回路システム基礎期末レポート

2231009　伊藤　大地

2024 年 1 月 3 日

1 ビット AD コンバータのシミュレーションを行った．入力信号には 1Hz の正弦波を用いており，サンプリング周波数 1kHz でオーバーサンプリングした．図 1 に入力信号の時間波形（1Hz の正弦波），図 2 に入力信号のスペクトル分布，図 3 に図 2 の低域部分を拡大したグラフを示す．

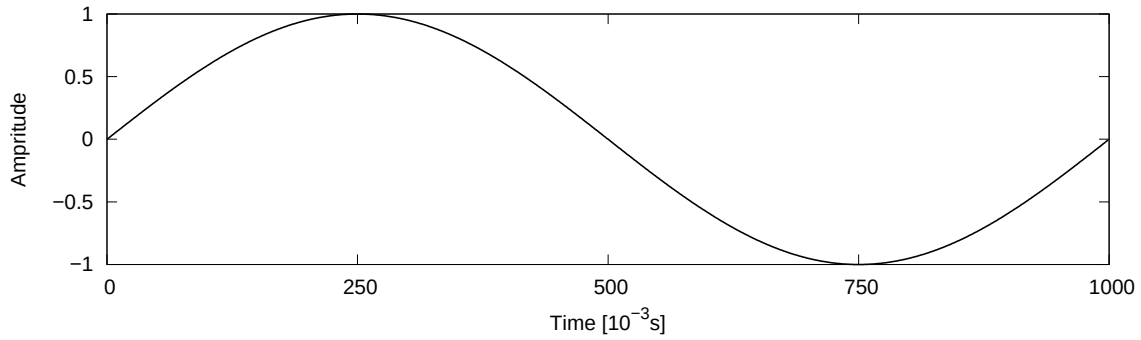

図 1: 入力信号の時間波形（1Hz の正弦波）

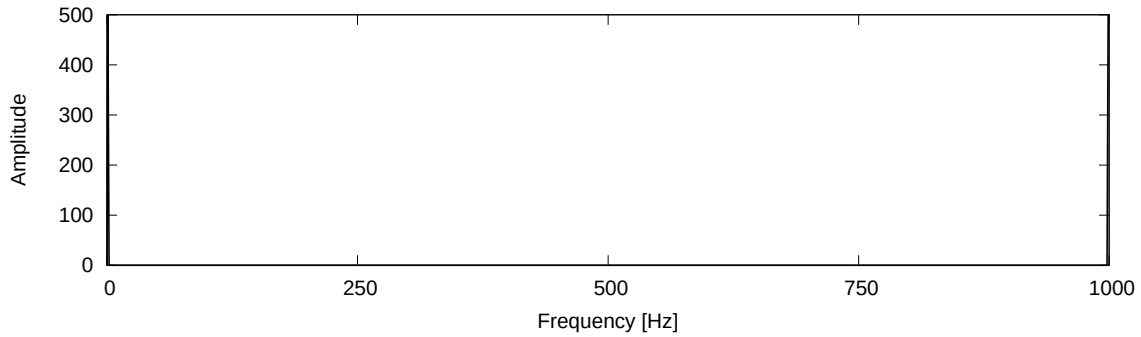

図 2: 入力信号のスペクトル分布

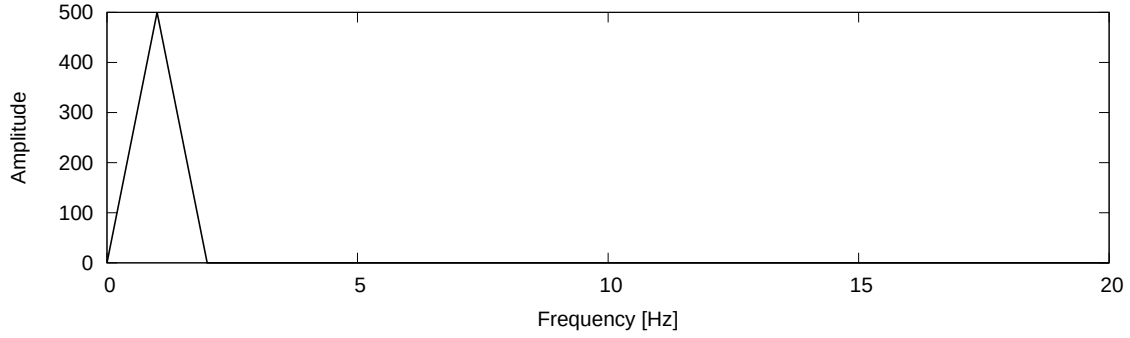

図 3: 図 2 の低域部分を拡大したグラフ

(1) Plot output signals and spectrums 1st and 2nd $\Delta\Sigma$ modulators.

(i) 1st $\Delta\Sigma$ modulators

入力信号に 1 次 $\Delta\Sigma$ 変調を行った結果を以下の図 4〜図 6 に示す．図 4 は 1 次 $\Delta\Sigma$ 変調信号の時間波形，図 5 はその信号のスペクトル分布，図 6 の低域部分を拡大したグラフである．

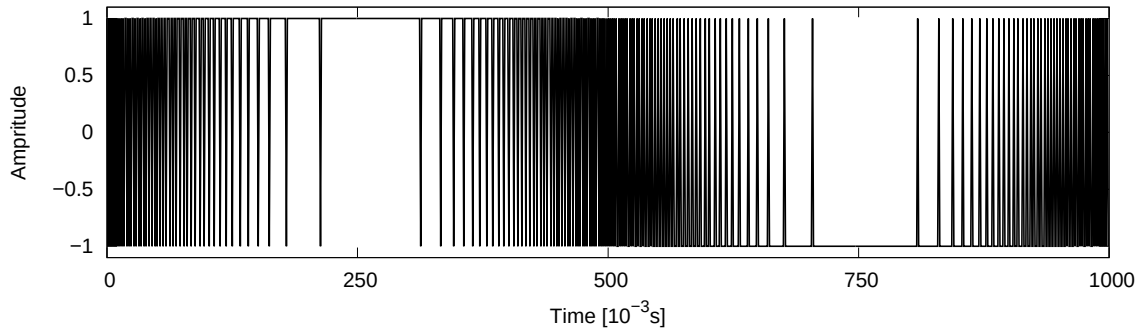

図 4: 1 次 $\Delta\Sigma$ 変調信号の時間波形

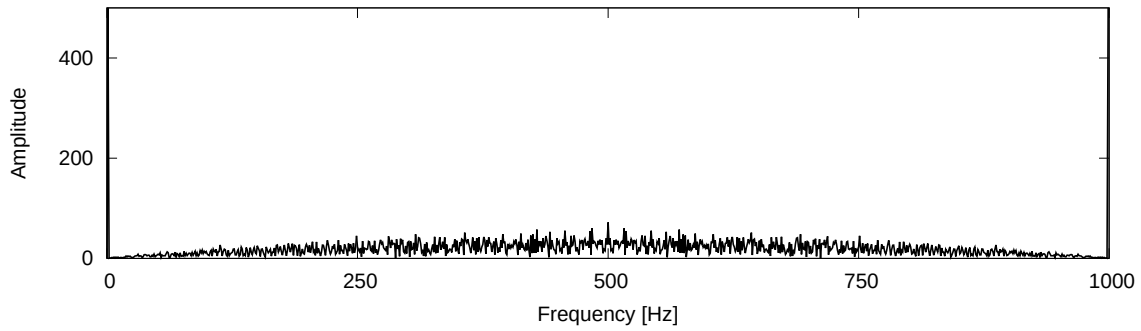

図 5: 1 次 $\Delta\Sigma$ 変調信号のスペクトル分布

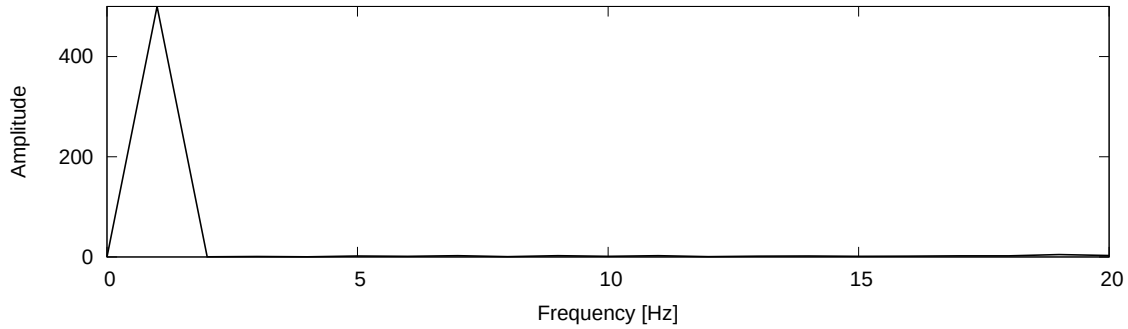

図 6: 図 5 の低域部分を拡大したグラフ

(ii) 2nd $\Delta\Sigma$ modulators

入力信号に 2 次 $\Delta\Sigma$ 変調を行った結果を以下の図 7〜図 9 に示す．図 7 は 2 次 $\Delta\Sigma$ 変調信号の時間波形，図 8 はその信号のスペクトル分布，図 9 は図 8 の低域部分を拡大したグラフである．

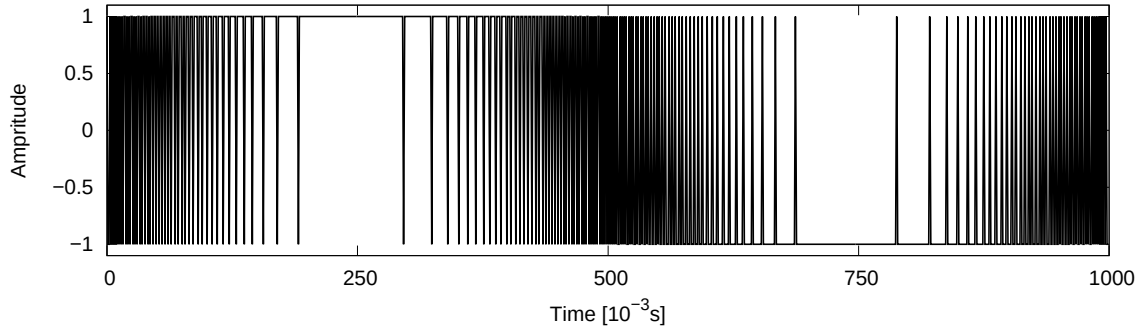

図 7: 2 次 ΔΣ 信号の時間波形

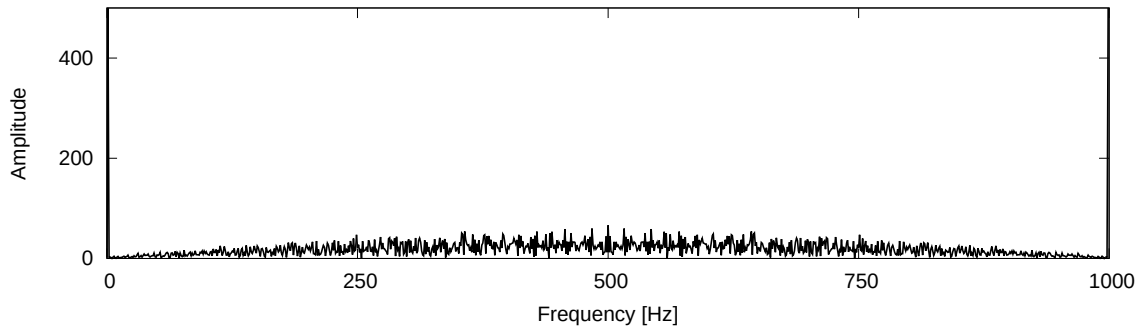

図 8: 2 次 ΔΣ 信号のスペクトル分布

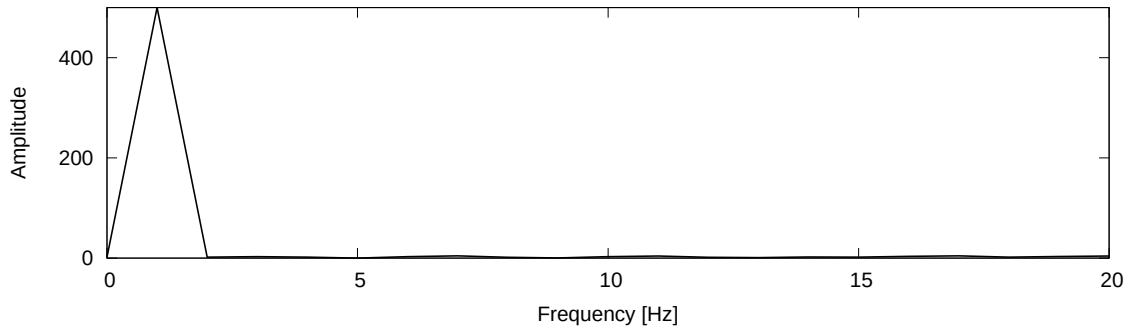

図 9: 図 8 の低域部分を拡大したグラフ

(2) Plot the low-pass filtered signals of each output.

ローパスフィルタとして 1 次のバタワースフィルタを用いた．また，カットオフ周波数は 5Hz に設定した．以下の図 10 にその周波数特性を示す．

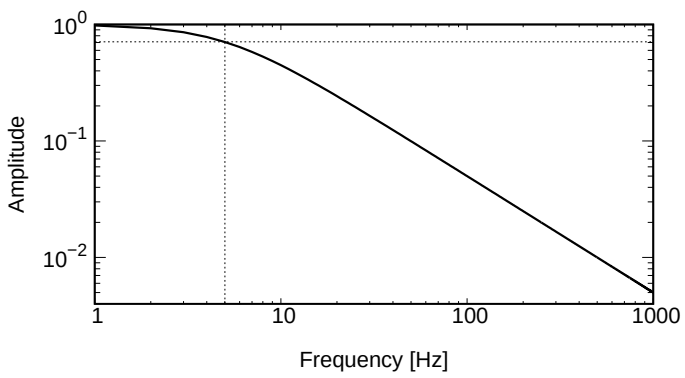

図 10: 1 次バタワース型ローパスフィルタの周波数特性

(i) 1st $\Delta\Sigma$ modulators

1 次 $\Delta\Sigma$ 変調信号に図 10 のローパスフィルタを適用した結果を以下の図 11〜図 13 に示す．図 11 はローパスフィルタを適用した 1 次 $\Delta\Sigma$ 変調信号の時間波形，図 12 はその信号のスペクトル分布，図 13 は図 12 の低域部分を拡大したグラフである．

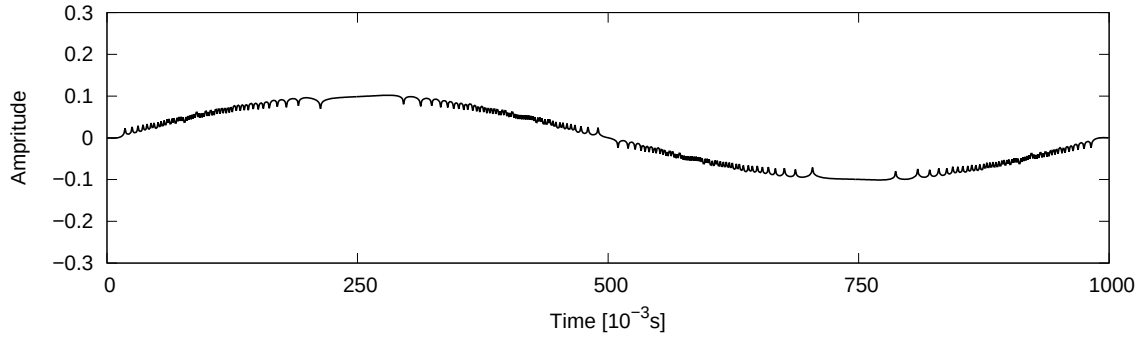

図 11: ローパスフィルタを適用した 1 次 $\Delta\Sigma$ 変調信号の時間波形

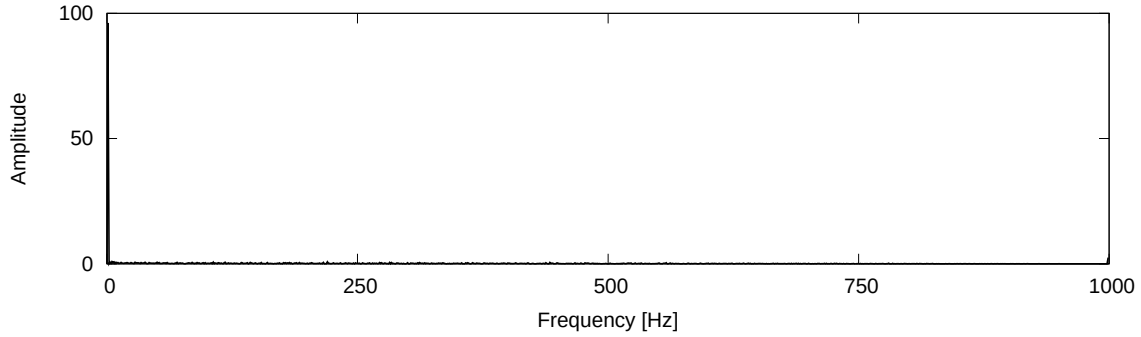

図 12: ローパスフィルタを適用した 1 次 $\Delta\Sigma$ 変調信号のスペクトル分布

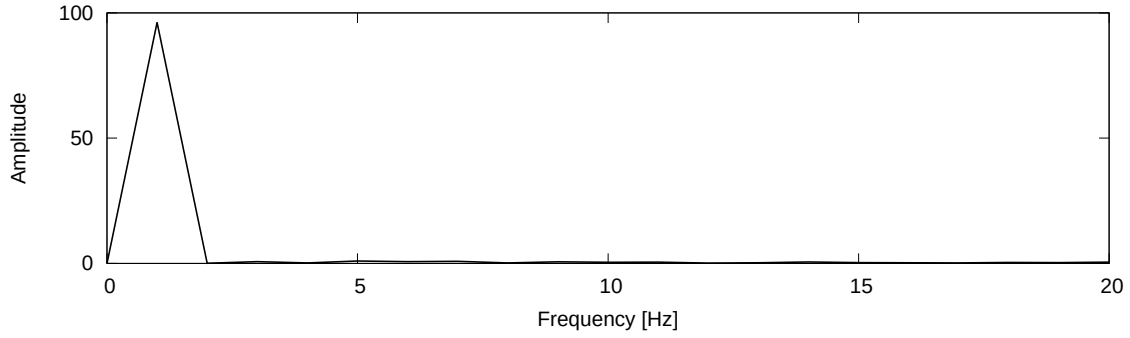

図 13: 図 12 の低域部分を拡大したグラフ

(ii) 2nd $\Delta\Sigma$ modulators

2 次 $\Delta\Sigma$ 変調信号に図 10 のローパスフィルタを適用した結果を以下の図 14〜図 16 に示す．図 14 はローパスフィルタを適用した 2 次 $\Delta\Sigma$ 変調信号の時間波形，図 15 はその信号のスペクトル分布，図 16 は図 15 の低域部分を拡大したグラフである．

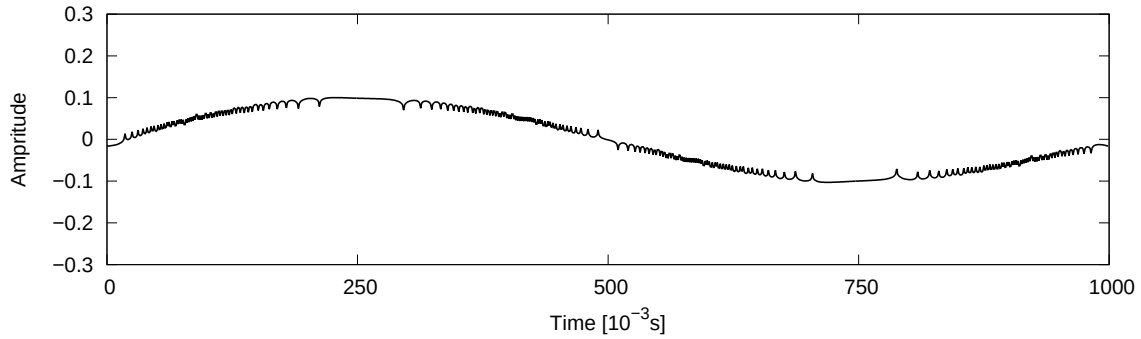

図 14: ローパスフィルタを適用した 2 次 ΔΣ 変調信号の時間波形

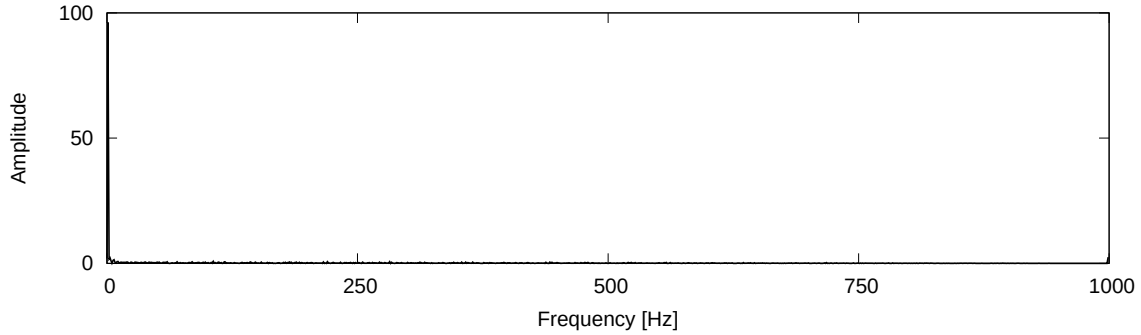

図 15: ローパスフィルタを適用した 2 次 ΔΣ 変調信号のスペクトル分布

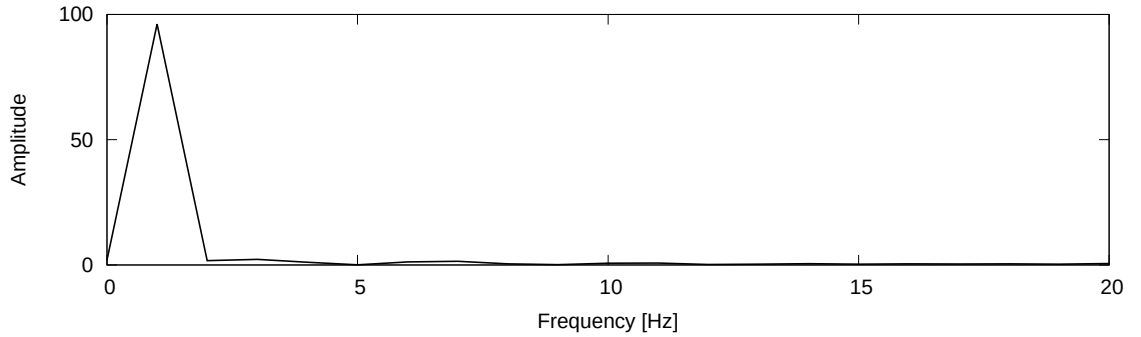

図 16: 図 15 の低域部分を拡大したグラフ

(3) Explain these results with exhibiting your created program.

はじめに ΔΣ 変調信号について述べる．図 4 と図 7 から，ΔΣ 変調により 1 ビット量子化されてデジタルに変換されていることがわかる．また図 5 と図 8 を見ると，図 2 には無かった 1Hz 以外の周波数成分がスペクトル全体に分散されて付加されていることがわかる．これは，ΔΣ 変調のフィードバックにより量子化誤差がスペクトル全体に拡散されているからであると考えられる．

次にローパスフィルタを適用した結果について述べる．図 11 と図 14 から，波形が若干歪んでいるがある程度正弦波に近い信号を出力していることがわかる．しかし信号の振幅が入力波形より小さくなっており，約 10 分の 1 になっている．また図 12 と図 15 を見ると，ΔΣ 変調により発生していた高周波成分をローパスフィルタによって低減できたことがわかる．これらの結果より，ΔΣ 変調で入力信号のエネルギーの一部がスペクトル全体に拡散され，その拡散されたエネルギーがローパスフィルタによって取り除かれるので，出力信号のエネルギー損失が発生したと考えられる．

作成したプログラムのソースコード

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

#define N 1000//サンプリング数
#define Wc 5//カットオフ周波数

/* LPF 関数（1 次，バタワース） */
void LPF(double *ReH, double *ImH){
  int k;
  double W;
  for(k=0;k<N;k++){
    W = (double)k/Wc;
    ReH[k] = ReH[k] / (1.0+W*W);
    ImH[k] = ImH[k] * (W/(1.0+W*W));
  }
}

int main(int argc, char *argv[]){
  double input[N];//入力信号
  double output1[N],output2[N];//出力信号
  double Re[N],Im[N];//DFT 後の実部成分と虚部成分
  double A;//LPF 伝達関数計算用
  double ReH[N],ImH[N];//LPF 伝達関数の実部成分と虚部成分
  double Z1,Z2;//遅延器出力信号
  double Qout;//1bit 量子化出力
  double tmp,tmp2;//格納用
  double Spec;//振幅スペクトル出力用
  FILE *fp;
  int i,j;
  double W = 2.0*M_PI/N;

  //LPF 伝達関数の振幅スペクトル計算
  for(i=0;i<N;i++){
    A = (double)i/Wc;
    ReH[i] = 1.0/(1.0+A*A);
    ImH[i] = (A/(1.0+A*A));
  }
  //LPF 伝達関数の振幅スペクトル出力
  fp = fopen("H_spec.txt", "wb");
  for(i=0; i<N; i++){
    Spec = sqrt(ReH[i]*ReH[i]+ImH[i]*ImH[i]);
    fprintf(fp, "%d %f\n", i, Spec);
  }

/*** 入力信号のサンプリング ***/
  //初期化
  for(i=0; i<N; i++){
    input[i] = 0.0;
    Re[i] = 0.0;
    Im[i] = 0.0;
    ReH[i] = 0.0;
    ImH[i] = 0.0;
  }
  //オーバーサンプリング
  for(i=0; i<N; i++){
```

作成したプログラムのソースコード

```
    input[i] = sin(W*i);
  }
  //入力信号の時間波形出力
  fp = fopen("Input_time.txt", "wb");
  for(i=0; i<N; i++){
    fprintf(fp, "%d %f\n", i, input[i]);
  }
  //入力信号を実部と虚部に分けて DFT
  for(i=0; i<N; i++){
    for(j=0; j<N; j++){
      Re[i] += input[j]*cos(W*i*j);
      Im[i] += -input[j]*sin(W*i*j);
    }
  }
  //入力信号の振幅スペクトル出力
  fp = fopen("Input_spec.txt", "wb");
  for(i=0; i<N; i++){
    Spec = sqrt(Re[i]*Re[i]+Im[i]*Im[i]);
    fprintf(fp, "%d %f\n", i, Spec);
  }

/*** １次のΔΣ変調 ***/
  //初期化
  for(i=0; i<N; i++){
    output1[i] = 0.0;
    Re[i] = 0.0;
    Im[i] = 0.0;
  }
  tmp = 0.0;
  //１次のΔΣ変調計算部分
  for(i=0; i<N; i++){
    //１つ前の出力を足す
    if(i!=0){
      tmp = input[i] + Z1;
    }
    //1bit 量子化
    if(tmp>=0.0){
      Qout = 1.0;
    }else if(tmp<0.0){
      Qout = -1.0;
    }
    //出力とフィードバック処理
    output1[i] = Qout;
    Z1 = tmp - output1[i];
  }
  //変調信号の時間波形出力
  fp = fopen("1stout_time.txt", "wb");
  for(i=0; i<N; i++){
    fprintf(fp, "%d %f\n", i, output1[i]);
  }
  //変調信号を実部と虚部に分けて DFT
  for(i=0; i<N; i++){
    for(j=0; j<N; j++){
      Re[i] += output1[j]*cos(W*i*j);
      Im[i] += -output1[j]*sin(W*i*j);
    }
  }
```

作成したプログラムのソースコード

```
  //変調信号の振幅スペクトル出力
  fp = fopen("1stout_spec.txt", "wb");
  for(i=0; i<N; i++){
    Spec = sqrt(Re[i]*Re[i]+Im[i]*Im[i]);
    fprintf(fp, "%d %f\n", i, Spec);
  }
  //LPF を適用
  LPF(Re, Im);
  //LPF を適用した後の振幅スペクトル出力
  fp = fopen("1stout_LPF_spec.txt", "wb");
  for(i=0; i<N; i++){
    Spec = sqrt(Re[i]*Re[i]+Im[i]*Im[i]);
    fprintf(fp, "%d %f\n", i, Spec);
  }
  //IDFT
  for(i=0;i<N;i++){
    output1[i] = 0.0;//初期化
    for(j=0;j<N;j++){
      output1[i] += Re[j]*cos(2*M_PI*j*i/N) - Im[j]*sin(2*M_PI*j*i/N);
    }
   output1[i] /= N;
  }
  //LPF した後の時間波形出力
  fp = fopen("1stout_LPF_time.txt", "wb");
  for(i=0; i<N; i++){
    fprintf(fp, "%d %f\n", i, output1[i]);
  }

/*** 2 次のΔΣ変調 ***/
  //初期化
  for(i=0; i<N; i++){
    output2[i] = 0.0;
    Re[i] = 0.0;
    Im[i] = 0.0;
  }
  tmp = 0.0;
  tmp2 = 0.0;
  //2 次のΔΣ変調計算部分
  for(i=0; i<N; i++){
    //2 つ前の出力を足す
    if(i>=2){
      tmp = input[i] - Z2;
    }
    //1 つ前の出力を足す
    if(i>=1){
      tmp2 = tmp + 2.0*Z1;
    }
    //1bit 量子化
    if(tmp2>=0.0){
      Qout = 1.0;
    }else if(tmp2<0.0){
      Qout = -1.0;
    }
    //出力とフィードバック処理
    output2[i] = Qout;
    Z1 = tmp2 - output2[i];
    Z2 = Z1;
```

作成したプログラムのソースコード

```
  }
  //変調信号の時間波形出力
  fp = fopen("2ndout_time.txt", "wb");
  for(i=0; i<N; i++){
    fprintf(fp, "%d %f\n", i, output2[i]);
  }
  //変調信号を実部と虚部に分けて DFT
  for(i=0; i<N; i++){
    for(j=0; j<N; j++){
      Re[i] += output2[j]*cos(W*i*j);
      Im[i] += -output2[j]*sin(W*i*j);
    }
  }
  //変調信号の振幅スペクトル出力
  fp = fopen("2ndout_spec.txt", "wb");
  for(i=0; i<N; i++){
    Spec = sqrt(Re[i]*Re[i]+Im[i]*Im[i]);
    fprintf(fp, "%d %f\n", i, Spec);
  }
  //LPF を適用
  LPF(Re, Im);
  //LPF した後の振幅スペクトル出力
  fp = fopen("2ndout_LPF_spec.txt", "wb");
  for(i=0; i<N; i++){
    Spec = sqrt(Re[i]*Re[i]+Im[i]*Im[i]);
    fprintf(fp, "%d %f\n", i, Spec);
  }
  //IDFT
  for(i=0;i<N;i++){
    output2[i] = 0.0;//初期化
    for(j=0;j<N;j++){
      output2[i] += Re[j]*cos(2*M_PI*j*i/N) - Im[j]*sin(2*M_PI*j*i/N);
    }
   output2[i] /= N;
  }
  //LPF した後の時間波形出力
  fp = fopen("2ndout_LPF_time.txt", "wb");
  for(i=0; i<N; i++){
    fprintf(fp, "%d %f\n", i, output2[i]);
  }
fclose(fp);
}
```