

**MATLAB SIMULATIONS FOR IRNSS AQUISITION AND TRACKING**

**Submitted to**

**Mr. Satish Dhawan**

**Deputy General Manager**

**Ananth Technologies LTD.**

**During the Training period: (3-May-2019 to 1-July-2019)**

**Submitted By**

**Vishnu Vytla**

**(17071A04I0)**

**Dept. Of Electronics and Communication Engineering**

**VNR VJIET**

# CERTIFICATE

This is to certify that  **Vishn Vyla(17071A04I0)** , student of *IIth year, B.Tech.* from *VNR Vignana Jyothi Institute of* *Engineering and Technology* has undergone Project based Internship from **3-May-2019 to 1-July-2019** at *Ananth Technologies Limited, Hyderabad* to fulfill the requirements for the award of degree B.Tech. (ECE). He worked on NAVIC Receiver project during this period under the supervision of **Mr. Satish Dhavan**. During his tenure with us we found his sincere and hard working. We wish him a great success in the future.

Date: (Name & Signature of Supervisor)

# ACKNOWLEDGEMENT

I express my sincere gratitude to Mr. Satish Dhawan sir for providing me an opportunity to do my project based internship at Ananth Technologies Limited, Hyderabad. The support and the guidance provided by colleagues is really valuable and would help me in my future endeavors.

This report has been prepared for the internship that has been done in the Ananth Technologies Limited, Hyderabad in order to study the acquisition and tracking techniques for IRNSS L5 signal .The aim of this internship is to be familiar to the practical aspect and uses of theoretical knowledge and clarifying the career goals, so I have successfully completed the internship and compiled this report as the summary and the conclusion that have drawn from the internship experience. I am thankful to Mr.Satish Dhawan (Deputy General Manager,Ananth Technologies LTD.) and other staff member for their co-operative support, and also presenting with an opportunity for me to have a practical experience in this organization. I sincerely thank Mr.Shiva and Ms.T. Bhavana for their valuable guidance.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl.no.** | **TOPIC** | **Pg. No.** |
| 1. | **INTRODUCTION**   1. Coarse/acquisition code 2. Precision code 3. Navigation Message | 4  5  6  6 |
| 2. | **Matlab Code Overview** | 7 |
| 3. | Simulation code1: CA.m | 11 |
| 4. | Simulation code2:CAcorr.m | 12 |
| 5. | Simulation code3:GPS\_PNcode\_gen.m | 13 |
| 6. | **Functional Codes**   1. GOLD\_code.m 2. local\_earlycode\_initial.m 3. localcode\_generate.m 4. loop\_canshu\_calculate.m 5. source\_.m 6. Signalcode.m 7. setting\_canshu.m 8. Main.m | 15  15  15  15  16  16  16  16  18 |
| 7. | **Bibliography** | 23 |

**1.INTRODUCTION**

The **Indian Regional Navigation Satellite System** (**IRNSS**), with an operational name of **NAVIC** ("sailor" or "navigator" in [Sanskrit](https://en.wikipedia.org/wiki/Sanskrit" \o "Sanskrit), [Hindi](https://en.wikipedia.org/wiki/Hindi" \o "Hindi) and many other Indian languages and also standing for **NAV**igation with **I**ndian **C**onstellation),is an autonomous regional [satellite navigation](https://en.wikipedia.org/wiki/Satellite_navigation" \o "Satellite navigation) system that provides accurate real-time positioning and timing services. It covers [India](https://en.wikipedia.org/wiki/India" \o "India) and a region extending 1,500 km (930 mi) around it, with plans for further extension. An Extended Service Area lies between the primary service area and a rectangle area enclosed by the [30th parallel south](https://en.wikipedia.org/wiki/30th_parallel_south" \o "30th parallel south) to the [50th parallel north](https://en.wikipedia.org/wiki/50th_parallel_north" \o "50th parallel north) and the [30th meridian east](https://en.wikipedia.org/wiki/30th_meridian_east" \o "30th meridian east) to the [130th meridian east](https://en.wikipedia.org/wiki/130th_meridian_east" \o "130th meridian east), 1,500–6,000 km beyond borders. The system at present consists of a constellation of seven satellites, with two additional satellites on ground as stand-by.

The constellation is in orbit as of 2018, and the system was expected to be operational from early 2018after a system check.NAVIC will provide two levels of service, the "standard positioning service", which will be open for civilian use, and a "restricted service" (an [encrypted](https://en.wikipedia.org/wiki/Encryption" \o "Encryption) one) for authorized users (including military). Due to the failures of one of the satellites and its replacement, no new date for operational status has been set.

There are plans to expand NavIC system by increasing constellation size from 7 to 11.

The system was developed partly because access to foreign government-controlled [global navigation satellite systems](https://en.wikipedia.org/wiki/Global_navigation_satellite_system" \o "Global navigation satellite system) is not guaranteed in hostile situations, as happened to the Indian military in 1999 when it was dependent on the American [Global Positioning System](https://en.wikipedia.org/wiki/Global_Positioning_System" \o "Global Positioning System) (GPS) during the [Kargil War](https://en.wikipedia.org/wiki/Kargil_War" \o "Kargil War). The Indian government approved the project in May 2014.

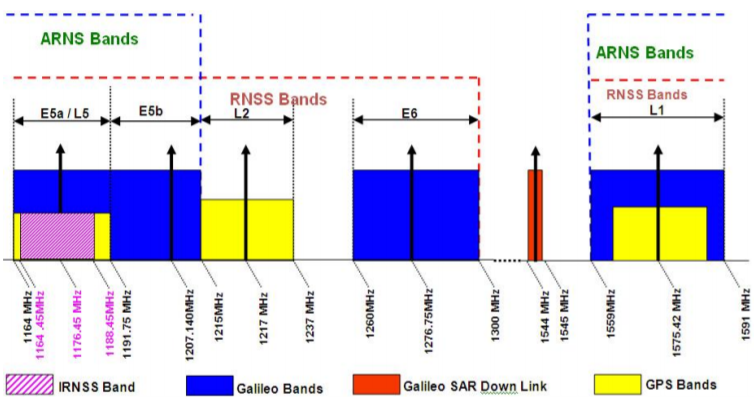
### **Signal**

NAVIC signals will consist of a Standard Positioning Service and a Precision Service. Both will be carried on L5 (1176.45 MHz) and S band (2492.028 MHz). The SPS signal will be modulated by a 1 MHz [BPSK](https://en.wikipedia.org/wiki/QPSK" \o "QPSK) signal. The Precision Service will use [BOC](https://en.wikipedia.org/wiki/Binary_offset_carrier" \o "Binary offset carrier). The navigation signals themselves would be transmitted in the [S-band](https://en.wikipedia.org/wiki/S-band" \o "S-band) frequency (2–4 GHz) and broadcast through a phased array antenna to maintain required coverage and signal strength. The satellites would weigh approximately 1,330 kg and their solar panels generate 1,400 watts.

A messaging interface is embedded in the NavIC system. This feature allows the command center to send warnings to a specific geographic area. For example, fishermen using the system can be warned about a cyclone.

**IRNSS FREQUENCY BANDS**

The IRNSS SPS service is transmitted on L5 (1164.45 – 1188.45 MHz) and S (2483.5-2500 MHz) bands. The frequency in L5 band has been selected in the allocated spectrum of Radio Navigation Satellite Services.



**Global Positioning System** (**GPS**) satellites broadcast [microwave](https://en.wikipedia.org/wiki/Microwave" \o "Microwave) signals to enable [GPS](https://en.wikipedia.org/wiki/Global_Positioning_System" \o "Global Positioning System) receivers on or near the Earth's surface to determine location and time, and to derive [velocity](https://en.wikipedia.org/wiki/Velocity" \o "Velocity). The system is operated by the [U.S. Department of Defense](https://en.wikipedia.org/wiki/U.S._Department_of_Defense" \o "U.S. Department of Defense) (DoD) for use by both the military and the general public.

[GPS signals](https://en.wikipedia.org/wiki/GPS_signals" \l "Navigation_message" \o "GPS signals) include ranging signals, used to measure the distance to the satellite, and navigation messages. The navigation messages include *[ephemeris](https://en.wikipedia.org/wiki/Ephemeris" \o "Ephemeris)* data, used to calculate the position of each satellite in orbit, and information about the time and status of the entire satellite constellation, called the *[almanac](https://en.wikipedia.org/wiki/GPS_signals" \l "Almanac)*.

There are four signals available for civilian use. In order of date of introduction, these are: [L1 C/A](https://en.wikipedia.org/wiki/GPS_signals" \l "Legacy_GPS_signals), [L2C](https://en.wikipedia.org/wiki/GPS_signals" \l "L2C), [L5](https://en.wikipedia.org/wiki/GPS_signals" \l "L5,_Safety_of_Life) and [L1C](https://en.wikipedia.org/wiki/GPS_signals" \l "L1C).[[1]](https://en.wikipedia.org/wiki/GPS_signals" \l "cite_note-1) L1 C/A is also called the *legacy signal* and is broadcast by all satellites. The other signals are called *modernized signals* and are not broadcast by all satellites. In addition, there are *restricted signals* with published frequencies and chip rates but encrypted coding intended to be used only by authorized parties. Some limited use of restricted signals can still be made by civilians without decryption; this is called *codeless* and *semi-codeless* access, and is officially supported.[[2]](https://en.wikipedia.org/wiki/GPS_signals" \l "cite_note-2)

The interface to the User Segment ([GPS receivers](https://en.wikipedia.org/wiki/GPS_receiver" \o "GPS receiver)) is described in the [Interface Control Documents (ICD)](http://www.gps.gov/technical/icwg/). The format of civilian signals is described in the [Interface Specification (IS)](https://en.wikipedia.org/wiki/GPS_signals" \l "Interface_Specification) which is a subset of the ICD.

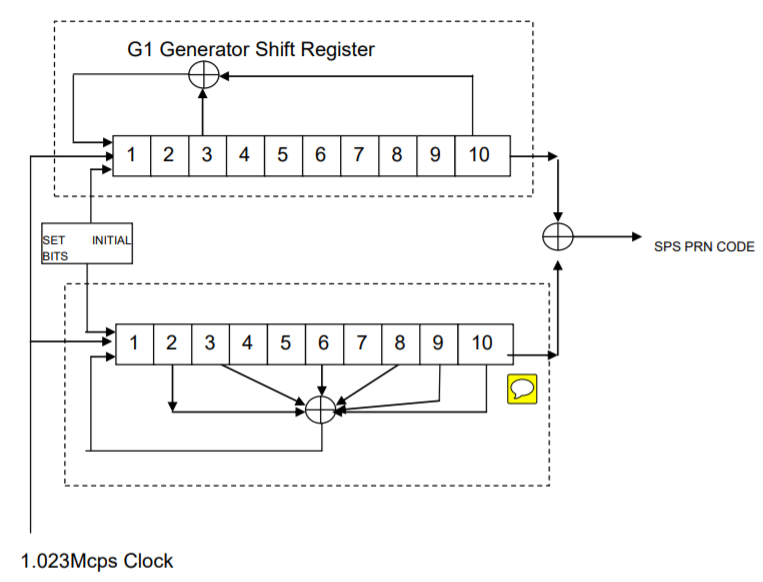
### **Coarse/acquisition code**

The C/A PRN codes are [Gold codes](https://en.wikipedia.org/wiki/Gold_code" \o "Gold code) with a period of 1023 chips transmitted at 1.023 Mchip/s, causing the code to repeat every 1 millisecond. They are [exclusive-ored](https://en.wikipedia.org/wiki/Exclusive_or" \o "Exclusive or) with a 50 bit/s [navigation message](https://en.wikipedia.org/wiki/GPS_signals" \l "Navigation_message) and the result phase modulates the carrier as [previously described](https://en.wikipedia.org/wiki/GPS_signals" \l "Common_characteristics). These codes only match up, or strongly [autocorrelate](https://en.wikipedia.org/wiki/Autocorrelation" \o "Autocorrelation) when they are almost exactly aligned. Each satellite uses a unique PRN code, which does not [correlate](https://en.wikipedia.org/wiki/Cross-correlation" \o "Cross-correlation) well with any other satellite's PRN code. In other words, the PRN codes are highly [orthogonal](https://en.wikipedia.org/wiki/Orthogonality" \l "Communications" \o "Orthogonality) to one another. The 1 ms period of the C/A code corresponds to 299.8 km of distance, and each chip corresponds to a distance of 293 m. (Receivers track these codes well within one chip of accuracy, so measurement errors are considerably smaller than 293 m.)

The C/A codes are generated by combining (using "exclusive or") 2-bit streams generated by maximal period 10 stage [linear feedback shift registers](https://en.wikipedia.org/wiki/Linear_feedback_shift_registers" \o "Linear feedback shift registers) (LFSR). Different codes are obtained by selectively delaying one of those bit streams.

The arguments of the functions therein are the number of *bits* or *chips* since their epochs, starting at 0. The epoch of the LFSRs is the point at which they are at the initial state; and for the overall C/A codes it is the start of any UTC second plus any integer number of milliseconds. The output of LFSRs at negative arguments is defined consistent with the period which is 1,023 chips (this provision is necessary because {\displaystyle B}IMG_256 may have a negative argument using the above equation).

The delay for PRN numbers 34 and 37 is the same; therefore their C/A codes are identical and are not transmitted at the same time[[4]](https://en.wikipedia.org/wiki/GPS_signals" \l "cite_note-FOOTNOTEGPS-IS-200%C2%A7&nbsp;3.2.1.3,_table_3-Ia_(p.&nbsp;4,_7)-4) (it may make one or both of those signals unusable due to mutual interference depending on the relative power levels received on each GPS receiver).



### **Precision code**

The P-code is a PRN sequence much longer than the C/A code: 6.187104 · 1012 chips (773,388 MByte). Even though the P-code chip rate (10.23 Mchips/s) is ten times that of the C/A code, it repeats only once per week, eliminating range ambiguity. It was assumed that receivers could not directly acquire such a long and fast code so they would first "bootstrap" themselves with the C/A code to acquire the spacecraft ephemerides (positions), produce an approximate time and position fix, and then acquire the P-code to refine the fix.

Whereas the C/A PRNs are unique for each satellite, each satellite transmits a different segment of a master P-code sequence approximately 2.35 · 1014 chips long (235,000,000,000,000 bits, ~26.716 terabytes). Each satellite repeatedly transmits its assigned segment of the master code, restarting every Sunday at 00:00:00 GPS time. (The GPS epoch was Sunday January 6, 1980 at 00:00:00 UTC, but GPS does not follow UTC leap seconds. So GPS time is ahead of UTC by an integral number of seconds.)

The P code is public, so to prevent unauthorized users from using or potentially interfering with it through [spoofing](https://en.wikipedia.org/wiki/Spoofing_attack" \o "Spoofing attack), the P-code is XORed with *W-code*, a cryptographically generated sequence, to produce the *Y-code*. The Y-code is what the satellites have been transmitting since the [anti-spoofing module](https://en.wikipedia.org/wiki/Selective_availability_anti-spoofing_module" \o "Selective availability anti-spoofing module) was set to the "on" state. The encrypted signal is referred to as the *P(Y)-code*.

The details of the W-code are secret, but it is known that it is applied to the P-code at approximately 500 kHz, about 20 times slower than the P-code chip rate. This has led to semi-codeless approaches for tracking the P(Y) signal without knowing the W-code.

### **Navigation message**

In addition to the PRN ranging codes, a receiver needs to know the time and position of each active satellite. GPS encodes this information into the *navigation message* and [modulates](https://en.wikipedia.org/wiki/Modulation" \o "Modulation) it onto both the C/A and P(Y) ranging codes at 50 bit/s. The navigation message format described in this section is called LNAV data (for *legacy navigation*).

The navigation message conveys information of three types:

* The GPS date and time and the satellite's status.
* The [ephemeris](https://en.wikipedia.org/wiki/Ephemeris" \o "Ephemeris): precise orbital information for the transmitting satellite.
* The almanac: status and low-resolution orbital information for every satellite.

An ephemeris is valid for only four hours; an almanac is valid for 180 days.[*[citation needed](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed" \o "Wikipedia:Citation needed)*] The receiver uses the almanac to acquire a set of satellites based on stored time and location. As each satellite is acquired, its ephemeris is decoded so the satellite can be used for navigation.

The navigation message consists of 30-second *frames* 1,500 bits long, divided into five 6-second *subframes* of ten 30-bit words each. Each subframe has the GPS time in 6-second increments. Subframe 1 contains the GPS date (week number) and satellite clock correction information, satellite status and health. Subframes 2 and 3 together contain the transmitting satellite's ephemeris data. Subframes 4 and 5 contain *page* 1 through 25 of the 25-page almanac. The almanac is 15,000 bits long and takes 12.5 minutes to transmit.

A frame begins at the start of the GPS week and every 30 seconds thereafter. Each week begins with the transmission of almanac page 1.

There are two navigation message types: LNAV-L is used by satellites with PRN numbers 1 to 32 (called *lower PRN numbers*) and LNAV-U is used by satellites with PRN numbers 33 to 63 (called *upper PRN numbers*). The 2 types use very similar formats. Subframes 1 to 3 are the same while subframes 4 and 5 are almost the same. Each message type contains almanac data for all satellites using the same navigation message type, but not the other.

Each subframe begins with a Telemetry Word (TLM) that enables the receiver to detect the beginning of a subframe and determine the receiver clock time at which the navigation subframe begins. Next is the handover word (HOW) giving the GPS time (actually the time when the first bit of the next subframe will be transmitted) and identifies the specific subframe within a complete frame. The remaining eight words of the subframe contain the actual data specific to that subframe. Each word includes 6 bits of parity generated using an algorithm based on Hamming codes, which take into account the 24 non-parity bits of that word and the last 2 bits of the previous word.

After a subframe has been read and interpreted, the time the next subframe was sent can be calculated through the use of the clock correction data and the HOW. The receiver knows the receiver clock time of when the beginning of the next subframe was received from detection of the Telemetry Word thereby enabling computation of the transit time and thus the pseudorange. The receiver is potentially capable of getting a new pseudorange measurement at the beginning of each subframe or every 6 seconds.

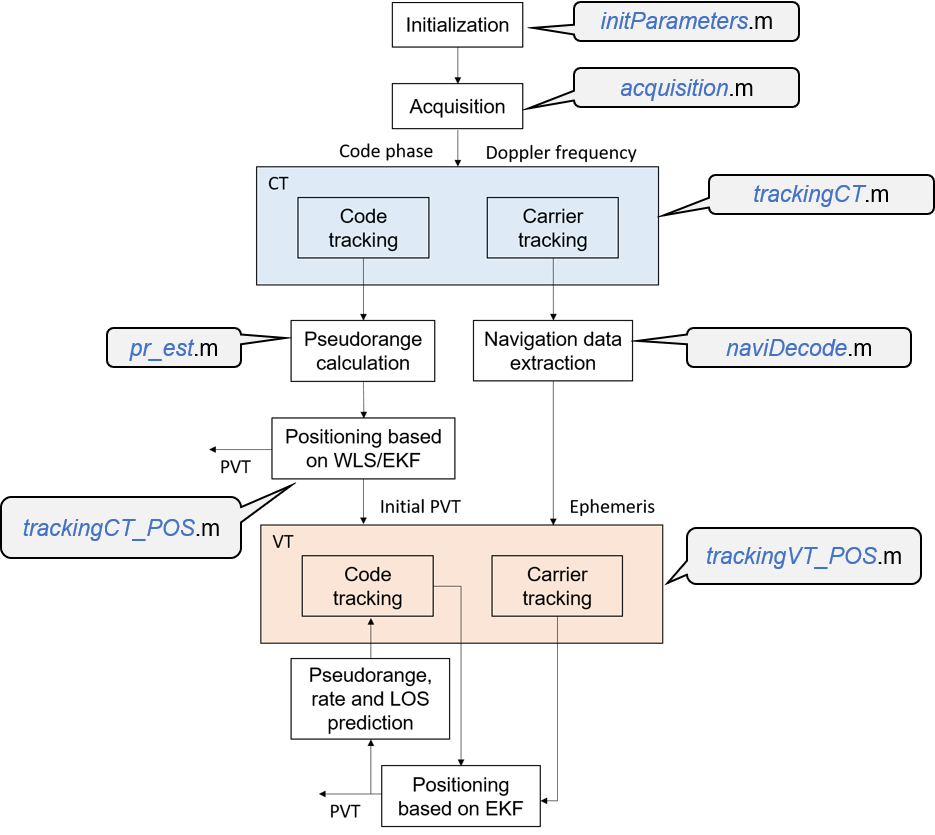
1. **Matlab Code Overview**

# Requirements

GPSSDR\_vt is currently developed and tested in MATLAB environments on Windows platforms. It can also work on a Linux operating system. It does not use any MATLAB toolboxes, but the MATLAB version is required to be greater than 7.6.

# Main Functionalities

The flowchart of GPSSDR\_vt is given in Figure 1, together with the name of the script for each functionality. Main functionalities include initialization, acquisition, conventional tracking and vector tracking, which are described in detail as follows.



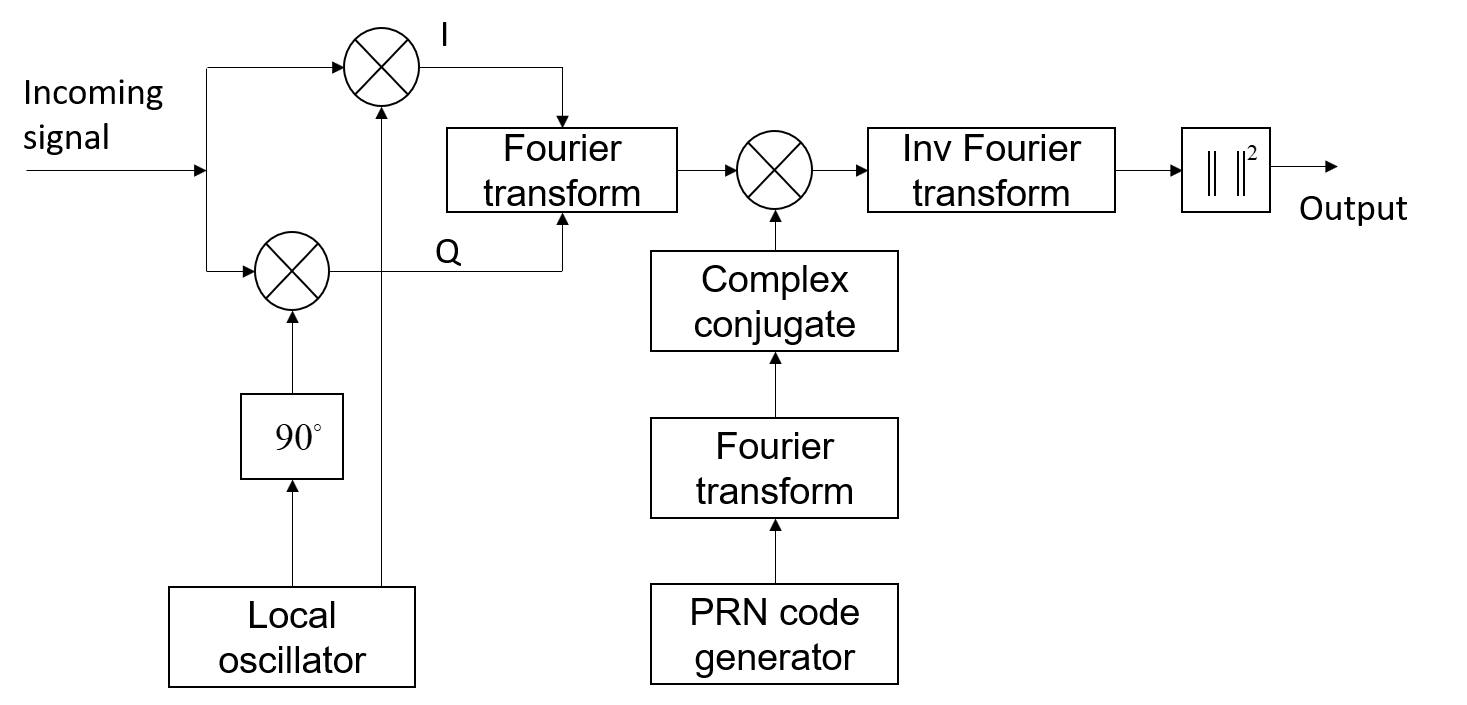
**Figure 1**. Software flowchart.

## *Initialization*

The first step to use this software is to complete configurations such as the sampling rate and intermediate frequency of the raw signal, the frequency step and band to be searched in the acquisition process, etc.

## *Acquisition*

The second module is signal acquisition, which determines code phase and Doppler frequency of visible satellites. A two-step coarse-to-fine acquisition method is used in this software. In the first step, the 4-ms data is used to detect the code phase and Doppler frequency coarsely via the parallel code phase search acquisition algorithm, as shown in Figure 2. The second step utilizes the long C/A code-stripped data to find the carrier frequency accurately via the fast Fourier transform.



**Figure 2**. Block diagram of the parallel code phase search algorithm (Van Nee and Coenen, 1991).

## *Conventional Tracking*

After obtaining the code phase and Doppler frequency, these two parameters should be refined in the tracking stage so that satellite ephemeris data can be decoded. Measurements of pseudorange and pseudorange rate can also be obtained during tracking. A second-order DLL and PLL is used in this software, as shown in Figure 3.

In conventional tracking loops, each acquired satellite is allocated to an individual tracking channel. Each channel has two closed loops, one for code and one for carrier. All tracking channels are independent of each other, i.e., no interaction between channels, and no information exchange between signal tracking and navigation processors. The pseudorange range and rate measurements are fed forward to the positioning module, e.g., EKF, to compute the navigation solution.



## *Vector Tracking*

To start vector tracking, initialization parameters, such as ephemeris data, initial receiver PVT, etc., should be provided. The pseudorange error and pseudo-range rate error extracted from the code and carrier tracking loops are used as the measurements of the EKF. The estimated receiver PVT is then used to predict the pseudorange, rate and the LOS vectors at the next epoch, closing the loop finally. The block diagram of vector tracking is shown in Figure 4.

Each acquired satellite in the incoming intermediate frequency signal is allocated to one tracking channel. In each channel, IF signals are first multiplied with the locally generated carrier replica in both in-phase and quadrature arms. Correlation is then performed between the code replicas and the received ones. In this software, three code replicas spacing of 0.5 chips are generated. Afterwards, correlation results are integrated and dumped. The output of these integrations is used as the input to the carrier/code loop discriminator to find the phase error of the local carrier and code replicas. In each carrier loop, the carrier discriminator output is filtered and fed back to the carrier numerical controlled oscillator (NCO), so as to modify the frequency of local carrier replica. For the code tracking loop, code discriminator outputs of all channels are forwarded to the navigation processor. In this software, an EKF is used. The output of the carrier loop filter, i.e., Doppler shift frequency information, is also fed into the EKF. Note that in practice the EKF update time is not necessary to be the same as the coherent integration time (typically 1 ms for GPS L1 signal). A pre-filter can be used to average the code discriminator outputs over multiple integration time, e.g., 20 ms.



Simulation code1: CA.m

function [CA\_array] = CA(Satellite\_number)

%% C/A sequence generation

%Input= CA([Satellite PRN ID])

%Output= CA\_array=array(1,1023)

%phase assignments same as NAVSTAR.

phase\_assignment=[2,6;3,7;4,8;5,9;1,9;2,10;1,8;2,9;3,10;

2,3;3,4;5,6;6,7;7,10];

n=phase\_assignment(Satellite\_number,1);

m=phase\_assignment(Satellite\_number,2);

%Creation of two Arrays g1 and g2. g1 and g2 act as shift registers.

g1=ones(1,10);

g2=g1;i=1;time=0;

%g2 initial values according to IRNSS SPS

initial\_values=[1,1,1,0,1,0,0,1,1,1;0,0,0,0,1,0,0,1,1,0;1,0,0,0,1,1,0,1,0,0;

0,1,0,1,1,1,0,0,1,0;1,1,1,0,1,1,0,0,0,0;0,0,0,1,1,0,1,0,1,1;

0,0,0,0,0,1,0,1,0,0;0,1,0,0,1,1,0,0,0,0;0,0,1,0,0,1,1,0,0,0;

1,1,0,1,1,0,0,1,0,0;0,0,0,1,0,0,1,1,0,0;1,1,0,1,1,1,1,1,0,0;

1,0,1,1,0,1,0,0,1,0;0,1,1,1,1,0,1,0,1,0];

g2(1,:)=initial\_values(Satellite\_number,:);

CA\_array=zeros(1,1023);CA\_arrayx=zeros(1,1023);

while i<1024

time=time+(1/(1.023\*(10^3)));

%PRN sequence requires Shifting and MLS output

% Register g1

output1=g1(1,10);

g1(1,2:10)=g1(1,1:9);

g1(1,1)=mod((g1(1,3)+g1(1,10)),2);

%Register g2

output2=mod((g2(1,n)+g2(1,m)),2);

g2(1,2:10)=g2(1,1:9);

g2(1,1)=mod((g2(1,2)+g2(1,3)+g2(1,6)+g2(1,8)+g2(1,9)+g1(1,10)),2);

%CA code output

output=mod((output1+output2),2);

CA\_array(1,i)=output;

CA\_arrayx(1,i)=time;

i=i+1;

end;

for i=1:1023

if CA\_array(1,i)==0;

CA\_array(1,i)=-1;

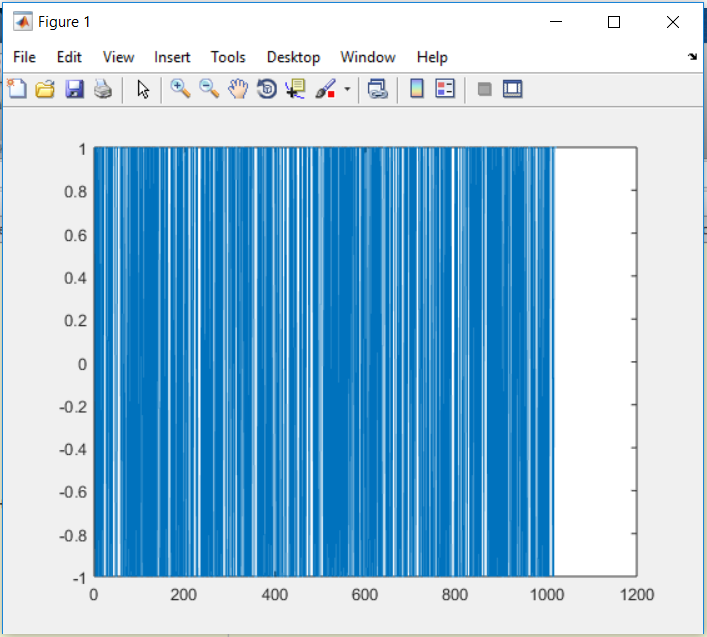
end;

end;

end

INPUT: plot( CA(2))

Output:



Simulation code2:CAcorr.m

function [correlation\_array,peak]= CAcorr(Sat1,Sat2);

%%AIM: To demonstrate the correlation properties of PRN codes.

%INPUTS: ID Numbers of PRN codes to be compared.

%OUTPUT: Correlation plot,max\_value(correlation)=peak.

%cross-correlation

correlation\_array=xcorr(CA(Sat1),CA(Sat2));

peak=max(correlation\_array);

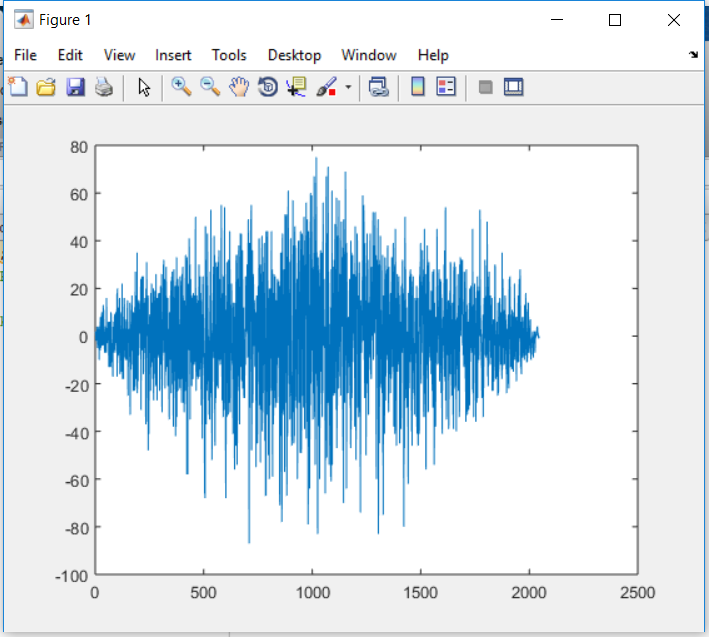
plot(correlation\_array);

assignin('base','peak',peak);

end

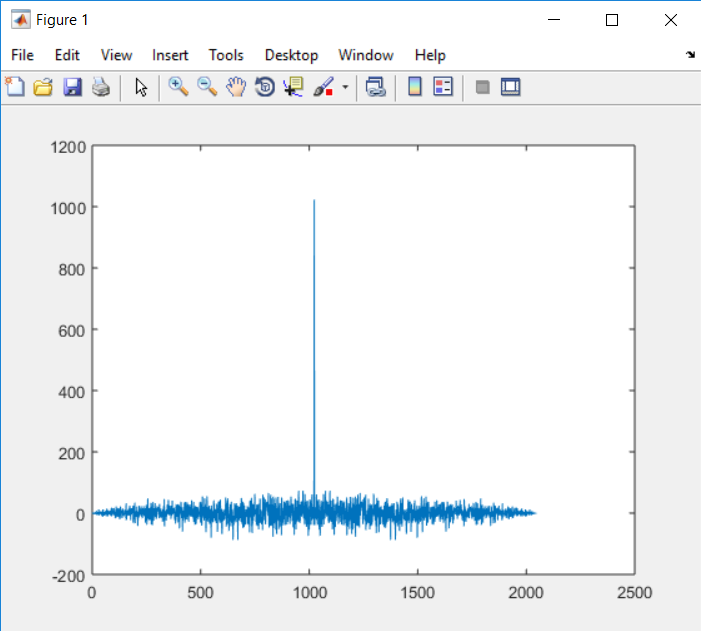
Input1:CAcorr(2,3)

Output:



Input2:CAcorr(11,11)

Output:



Simulation code3:GPS\_PNcode\_gen.m

%% C/A sequence generation

close all;clear all;clc;

shift\_G1\_reg=ones(1,10);

shift\_G2\_reg=ones(1,10);

G1=zeros(1,1023);

G2=zeros(1,1023);

L1\_CA\_code=zeros(1,1023);

for k1=1:1:1023

G1(k1)=shift\_G1\_reg(10);

G2(k1)=mod(shift\_G2\_reg(2)+shift\_G2\_reg(6),2);

L1\_CA\_code(k1)=mod(G1(k1)+G2(k1),2);

feedback\_G1=mod(shift\_G1\_reg(3)+shift\_G1\_reg(10),2);

feedback\_G2=mod(shift\_G2\_reg(2)+shift\_G2\_reg(3)+shift\_G2\_reg(6)+shift\_G2\_reg(8)+shift\_G2\_reg(9)+shift\_G2\_reg(10),2);

for k2=10:-1:2

shift\_G1\_reg(k2)=shift\_G1\_reg(k2-1);

shift\_G2\_reg(k2)=shift\_G2\_reg(k2-1);

end;

shift\_G1\_reg(1)=feedback\_G1;

shift\_G2\_reg(1)=feedback\_G2;

end;

L1\_CA\_code=(L1\_CA\_code-0.5)\*2;

k1=1:1:1023;

%L1\_CA\_code is an (1x1023) array with +1 and -1 Values.

figure,subplot(1,1,1),plot(k1,L1\_CA\_code),title('C/A 1023chip sequence');

%%GPS\_sig\_gen.m

L1\_CA\_code\_half=zeros(1,length(L1\_CA\_code)\*2);

for kk1=1:length(L1\_CA\_code)

L1\_CA\_code\_half((2\*kk1-1):(2\*kk1))=L1\_CA\_code(kk1)\*ones(1,2);

end;

for kk2 =1:length(L1\_CA\_code)

L1\_CA\_code\_samp((4\*kk2-3):(4\*kk2))=L1\_CA\_code(kk2)\*ones(1,4);

end;

Fs=4.092e6;

Fc=1.023e6;

Fd=2.5e3;

Cr=1.023e6;

Sn=Fs/Cr;

C\_NO=70;

SNR=C\_NO-10\*log10(Fs);

t=1/Fs:1/Fs:(1023\*4)/Fs;

cos\_data=cos(2\*pi\*Fc\*t);

sig\_pure=L1\_CA\_code\_samp.\*cos\_data;

signal=awgn(sig\_pure,SNR);

k=1/Fs:1/Fs:(1023\*4)/Fs;

figure,subplot(3,1,1),plot(k,L1\_CA\_code\_samp),title('L5CA code samp');

subplot(3,1,2),plot(k,cos\_data),title('cos Carrier');

subplot(3,1,3),plot(k,signal),title('Modulated BPSK signal');

figure,plot(k,abs(fftshift(fft(signal)))),title('Signal FFT Search');

%%GPS\_PNcode\_acquisition.m

fdop=linspace(0,5e3,100);

L1\_CA\_code\_local=[L1\_CA\_code\_samp(4042:4092),L1\_CA\_code\_samp(1:4041)]

peak\_out=zeros(100,100);

for dd1=1:100;

t=1/Fs:1/Fs:4092/Fs;

I\_cos=cos(2\*pi\*(Fc+fdop(dd1))\*t);

cos\_data=cos(2\*pi\*(Fc+fdop(dd1))\*t);

deal\_fd\_data=signal.\*I\_cos;

for dd2=1:100

L1\_CA\_code\_local\_kk=[L1\_CA\_code\_local(dd2:4092),L1\_CA\_code\_local(1:(dd2-1))];

deal\_xor\_data=L1\_CA\_code\_local\_kk.\*deal\_fd\_data;

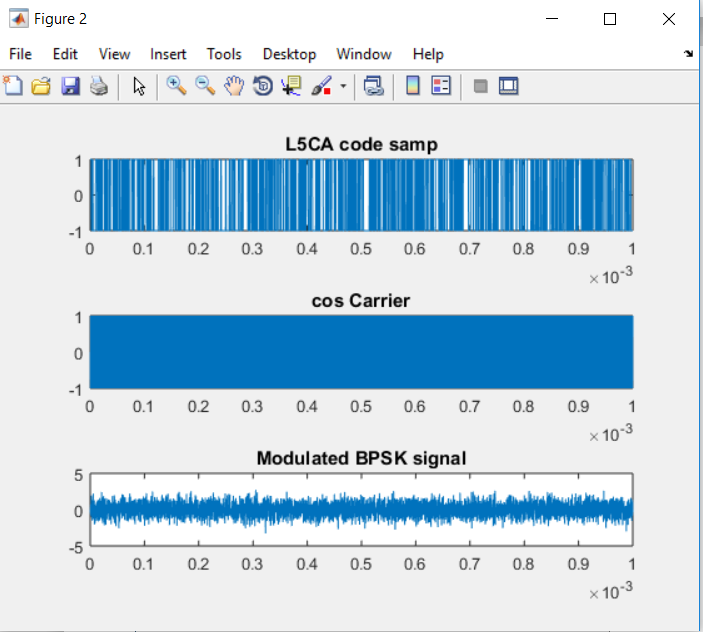
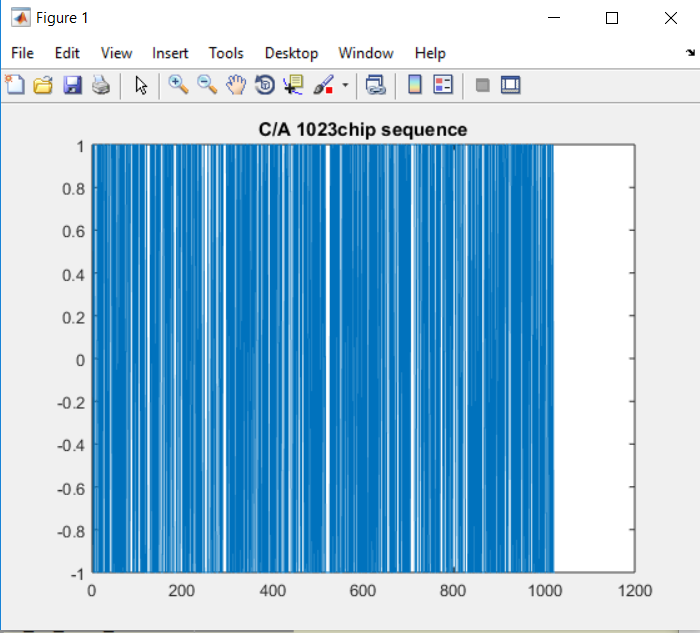
peak\_out(dd1,dd2)=sum(deal\_xor\_data);

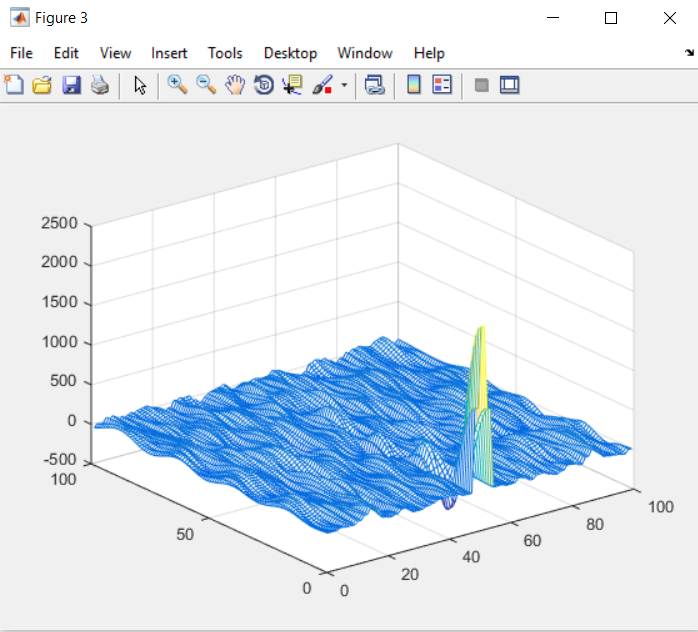
end;

end;

mesh(peak\_out);

Simulation Outputs:





**Functional Codes**

1. GOLD\_code.m

function CB1 = GOLD\_code(Satellite\_PRN\_ID)

G1=ones(1,10);

G2=G1;

%G2 initial values according to IRNSS SPS

initial\_values=[1,1,1,0,1,0,0,1,1,1;0,0,0,0,1,0,0,1,1,0;1,0,0,0,1,1,0,1,0,0;

0,1,0,1,1,1,0,0,1,0;1,1,1,0,1,1,0,0,0,0;0,0,0,1,1,0,1,0,1,1;

0,0,0,0,0,1,0,1,0,0;0,1,0,0,1,1,0,0,0,0;0,0,1,0,0,1,1,0,0,0;

1,1,0,1,1,0,0,1,0,0;0,0,0,1,0,0,1,1,0,0;1,1,0,1,1,1,1,1,0,0;

1,0,1,1,0,1,0,0,1,0;0,1,1,1,1,0,1,0,1,0];

phase\_assignment=[2,6;3,7;4,8;5,9;1,9;2,10;1,8;2,9;3,10;

2,3;3,4;5,6;6,7;7,10];

G2(1,:)=initial\_values(Satellite\_PRN\_ID,:);

CA\_array=zeros(1,1023);

n=phase\_assignment(Satellite\_PRN\_ID,1);

m=phase\_assignment(Satellite\_PRN\_ID,2);

for i=1:1023

%PRN sequence requires Shifting and MLS output

% Register g1

output1=G1(1,10);

G1(1,2:10)=G1(1,1:9);

G1(1,1)=mod((G1(1,3)+G1(1,10)),2);

%Register g2

output2=mod((G2(1,n)+G2(1,m)),2);

G2(1,2:10)=G2(1,1:9);

G2(1,1)=mod((G2(1,2)+G2(1,3)+G2(1,6)+G2(1,8)+G2(1,9)+G1(1,10)),2);

%CA code output

output=mod((output1+output2),2);

CA\_array(1,i)=output;

end

c\_out=CA\_array;

CB1=(c\_out-0.5)\*2;

1. local\_earlycode\_initial.m

function local\_earlycode = local\_earlycode\_initial(settings,code\_table)

global early\_code\_nco;

code\_word = settings.code\_word;

early\_code\_temp=[];

early\_code\_nco = settings.e\_code\_original\_phase;

Ncoh = settings.Ncoh;

for n=1:Ncoh

early\_code\_nco = early\_code\_nco+ code\_word ;

early\_code\_nco = mod(early\_code\_nco,2^32\*1023);

index=1+fix(early\_code\_nco/2^32);

c=code\_table(index);

early\_code\_temp=[early\_code\_temp,c];

end

local\_earlycode=early\_code\_temp;

1. localcode\_generate.m

function [local\_early\_code,local\_prompt\_code,local\_late\_code,local\_phase]=localcode\_generate(local\_early\_code\_last,code\_nco\_sum,code\_table,settings)

global early\_code\_nco;

Ncoh = settings.Ncoh;

code\_temp = [];

for n=1:Ncoh

early\_code\_nco = early\_code\_nco + code\_nco\_sum;

early\_code\_nco = mod(early\_code\_nco,2^32\*1023);

index = 1 + fix(early\_code\_nco/2^32);

c = code\_table(index);

code\_temp = [code\_temp,c];

if 1 == n

local\_phase = early\_code\_nco/2^32\*360;

end

end

local\_early\_code = code\_temp;

local\_prompt\_code = [local\_early\_code\_last(Ncoh-2:Ncoh),local\_early\_code(1:Ncoh-3)];

local\_late\_code = [local\_early\_code\_last(Ncoh-5:Ncoh),local\_early\_code(1:Ncoh-6)];

1. loop\_canshu\_calculate.m

function loop\_canshu = loop\_canshu\_calculate (settings)

WnF = 1.89 \* settings.FLL\_bandwidth;

WnP = 1.27 \* settings.PLL\_bandwidth;

WnD = 1.89 \* settings.DDLL\_bandwidth;

Tcoh = settings.Tcoh;

K = settings.K;

carrier\_k=0.25;

loop\_canshu.cofeone\_FLL = (sqrt(2)\*WnF\*Tcoh+WnF^2\*Tcoh^2)/carrier\_k;

loop\_canshu.cofetwo\_FLL = -(sqrt(2)\*WnF\*Tcoh)/carrier\_k;

loop\_canshu.cofeone\_PLL = (2\*WnP+2\*WnP^2\*Tcoh+WnP^3\*Tcoh^2)/carrier\_k;

loop\_canshu.cofetwo\_PLL = -(4\*WnP+2\*WnP^2\*Tcoh)/carrier\_k;

loop\_canshu.cofethree\_PLL = 2\*WnP/carrier\_k;

loop\_canshu.cofeone\_DDLL = (sqrt(2)\*WnD+WnD^2\*Tcoh)/K;

loop\_canshu.cofetwo\_DDLL = -sqrt(2)\*WnD/K;

1. source\_.m

%function source

%input : settings

%output : signal\_original

%function : generate the signal with noise;

function signal\_original = source\_ (settings)

noise = settings.noise\_std\*rand(1,settings.Ncoh);

signal\_amplitude = sqrt(10^(settings.snr/10)\*(settings.noise\_std^2\*2));

signal\_original = signal\_amplitude\*cos(2\*pi\*(settings.middle\_freq + settings.dup\_freq )\*settings.dot\_length\*settings.sample\_t) + noise;

1. Signalcode.m

function [signal\_modulate\_code,signal\_phase] = signalcode(settings,code\_table)

global modulate\_code\_nco;

Ncoh = settings.Ncoh;

signal\_modulate\_code=[];

for i=1:Ncoh

modulate\_code\_nco = modulate\_code\_nco + settings.code\_word + settings.fd\_code;

modulate\_code\_nco = mod(modulate\_code\_nco,2^32\*1023);

index = 1+fix(modulate\_code\_nco/2^32);

c = code\_table(index);

signal\_modulate\_code = [signal\_modulate\_code,c];

if 1 == i

signal\_phase = modulate\_code\_nco/2^32\*360;

end

end

1. setting\_canshu.m

%function initinal

%input: none

%output: setting

%function: set the runing condition

%including middle\_freq,dup\_freq,sample\_freq,code\_freq

%the bandwidth of PLL , FLL, DDLL

%cofe\_from\_FLL\_2\_DDLL;

%the Nco\_Length;

function settings = setting\_canshu()

settings.middle\_freq = 1.17645e6

settings.dup\_freq = 80;

settings.sample\_freq = 10e+6;

settings.code\_freq = 1.023e+6;

settings.code\_length = 1023;

settings.snr = -20;

settings.FLL\_flag = 1;

settings.PLL\_flag = 0;

settings.FLL\_bandwidth = 4.2;%FLL

settings.PLL\_bandwidth = 10;%PLL

settings.DDLL\_bandwidth = 2;

settings.cofe\_FLL\_auxi\_DDLL = 1/763;

settings.nco\_Length = 32;

settings.noise\_std = 1;

% setting.length = (1:10000);

% setting.length\_no = 10000;

settings.sample\_t = 1/settings.sample\_freq;

settings.K = 1;

settings.transfer\_coef = (2^settings.nco\_Length)/settings.sample\_freq;

settings.middle\_freq\_nco = settings.middle\_freq\*settings.transfer\_coef;

settings.Ncoh = (settings.sample\_freq / settings.code\_freq)\*settings.code\_length;

settings.Tcoh = settings.Ncoh \*settings.sample\_t;

settings.dot\_length = [1:settings.Ncoh];

settings.code\_word = settings.code\_freq \* settings.transfer\_coef;

settings.fd\_code = settings.dup\_freq\*(1/763)\*settings.transfer\_coef;

%setting.fd\_code = setting.dup\_freq\*settings.cofe\_FLL\_auxi\_DDLL\*setting.transfer\_coef;

settings.e\_code\_original\_phase = (2^settings.nco\_Length)/2;

settings.modulate\_code\_bias\_phsae = (2^settings.nco\_Length)/8;

settings.signal\_phase = 0;

settings.local\_phase = 0;

1. Main.m

%% Main Function

% For MATLAB computaional practicality, the carrier frequency of IRNSS has

% been reduced by a factor of 100 from 1176.45MHz to 1.17645MHz

clc;

clear all;

close all;

format long g;

Satellite\_PRN\_ID=input('Enter the PRN ID of the Satellite to be considered.');

settings = setting\_canshu();

loop\_para = loop\_canshu\_calculate(settings);

code\_table=GOLD\_code(Satellite\_PRN\_ID);

fll\_nco\_adder = 0;

carrier\_nco\_sum = 0;

pll\_nco\_adder = 0;

loop\_count = 0;

code\_nco\_sum = 0;

code\_nco\_adder = 0;

n\_IQ = 2;

n = 3;

output\_fll(2:3) = 0;

output\_filter\_fll(1:3) = 0;

output\_filter\_pll(1:3) = 0;

output\_pll(2:3) = 0;

output\_filter\_ddll(1:3) = 0;

pll\_after\_filter = 0;

Tcoh = settings.Tcoh;

global modulate\_code\_nco;

modulate\_code\_nco = settings.modulate\_code\_bias\_phsae;

global early\_code\_nco;

% early\_code\_nco = setting.e\_code\_original\_phase;

local\_early\_code\_last = local\_earlycode\_initial(settings,code\_table);

for loop\_num = 1 : 1500

signal\_original = source\_(settings);

settings.dot\_length = settings.dot\_length + 10000;

flag(loop\_num) = settings.PLL\_flag;

fd\_plot(loop\_num) = settings.dup\_freq;

[signal\_modulate\_code,settings.signal\_phase] = signalcode(settings,code\_table);

receive\_signal = signal\_modulate\_code.\* signal\_original;

% receive\_signal = original\_signal;

for demond\_num = 1:settings.Ncoh

local\_cos(demond\_num) = cos(2\*pi\*carrier\_nco\_sum/2^settings.nco\_Length);

local\_sin(demond\_num) = -sin(2\*pi\*carrier\_nco\_sum/2^settings.nco\_Length);

carrier\_nco\_sum = carrier\_nco\_sum + settings.middle\_freq\_nco + fll\_nco\_adder + pll\_nco\_adder ;

% carrier\_nco\_sum = mod(carrier\_nco\_sum,2^setting.nco\_Length);

end,

code\_nco\_sum = code\_nco\_adder + settings.code\_word + fll\_nco\_adder\*settings.cofe\_FLL\_auxi\_DDLL;

%code\_nco\_sum = code\_nco\_adder + settings.code\_word + fll\_nco\_adder\*(1/763);

[local\_early\_code,local\_prompt\_code,local\_late\_code,settings.local\_phase]=localcode\_generate(local\_early\_code\_last,code\_nco\_sum,code\_table,settings);

local\_early\_code\_last = local\_early\_code;

I\_demon\_carrier = local\_cos.\*receive\_signal;

Q\_demon\_carrier = local\_sin.\*receive\_signal;

% save\_I\_demon\_carrier = [save\_I\_demon\_carrier I\_demon\_carrier];

% save\_Q\_demon\_carrier = [save\_Q\_demon\_carrier Q\_demon\_carrier];

I\_E\_final = sum(I\_demon\_carrier.\*local\_early\_code);

Q\_E\_final = sum(Q\_demon\_carrier.\*local\_early\_code);

I\_P\_final(n\_IQ) = sum(I\_demon\_carrier.\*local\_prompt\_code);

Q\_P\_final(n\_IQ) = sum(Q\_demon\_carrier.\*local\_prompt\_code);

I\_L\_final = sum(I\_demon\_carrier.\*local\_late\_code);

Q\_L\_final = sum(Q\_demon\_carrier.\*local\_late\_code);

% I\_P\_final(n\_IQ) = sum(I\_demon\_carrier);

% Q\_P\_final(n\_IQ) = sum(Q\_demon\_carrier);

if 1 == loop\_num

I\_P\_final(n\_IQ - 1) = I\_P\_final(n\_IQ);

Q\_P\_final(n\_IQ - 1) = Q\_P\_final(n\_IQ);

else

% %

dot\_fll = I\_P\_final(n\_IQ - 1) \* I\_P\_final(n\_IQ) + Q\_P\_final(n\_IQ - 1) \* Q\_P\_final(n\_IQ);

cross\_fll = I\_P\_final(n\_IQ - 1) \* Q\_P\_final(n\_IQ) - I\_P\_final(n\_IQ) \* Q\_P\_final(n\_IQ - 1);

output\_fll(n) = atan2(cross\_fll,dot\_fll)/(Tcoh\*2\*pi);

result\_discriminator\_Fll(loop\_num) = output\_fll(n);

output\_filter\_fll(n) = (loop\_para.cofeone\_FLL \* output\_fll(n)) + (loop\_para.cofetwo\_FLL \* output\_fll(n - 1)) + (2 \* output\_filter\_fll(n - 1)) - output\_filter\_fll(n - 2);

fll\_after\_filter(loop\_num) = output\_filter\_fll(n);

fll\_nco\_adder = output\_filter\_fll(n) \* settings.transfer\_coef ;

output\_fll(n - 1)=output\_fll(n);

output\_filter\_fll(n - 2)=output\_filter\_fll(n - 1);

output\_filter\_fll(n - 1)=output\_filter\_fll(n);

if settings.PLL\_flag == 1

output\_pll(n) = atan2(Q\_P\_final(n\_IQ),I\_P\_final(n\_IQ));

output\_filter\_pll(n) = loop\_para.cofeone\_PLL\*output\_pll(n) + loop\_para.cofetwo\_PLL\*output\_pll(n-1)+loop\_para.cofethree\_PLL\*output\_pll(n-2)+2\*output\_filter\_pll(n-1)-output\_filter\_pll(n-2);

result\_discriminator\_Pll(loop\_num) = output\_pll(n);

pll\_after\_filter(loop\_num) = output\_filter\_pll(n);

pll\_nco\_adder = (output\_filter\_pll(n)/(2\*pi)) \* settings.transfer\_coef;

% output\_pll(1:2) = output\_pll(2:3);

% output\_filter\_pll(1:2) = output\_filter\_pll(2:3);

output\_pll(n-2) = output\_pll(n-1);

output\_pll(n-1) = output\_pll(n);

output\_filter\_pll(n-2) = output\_filter\_pll(n-1);

output\_filter\_pll(n-1) = output\_filter\_pll(n);

end

I\_P\_final(n\_IQ - 1) = I\_P\_final(n\_IQ);

Q\_P\_final(n\_IQ - 1) = Q\_P\_final(n\_IQ);

if 0 == settings.PLL\_flag && abs(output\_fll(n))<10

loop\_count = loop\_count + 1;

if loop\_count>200

settings.PLL\_flag = 1;

end

elseif 1 == settings.PLL\_flag && abs(output\_fll(n))>30

loop\_count = loop\_count-1;

if 0 == loop\_count

settings.PLL\_flag = 0;

end

end

end,

output\_ddll(n) = ((I\_E\_final - I\_L\_final)\*I\_P\_final(n\_IQ) + (Q\_E\_final - Q\_L\_final)\*Q\_P\_final(n\_IQ) )/((I\_P\_final(n\_IQ)^2 + Q\_P\_final(n\_IQ)^2)\*2); % DDLL\_discri\_1

result\_ddll(loop\_num) = output\_ddll(n);

output\_filter\_ddll(n) = output\_filter\_ddll(n -1) + (loop\_para.cofeone\_DDLL\*output\_ddll(n)) + loop\_para.cofetwo\_DDLL\*output\_ddll(n - 1);

result\_DDLL\_filter(loop\_num) = output\_filter\_ddll(n);

code\_nco\_adder = output\_filter\_ddll(n) \* settings.transfer\_coef ;

% Code\_NCO=0;

% C(loop\_num)=code\_nco\_adder;

output\_ddll(n - 1)=output\_ddll(n);

output\_filter\_ddll(n - 1) = output\_filter\_ddll(n);

code\_phase\_discrim(loop\_num) = settings.signal\_phase - settings.local\_phase ;

end

%%Ploting Results

figure ;

subplot(2,1,1);

plot(flag);

title('FLL+PLL');

legend('PLL Output');

xlabel('Time(micro-second)')

subplot(2,1,2);

plot(fll\_after\_filter + (pll\_after\_filter/(2\*pi)),'b');

hold on;

plot(fd\_plot,'r');

legend('FLL after Fliter','PLL after Filter/(2\*pi)');

xlabel('Time(micro-second)')

figure ;

subplot(2,1,1);

plot(result\_ddll);

title('FLL+PLL');

legend('DLL output');

xlabel('Time(micro-second)')

subplot(2,1,2);

plot(result\_DDLL\_filter,'b');

hold on;

plot(fd\_plot,'r');%

legend('Result DLL using FILTER','');

legend('DLL using FILTER ');

xlabel('Time(micro-second)')

figure ;

subplot(2,1,1);

plot(result\_discriminator\_Fll);

title('FLL');

legend('PLL');

xlabel('Time(micro-second)')

subplot(2,1,2);

plot(fll\_after\_filter,'b');

hold on;

plot(fd\_plot,'r');

legend('FLL after FILTER');

xlabel('Time(micro-second)')

figure ;

subplot(2,1,1);

plot(result\_discriminator\_Pll);

title('PLL');

legend('PLL output');

xlabel('Time(micro-second)?')

subplot(2,1,2);

plot((pll\_after\_filter/(2\*pi)),'b');

hold on;

plot(fd\_plot,'r');

legend('PLL output using FILTER');

xlabel('Time(micro-second)')

figure ;

subplot(2,2,1);

plot(result\_discriminator\_Fll);

ylabel('FLL Output');

xlabel('Time(micro-second)')

subplot(2,2,2);

plot(fll\_after\_filter + (pll\_after\_filter/(2\*pi)));

ylabel('FLL FILTER+(PLL FILTER)/(2\*pi)');

xlabel('Time(micro-second)')

subplot(2,2,3);

plot( flag);

ylabel('PLL Output');

xlabel('Time(micro-second)')

subplot(2,2,4);

plot( fll\_after\_filter);

ylabel(' FLL Output');

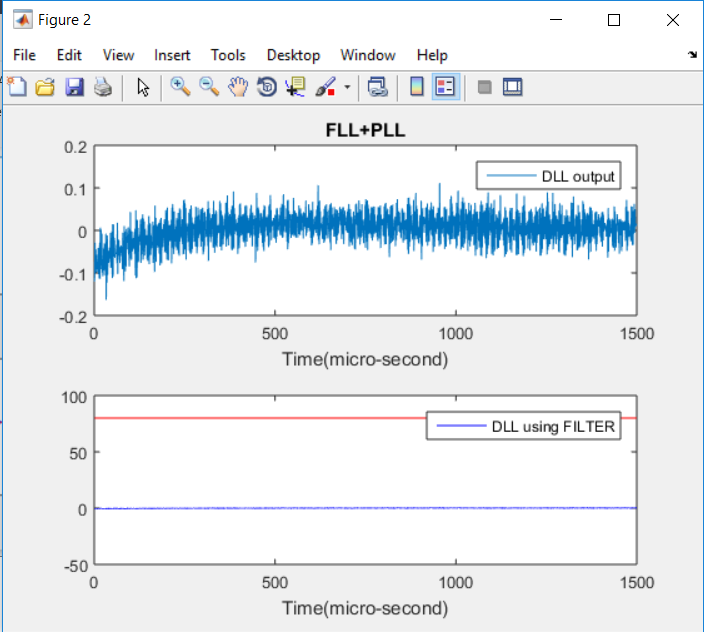
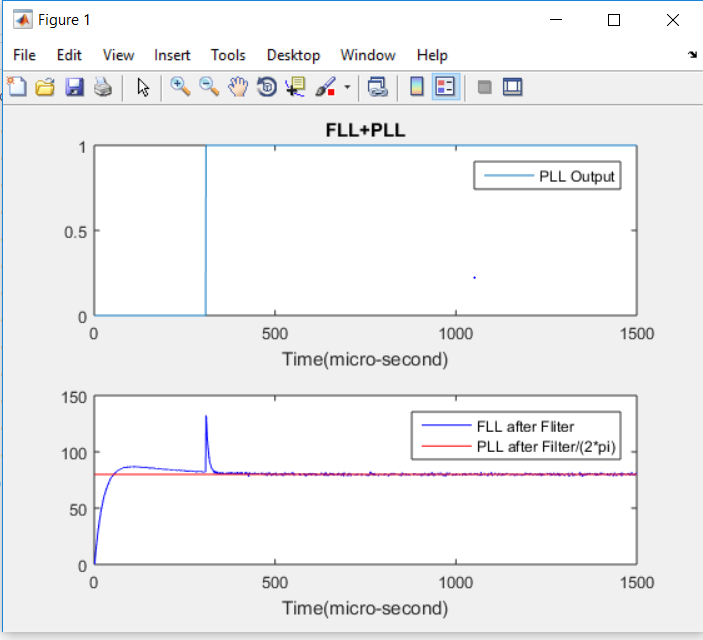
xlabel('Time(micro-second)');

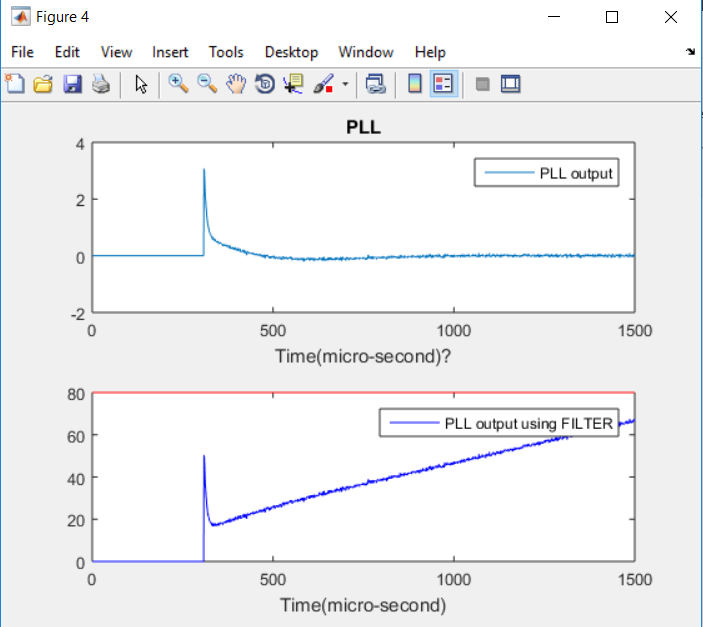
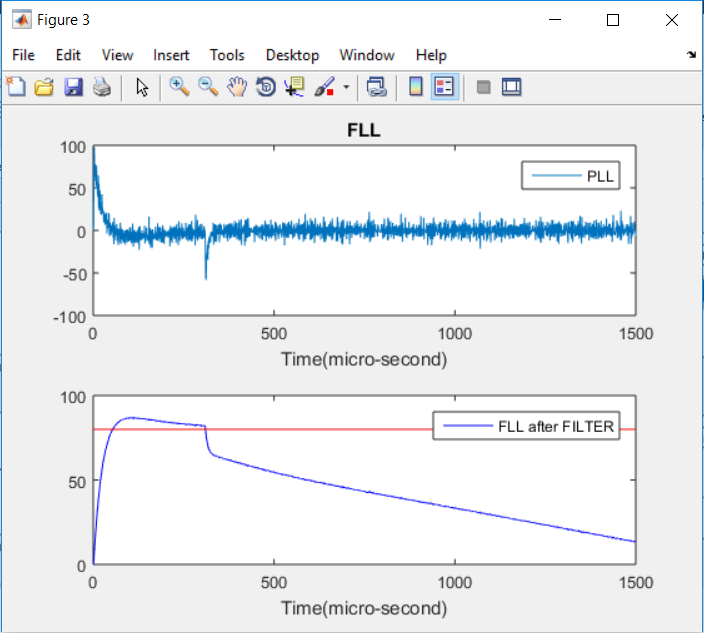
disp('Respective Outputs Have been Generated');

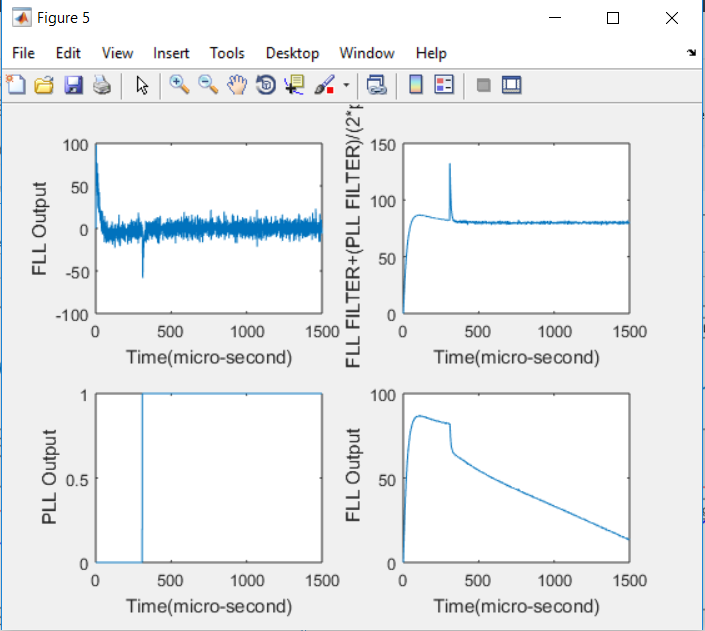
Input Prompt:Enter the PRN ID of the Satellite to be considered.

Input:5

Outputs:







**Bibliography**

**Documents:**

1. Indian Regional Navigation Satellite System, Signal in space ICS for SPS ISRO-IRNSS-ICS-SPS-1.1
2. Global Positioning System Directorate Interface specifications IS-GPS-200J
3. GPS SIGNAL ACQUISITION AND TRACKING USING SOFTWARE GPS RECEIVER Falin Wu∗ and Kefei Zhang The RMIT University Melbourne, Australia ∗falin.wu@rmit.edu.au Akio Yasuda Tokyo University of Marine Science and Technology Tokyo, Japan
4. GPS. The First Global Navigation Satellite System,Trimble Navigation Limited 935 Stewart Drive Sunnyvale, California 94085
5. Fundamentals of Global Positioning System Receivers: A Software Approach James Bao-Yen Tsui Copyright  2000 John Wiley & Sons, Inc. Print ISBN 0-471-38154-3 Electronic ISBN 0-471-20054-9
6. Fundamentals of Global Positioning System Receivers: A Software Approach James Bao-Yen Tsui Copyright  2000 John Wiley & Sons, Inc. Print ISBN 0-471-38154-3 Electronic ISBN 0-471-20054-9
7. REAL-TIME ACQUISITION AND TRACKING FOR GPS RECEIVERS Abdulqadir Alaqeeli, Janusz Starzyk, Frank van Graas School of Electrical Engineering and Computer Science Ohio University Athens, OH 4570 1
8. SPS Gold Code Generation and Implementation for IRNSS User Receiver Shachi Varku1 , Swetha A2 , Sharanya S Konandur3 , Dileep D4 , Aklpita L Kulkarni5 1,2,3,5(Department Electronics and Communication, Dr.Ambedkar Institute of Technology, Bangalore,India 4 (ISTRAC/ISRO, Bangalore, India

**WEB PAGES**

1. <http://www.codeforge.com/article/211133>
2. <http://www.codeforge.com/article/239624>
3. github.com