs(countfinal));

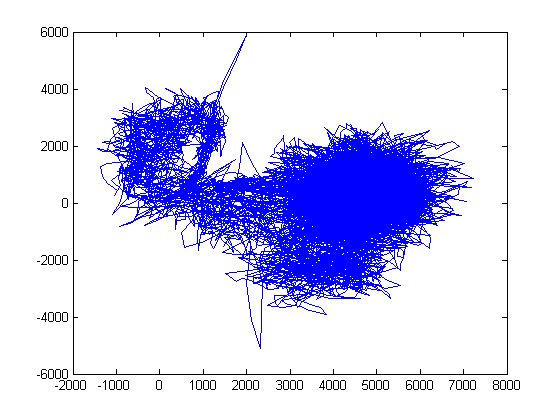
%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end of the file\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*%

**RELATED GRAPHS:**

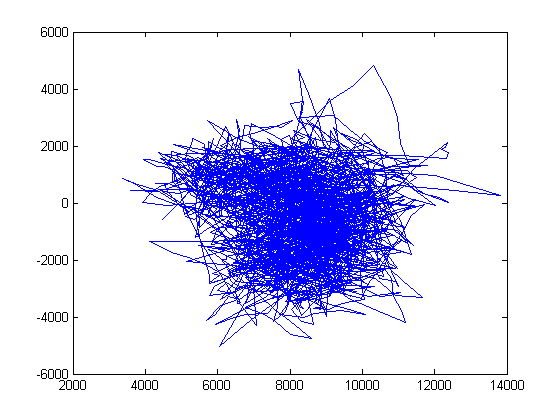
1. **When noise is not added:**

****

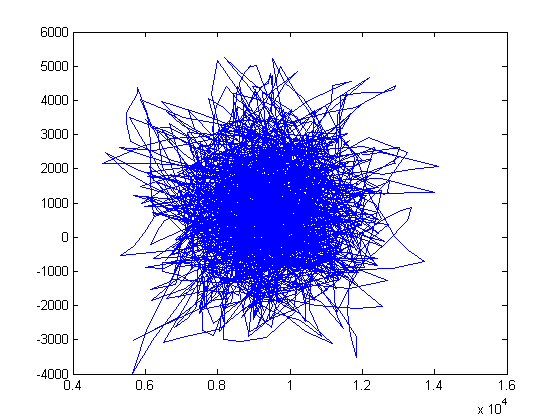
**Figure: code1(m,:)**

****

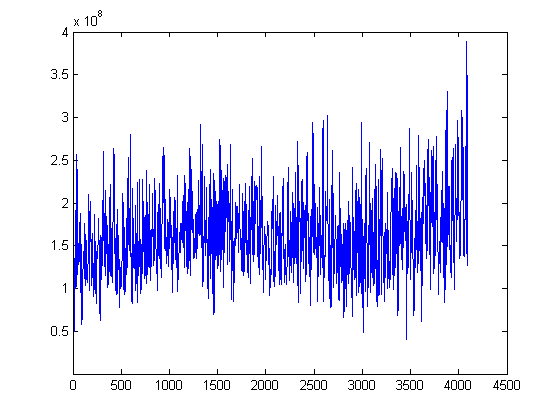
**Figure: num1**

****

**Figure: sum1**

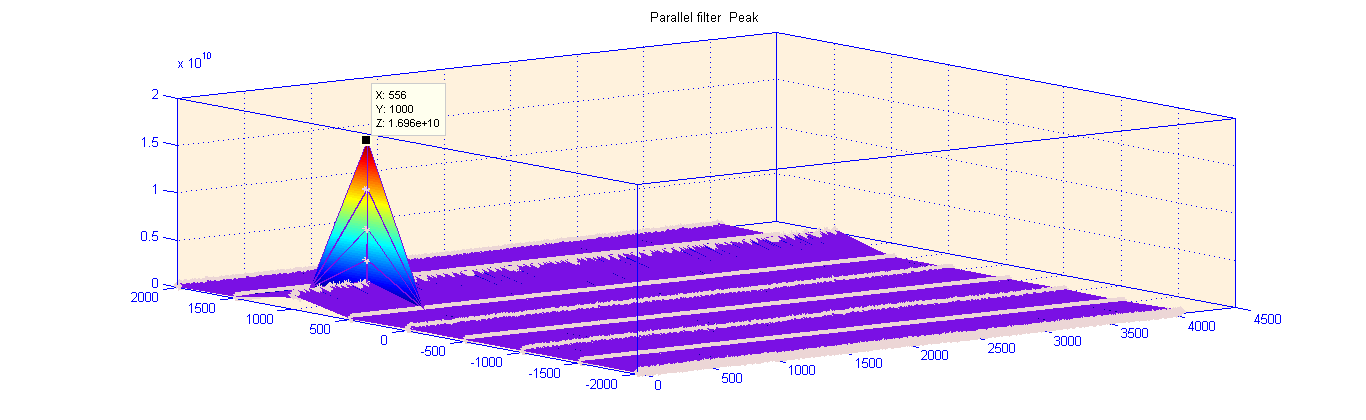
****

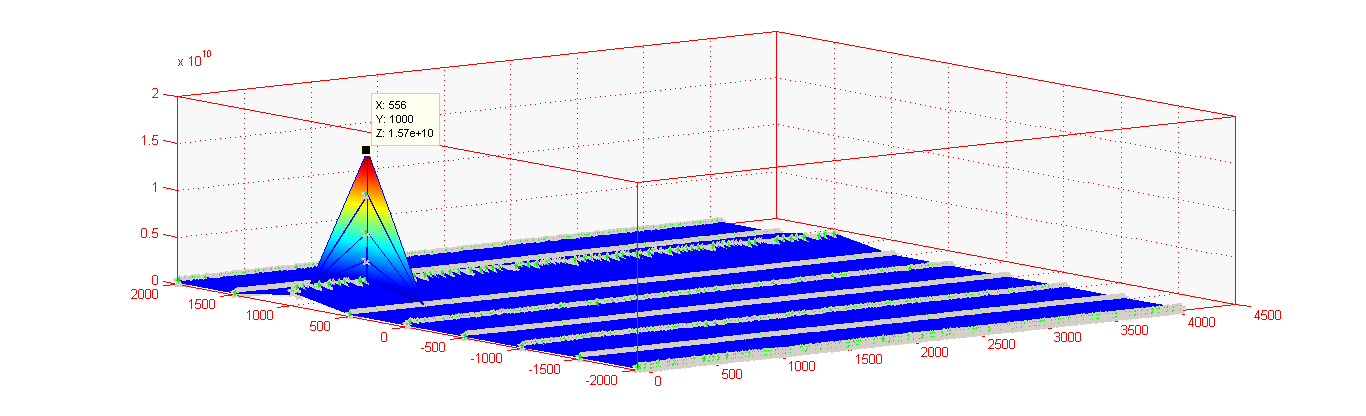
**Figure: change2**

****

**Figure: count(7m, :)**

**Figure: parallel filter peak value at f=1000hz**

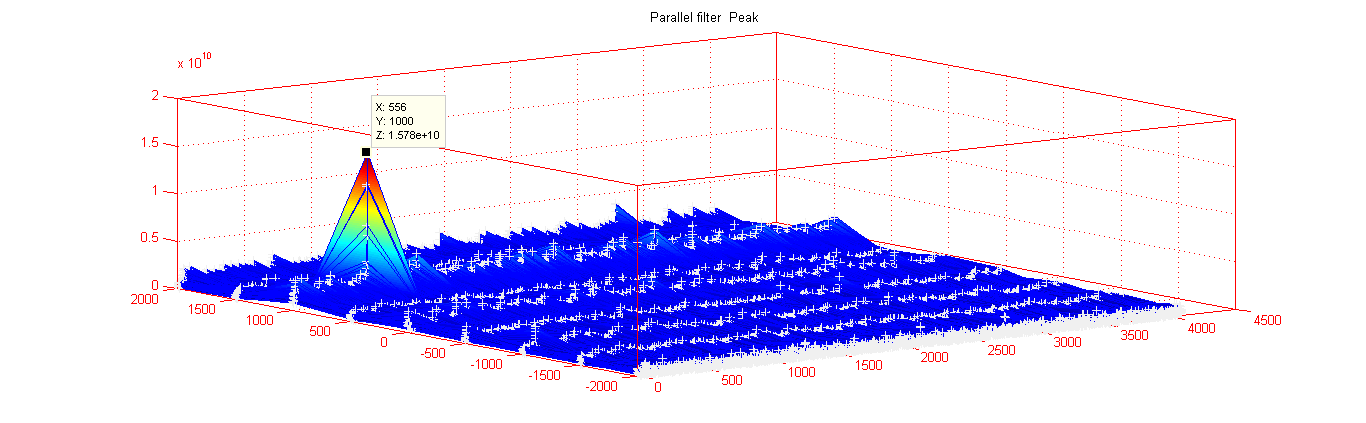
****

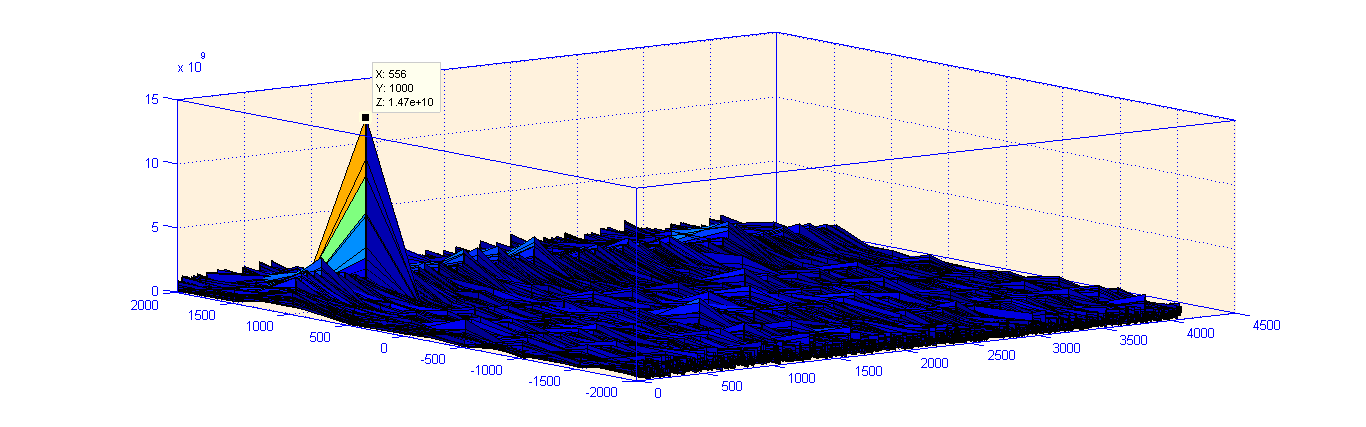
****

**Figure: matched filter peak value at f=1000hz**

**2.When Noise is added:**

**Figure: parallel filter peak value at f=1000hz**

****

****

**Figure: matched filter peak value at f=1000hz**