C 实现 QPSK

```
#include "math.h"
#include "stdlib.h"
#include "stdio.h"
#include   "time.h"
#include   "malloc.h"
#include   "string.h"
#include   "iostream.h"

#define source_length 1000000
#define symbol_length source_length/2
#define SNR_start 1
#define SNR_end 10
#define SNR_step 1
#define PI 3.1415926
#define Coderate 1

typedef struct {
      double Rpart;
      double Ipart;
}complex;

 int source[source_length];//message(),modulate(),error()
 int change[source_length];//modulate()
 int resource[source_length];       //demodulate(),error()
 complex modulatesym[symbol_length];     //demodulate(),channel(),moudulate()
 int snr;//channel(),error()
int errorbit, errorsym;//error()
 double BER,SER;//error()

void message();
void modulate(int source[source_length]);
void channel(complex modulatesym[symbol_length],int snr);
void demodulate();
void error();

//随机信号产生
void message()
{
      int i;
      //以当前时间作为时间种子
```

```
        srand((unsigned)time(NULL));
        //产生 0,1 随机信号
        for(i=0;i<source_length;i++)
        { source[i]=rand()%2;
              //cout<<source[i];
        }
        //cout<<endl;
}

//调制
void modulate(int source[source_length])
{
        int i,j;
        //0->-1,1->1
    for(i=0;i<source_length;i++)
    {
        change[i]=1-2*source[i];

    }

      for(j=0;j<symbol_length;j++)
      {
            modulatesym[j].Rpart=change[2*j];//cout<<change[2*j];
            modulatesym[j].Ipart=change[2*j+1];//cout<<change[2*j+1];
      }
    // cout<<endl;
}

//调制信号通过信道
void channel(complex modulatesym[],int snr)
{
        long int j;
        double r1,r2;
        double amp,phase;
        double sn,SNR,noise[2];
        SNR=snr+10*log10((double)Coderate);
        sn=pow(10.0,SNR/10.0);
        for(j=0;j<symbol_length;j++)
        {
              r1=(double)rand()/RAND_MAX;
              r2=(double)rand()/RAND_MAX;
              if(r1<=1.0e-8)     r1=1.0e-8; //防止出现 log0 的操作
              phase=2.0*PI*r2;
              amp=sqrt(-log(r1)/sn);
```

```
                noise[0]=amp*cos(phase);
                noise[1]=amp*sin(phase);

modulatesym[j].Rpart=modulatesym[j].Rpart+noise[0];//cout<<modulatesym[j].Rpart;

modulatesym[j].Ipart=modulatesym[j].Ipart+noise[1];//cout<<modulatesym[j].Ipart;

            }
            //cout<<endl;
        }

        //解调
        void demodulate()
        {
            for(int j=0;j<symbol_length;j++)
            {
                if (modulatesym[j].Rpart>0)
                    resource[2*j]=0;
                else //if(modulatesym[j].Rpart<=0)
                     resource[2*j]=1;
            }
            for(int i=0;i<symbol_length;i++)
            {
                if (modulatesym[i].Ipart>0)
                     resource[2*i+1]=0;
                else //if(modulatesym[j].Ipart<=0)
                     resource[2*i+1]=1;
                }
        }

        void error()
        {
           long int i,j;
           errorbit=0;
           errorsym=0;

           for(i=0;i<source_length;i++)
           { if(resource[i]!=source[i])
                    errorbit++;
           }
           for(j=0;j<=symbol_length;j++)
           {
                 if(resource[2*j]!=source[2*j]||resource[2*j+1]!=source[2*j+1])
                      errorsym++;
```

```
    }

    BER=(double)errorbit/source_length;
    SER=(double)errorsym/symbol_length;
cout<<"snr="<<snr<<endl;
cout<<"source_length="<<source_length<<endl;
cout<<"symbol_length="<<symbol_length<<endl;
cout<<"errorbit="<<errorbit<<endl;
cout<<"errorsym="<<errorsym<<endl;
cout<<"BER="<<BER<<endl;
cout<<"SER="<<SER<<endl;
}

void main()
{
    for(snr=SNR_start;snr<=SNR_end;snr+=SNR_step)

    {
        message();

        modulate(source);
        channel(modulatesym,snr);
        demodulate();
        /*for(int i=0;i<source_length;i++)
            {
                cout<<resource[i];
            }
        cout<<endl;*/
        error();
    }
}
```

C++ 实现 QPSK

```
#include<iostream>
#include<time.h>
using namespace std;

#define source_length 1000000
#define symbol_length source_length/2
#define SNR_start 1
```

```
#define SNR_end 10
#define SNR_step 1
#define PI 3.1415926
#define Coderate 1

typedef struct {
	double Rpart;
	double Ipart;
}complex;

int source[source_length];
int change[source_length];
int resource[source_length];
complex modulatesym[symbol_length];
int snr;
int errorbit, errorsym;
double BER, SER;

//随机信号产生
void message()
{
	int i;
	//以当前时间作为时间种子
	srand((unsigned)time(NULL));
	//产生 0,1 随机信号
	for (i = 0; i<source_length; i++)
	{
		source[i] = rand() % 2;
		//cout<<source[i];
	}
	//cout<<endl;
}

//调制
void modulate(int source[source_length])
{
	int i, j;
	//0->-1,1->1
	for (i = 0; i<source_length; i++)
	{
		change[i] = 1 - 2 * source[i];
	}

	for (j = 0; j<symbol_length; j++)
```

```
    {
        modulatesym[j].Rpart = change[2 * j];//cout<<change[2*j];
        modulatesym[j].Ipart = change[2 * j + 1];//cout<<change[2*j+1];
    }
    // cout<<endl;
}

//解调
void demodulate()
{
    for (int j = 0; j<symbol_length; j++)
    {
        if (modulatesym[j].Rpart>0)
            resource[2 * j] = 0;
        else //if(modulatesym[j].Rpart<=0)
            resource[2 * j] = 1;
    }
    for (int i = 0; i<symbol_length; i++)
    {
        if (modulatesym[i].Ipart>0)
            resource[2 * i + 1] = 0;
        else //if(modulatesym[j].Ipart<=0)
            resource[2 * i + 1] = 1;
    }
}

//调制信号通过信道
void channel(complex modulatesym[], int snr)
{
    long int j;
    double r1, r2;
    double amp, phase;
    double sn, SNR, noise[2];
    SNR = snr + 10 * log10((double)Coderate);
    sn = pow(10.0, SNR / 10.0);
    for (j = 0; j<symbol_length; j++)
    {
        r1 = (double)rand() / RAND_MAX;
        r2 = (double)rand() / RAND_MAX;
        if (r1 <= 1.0e-8)       r1 = 1.0e-8; //防止出现 log0 的操作
        phase = 2.0*PI*r2;
        amp = sqrt(-log(r1) / sn);
        noise[0] = amp*cos(phase);
        noise[1] = amp*sin(phase);
```

```
                modulatesym[j].Rpart = modulatesym[j].Rpart +
noise[0];//cout<<modulatesym[j].Rpart;
                modulatesym[j].Ipart = modulatesym[j].Ipart +
noise[1];//cout<<modulatesym[j].Ipart;
            }
            //cout<<endl;
        }

        void error()
        {
            long int i, j;
            errorbit = 0;
            errorsym = 0;

            for (i = 0; i<source_length; i++)
            {
                if (resource[i] != source[i])
                    errorbit++;
            }
            for (j = 0; j <= symbol_length; j++)
            {
                if (resource[2 * j] != source[2 * j] || resource[2 * j + 1] != source[2 * j + 1])
                    errorsym++;
            }

            BER = (double)errorbit / source_length;
            SER = (double)errorsym / symbol_length;
            cout << "snr=" << snr << endl;
            cout << "source_length=" << source_length << endl;
            cout << "symbol_length=" << symbol_length << endl;
            cout << "errorbit=" << errorbit << endl;
            cout << "errorsym=" << errorsym << endl;
            cout << "BER=" << BER << endl;
            cout << "SER=" << SER << endl;
        }

        int main()
        {
            message();
            modulate(source);
            channel(modulatesym, snr);
            //cout << endl;
            demodulate();
            //for (int i = 0; i < source_length; i++)cout << resource[i];
```

```
    error();
    return 0;
}
```