Appendix D

A Source Code of Simulation Program for the Proposed Noncoherent Scheme

```
/*
This is a program for simulating the noncoherent close-loop acquisition. The program has an output in
the file's name
" t_close.txt".
*/

#include <math.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

const double Tc = 0.000001; // chip rate = 1 MHz
float Y,Power,SNR,initialY,ran_phase;
int Hf;
int far *alpha;
int loop1,J,JJ,n0,sum_1,sum_2,initialJ;
int SX[9] = {1,1,1,1,1,1,1,1,1}, SR[9] = {1,1,1,1,1,1,1,1,1};
long KK, K = 511;
double
KVCC,Town,Townhat,S1_signal,S1_signal_sin,S1_signal_cos,S2_signal,S2_signal_sin,S2_signal_cos
;
double far sum_r1 ,sum_r2 = 1.,sum_i1,sum_i2 = 1.,*HH1,*HH2,*HH3,*HH4;

void get_J_Y(double T1,double T2);//change the phase difference to be in term of J and Y
void forward_shift(int sh[9]);// function for shifting the shift-regester in the forward direction
void reward_shift(int sh[9]);// function for shifting the shift-regester in the reward direction
void equal_shift(int sh1[9],int sh2[9]); // function for setting sh1 by sh2
void print_shift(int sh[9]);//function for printting the value in the shift-regester
double randn(double qq);// generate noise
void initial(void);
void VCC_loop(void);//VCC loop in the receiver
void gen_alpha(void);//generate the alpha signal
void clearH(double far *H);//fuction for clearing the value in buffer H
void set_shift(int sh1[9]);//setting the value in shift regester  = 1


void main()
{
        FILE  *looptime;
        float threshold,factor;

        int declare = 0,Kp,miss;
        long int time_1,time_acquisition,square,var,KKK,Trial,false_alarm;
        alpha = (int far *) farmalloc(511*sizeof(int));
        HH1 = (double far *) farmalloc(8000*sizeof(double));
        HH2 = (double far *) farmalloc(8000*sizeof(double));
        HH3 = (double far *) farmalloc(8000*sizeof(double));
        HH4 = (double far *) farmalloc(8000*sizeof(double));
```

```
        Power = 1;
        printf("Enter the value of integral lenght = ");
        scanf("%d",&n0);
        printf("Enter the value of SNR = ");
        scanf("%f",&SNR);
        printf("Enter the value of penalty time = ");
        scanf("%d",&Kp);
        printf("Enter the value of trial = ");
        scanf("%ld",&Trial);
        printf("Enter the factor of threshold = ");
        scanf("%f",&factor);
        printf("Enter the size of filter = ");
        scanf("%d",&Hf);
        printf("Hf = %d",Hf);
  getch();


        threshold = Tc*sqrt(2*Power)*n0/factor*Tc*sqrt(2*Power)*n0/factor;
        time_acquisition = 0;
        miss = 0;
        false_alarm = 0;
        square = 0;
        KVCC = 1/(n0*sqrt(2*Power));
        printf("program is running");
        gen_alpha();

for(KKK = 1;KKK <= Trial; KKK++)
{

        time_1 = 0;
        declare = 0;
        Townhat = 0;
        sum_r1 = 0.0;
        sum_i1 = 0.0;
        sum_r2 = 0.0;
        sum_i2 = 0.0;
        clearH(HH1);
        clearH(HH2);
        clearH(HH3);
        clearH(HH4);
        KK = 0;
        set_shift(SX);
        Town = (KKK%511 + (double)(rand()%100)/100 )*Tc;
        ran_phase =  (float)(( rand() % 1000 ) / 1000.)*8.0*atan(1.0);
        get_J_Y(Townhat,Town);
        initialJ = J;
        initialY = Y;
        printf("J = %d",J);
        initial();
        while (declare == 0)
        {
                get_J_Y(Townhat,Town);
                equal_shift(SR,SX);

                if(J <= 0)
                        {
                                for(loop1 = 0; loop1 < (-1)*J; loop1++)reward_shift(SR);
```

```
                }
        else
                {
                        for(loop1 = 0; loop1 < J; loop1++)forward_shift(SR);
                }

/******************upper brance******************/
        sum_1 = 0;
        sum_2 = 0;
        if(Y >= 0)
        {
        for (loop1 = 0; loop1 < n0 ; ++loop1)
                {  reward_shift(SR);
                        sum_2 = sum_2 + SX[8]*SR[8];
                        forward_shift(SR);
                        sum_1 = sum_1 + SX[8]*SR[8];
                        forward_shift(SX);
                        forward_shift(SR);
                }
        }
        else
        {
                        for (loop1 = 0; loop1 < n0 ; ++loop1)
                {  forward_shift(SR);
                        sum_2 = sum_2 + SX[8]*SR[8];
                        reward_shift(SR);
                        sum_1 = sum_1 + SX[8]*SR[8];
                        forward_shift(SX);
                        forward_shift(SR);
                }
        }
    for(loop1 =0; loop1 <n0; loop1++)
        {
                reward_shift(SX);
        }

    S1_signal_sin = sqrt(2*Power)*Tc*(sum_1*(1-fabs(Y)) + sum_2*fabs(Y))*sin(ran_phase) +
Tc*randn((double)(2*Power*n0/SNR));
    S1_signal_cos = sqrt(2*Power)*Tc*(sum_1*(1-fabs(Y)) + sum_2*fabs(Y))*cos(ran_phase) +
Tc*randn((double)(2*Power*n0/SNR));
    S1_signal = S1_signal_sin*S1_signal_sin + S1_signal_cos*S1_signal_cos ;


 /******************lower brance******************/

        Townhat = Townhat - 256.*Tc;
        get_J_Y(Townhat,Town);
        Townhat = Townhat + 256.*Tc;
        equal_shift(SR,SX);

        if(J <= 0)
                {
                        for(loop1 = 0; loop1 < (-1)*J; loop1++)reward_shift(SR);
                }
        else
                {
                        for(loop1 = 0; loop1 < J; loop1++)forward_shift(SR);
```

```
                    }
        sum_1 = 0;
        sum_2 = 0;
        if(Y >= 0)
                {
                for (loop1 = 0; loop1 < n0 ; ++loop1)
                        {  reward_shift(SR);
                                sum_2 = sum_2 + SX[8]*SR[8];
                                forward_shift(SR);
                                sum_1 = sum_1 + SX[8]*SR[8];
                                forward_shift(SX);
                                forward_shift(SR);
                        }
                }
                else
                {
                                for (loop1 = 0; loop1 < n0 ; ++loop1)
                        {  forward_shift(SR);
                                sum_2 = sum_2 + SX[8]*SR[8];
                                reward_shift(SR);
                                sum_1 = sum_1 + SX[8]*SR[8];
                                forward_shift(SX);
                                forward_shift(SR);
                        }
                }

for(loop1 =0; loop1 <n0; loop1++)reward_shift(SX);

S2_signal_sin = sqrt(2*Power)*Tc*(sum_1*(1-fabs(Y)) + sum_2*fabs(Y))*sin(ran_phase) + Tc*randn
((double)(2*Power*n0/SNR));
S2_signal_cos = sqrt(2*Power)*Tc*(sum_1*(1-fabs(Y)) + sum_2*fabs(Y))*cos(ran_phase) +
Tc*randn((double)(2*Power*n0/SNR));
S2_signal = S2_signal_sin*S2_signal_sin + S2_signal_cos*S2_signal_cos ;


time_1 = time_1 + n0;

/**************check threshold ************/
        get_J_Y(Townhat,Town);
        JJ = J%511;
        if ((S1_signal >S2_signal) && (S1_signal >=threshold))
                {
                if ( J%511 == 0)
                        {
                                        declare = 1;
                                        printf("\nhi 1    time = %ld",time_1);

                        }
                else
                        {
                                        time_1 = time_1 + n0*Kp;
                                        printf("\nfalse alarm1");
                                        false_alarm++;

                        }
                }
         else if((S2_signal >S1_signal) && (S2_signal >=threshold))
```

```
                {
                if ( (fabs(JJ+Y)>255)&&(fabs(JJ+Y)<256) )
                        {
                                declare = 1;
                                printf("\nha 2  time = %ld",time_1);

                        }
                else
                        {
                                time_1 = time_1 + n0*Kp;
                                printf("\nfalse alarm2   " );
                                false_alarm++;
                        }
                }
        else
                {
                if (J%511 == 0 ||(fabs(JJ+Y)>255)&&(fabs(JJ+Y)<256))
                        {
                                printf("\nmiss detection");
                                miss++;
                        };
                }
        VCC_loop();
 }//end while

square = square + ((long)time_1/1000)*((long)time_1/1000);
time_acquisition = time_acquisition + ((long)time_1);

}//end for

time_acquisition = time_acquisition/Trial;
var =  (long)square/Trial - (long)(time_acquisition/1000)*(time_acquisition/1000);
printf("\nacquisition time = %ld",time_acquisition);
printf("\nsquare =%ld",square);
printf("\nvariance = %ld",var);
printf("\nmiss detection = %d",miss);
printf("\nfalse alarm = %d",false_alarm);
printf("\nHf = %d",Hf);


/***************write to file  ************/

        if((looptime = fopen("t_nonclo.txt","w")) == NULL)
        {
        printf("Error ya");
        exit(1);
        }

  fprintf(looptime,"The result of serial search");
  fprintf(looptime,"\nSNR = %f",SNR);
  fprintf(looptime,"\nmean time = %ld\nvar = %ld\nfalse alarm =
%ld",time_acquisition,var,false_alarm);
  fprintf(looptime,"\n miss detection = %d",miss);
  fprintf(looptime,"\n Kp = %d",Kp);
  fprintf(looptime,"\nthe filter size = %d",Hf);
  fprintf(looptime,"\nfactor = %f",factor);
  fprintf(looptime,"\nthreshold value = %e",threshold);
```

```
        fclose(looptime);

}//end main


void get_J_Y(double T1,double T2)
{
double dif_phase = 0;
int sign;
                dif_phase = fabs(T1 - T2);
                if (T1 -T2 >= 0)sign = 1;
                else sign = -1;
                J = dif_phase/Tc;
                Y = (float)fmod(dif_phase,Tc)/Tc;
                if (Y >= 0.5)
                {
                 Y = 1 - Y;
                 J = J + 1;
                }
                Y = (float)Y*sign;
                J = (int)J*sign;
}



void equal_shift(int sh1[9],int sh2[9])
{
 int i;
 for(i = 0; i < 9 ; i++)
                sh1[i] = sh2[i];
}
void print_shift(int sh[9])
{
        int i;
        printf("\n");
        for(i = 0; i < 9;i++) printf("%d ",sh[i]);
}

void forward_shift(int sh[9])
{
        int doomy,i;
        doomy = (-1)*sh[8]*sh[3];
        for(i = 8;i>=1;--i)
                sh[i]=sh[i - 1];
        sh[0]=doomy;
}
void reward_shift(int sh[9])
{
        int doomy,i;
        doomy = (-1)*sh[4]*sh[0];
        for(i = 1;i<=8;i++)
                sh[i-1]=sh[i];
        sh[8]=doomy;
}
```

```
double randn(double qq)
{

        double PI,B,XIN,R,revalue;
        PI = 4.0*atan(1.0);
        B   = (double)(( rand() % 1000 ) / 1000.)*2*PI;
        XIN = (double)(( rand() % 1000 ) / 1000.);
        if(qq < 0.0)
        {
                printf("qq = %f", qq);
        //      getch();
        }
        R = sqrt(2.0*qq*log(1.0/(1.0 - XIN)));
        revalue = R*cos(B);
        return(revalue);
}

void VCC_loop(void)
{
                int sum_5,sum_6,BB,i;
                float  sum_3,sum_4;
                double zz,doomy,LLL,sum_34;
                for (i = 0; i < n0; i++)
                {

                        BB = KK%Hf;
/**********************upper branch******************/
                        get_J_Y(Townhat,Town);

                        sum_5 = ((*(alpha+((2*K + KK-1+J)%K)))-(*(alpha+((2*K +
KK+J)%K))));
                        sum_6 = ((*(alpha+((2*K + KK+J)%K)))-(*(alpha+((2*K +
1+KK+J)%K))));
                        sum_3 =  (float)(0.5+Y)*SX[8]*sum_5;
                        sum_4 =  (float)(0.5-Y)*SX[8]*sum_6;


                        zz = (double)(sum_5*sum_5*(0.5+Y) + sum_6*sum_6*(0.5-Y))/SNR;
                //      printf("\nzz = %f sum_5 = %d  sum_6 = %d  KK =
%d",zz,sum_5,sum_6,KK);
                        doomy = *(HH1+BB);
                        //printf("\n doomy = %f",doomy);
                        LLL = randn(zz);
                        *(HH1+BB)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*cos(ran_phase) +
LLL)/Hf;
                         sum_r1 = sum_r1 + (*(HH1+BB)) - doomy;



                        doomy = *(HH2+BB);
                        LLL =  randn(zz)/sqrt(2);
                        *(HH2+BB)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*sin(ran_phase) +
LLL)/Hf;
                        sum_i1 = sum_i1 + (*(HH2+BB)) - doomy;

/****************lower branch*********************/
                        Townhat = Townhat + (127.75)*Tc;
```

```
                        get_J_Y(Townhat,Town);
                        Townhat = Townhat - (127.75)*Tc;
                        sum_5 = ((*(alpha+((2*K + KK-1+J)%K)))-(*(alpha+((2*K +
KK+J)%K))));
                        sum_6 = ((*(alpha+((2*K + KK+J)%K)))-(*(alpha+((2*K +
1+KK+J)%K))));
                        sum_3 =  (0.5+Y)*SX[8]*sum_5;
                        sum_4 =  (0.5-Y)*SX[8]*sum_6;

                        zz = (double)(sum_5*sum_5*(0.5+Y) + sum_6*sum_6*(0.5-Y))/SNR;
                        //printf("\nzz = %f sum_5 = %d  sum_6 = %d  KK =
%d",zz,sum_5,sum_6,KK);
                        doomy = *(HH3+BB);
                        LLL = randn(zz);
                        *(HH3+BB)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*cos(ran_phase) +
LLL)/Hf;
                        sum_r2 = sum_r2 + (*(HH3+BB)) - doomy;
                        doomy = *(HH4+BB);
                        LLL = randn(zz)/sqrt(2);
                        *(HH4+BB)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*sin(ran_phase) +
LLL)/Hf;
                        sum_i2 = sum_i2 + (*(HH4+BB)) - doomy;


/**********************VCC********************/
                        sum_34 = (sum_r1*sum_r2+ sum_i1*sum_i2)/Tc;
                        Townhat = Townhat - sum_34/(2*Power*1.001*n0);
                        forward_shift(SX);
                        KK++;

                        }

}

void initial(void)
{
                int sum_5,sum_6,BB,i;
                float  sum_3,sum_4;
                double zz,doomy,LLL,sum_34;
                for(i = 0; i < Hf; i++) reward_shift(SX);
                for (i = 0; i < Hf; i++)
                {
/**********************upper branch******************/
                        get_J_Y(Townhat,Town);

                        sum_5 = ((*(alpha+((20*K + i-1+J-Hf)%K)))-(*(alpha+((20*K + i+J-
Hf)%K))));
                        sum_6 = ((*(alpha+((20*K + i+J-Hf)%K)))-(*(alpha+((20*K + 1+ i +J -
Hf)%K))));
                        sum_3 =  (float)(0.5+Y)*SX[8]*sum_5;
                        sum_4 =  (float)(0.5-Y)*SX[8]*sum_6;


                        zz = (double)(sum_5*sum_5*(0.5+Y) + sum_6*sum_6*(0.5-Y))/SNR;
                //      printf("\nzz = %f sum_5 = %d  sum_6 = %d  KK =
%d",zz,sum_5,sum_6,KK);
                        //printf("\n doomy = %f",doomy);
```

```
                        LLL = randn(zz);
                        *(HH1+i)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*cos(ran_phase) +
LLL)/Hf;
                        sum_r1 = sum_r1 + (*(HH1+i));

                        LLL =  randn(zz);
                        *(HH2+i)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*sin(ran_phase) +
LLL)/Hf;
                        sum_i1 = sum_i1 + (*(HH2+i));

/****************lower branch**********************/
                        Townhat = Townhat + (127.75)*Tc;
                        get_J_Y(Townhat,Town);
                        Townhat = Townhat - (127.75)*Tc;
                        sum_5 = ((*(alpha+((20*K + i-1+J-Hf)%K)))-(*(alpha+((20*K + i+J -
Hf)%K))));
                        sum_6 = ((*(alpha+((20*K + i + J-Hf)%K)))-(*(alpha+((20*K + 1+i+J -
Hf)%K))));
                        sum_3 =  (0.5+Y)*SX[8]*sum_5;
                        sum_4 =  (0.5-Y)*SX[8]*sum_6;

                        zz = (double)(sum_5*sum_5*(0.5+Y) + sum_6*sum_6*(0.5-Y))/SNR;
                        //printf("\nzz = %f sum_5 = %d  sum_6 = %d  KK =
%d",zz,sum_5,sum_6,KK);

                        LLL = randn(zz);
                        *(HH3+i)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*cos(ran_phase) +
LLL)/Hf;
                        sum_r2 = sum_r2 + (*(HH3+i));

                        LLL = randn(zz);
                        *(HH4+i)= sqrt(2*Power)*Tc*((sum_3 + sum_4)*sin(ran_phase) +
LLL)/Hf;
                        sum_i2 = sum_i2 + (*(HH4+i));

                        forward_shift(SX);


                        }

}
void gen_alpha(void)
{
int shb[9] = {1,1,1,1,1,1,1,1,1},i,bsh[9],fsh[9],mm;
for (i = 0;i<511;i++)
        {
         *(alpha+i) = shb[8]*255;
         equal_shift(fsh,shb);
         equal_shift(bsh,shb);
         for(mm = 1;mm <=255;mm++)
                {
                        forward_shift(fsh);
                        reward_shift(bsh);
                        *(alpha+i) = *(alpha+i)+(fsh[8]*(255-mm)+bsh[8]*(255-mm));
                }
                printf("\n%d",*(alpha+i));
```

```
            forward_shift(shb);
          }

}

void  clearH(double far *H)
{
 int i;
 for ( i = 0 ; i < 8000; i++)
          *(H+i) = 0.0 ;
}
void set_shift(int sh1[9])
{
          int i;
          for(i = 0; i < 9 ; i++)
                    sh1[i] = 1;
}
```