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]=localco
    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*o
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');
```

```
Error using input
Cannot call INPUT from EVALC.

Error in Main (line 8)
Satellite_PRN_ID=input('Enter the PRN ID of the Satellite to be
 considered.');
```

Published with MATLAB® R2015a