

2007年度 卒業研究レポート

# Javaによる音のFourier解析

明治大学 理工学部 数学科

一木裕貴

2008年3月5日

# 目次

第1章	はじめに	2
1.1	目標	2
第2章	実際の音に対する DFT	3
2.1	Fourier 級数	3
2.2	離散 Fourier 変換	3
2.3	ヴァイオリンの音の波形	4
2.4	ヴァイオリンの音に対する DFT	5
第3章	大浦 FFT の書き換え	7
3.1	はじめに	7
3.2	C から Java への移植	7
3.3	テストプログラムの実行	11
第4章	音の読み込み方	14
4.1	はじめに	14
4.2	使い方	14
4.3	小さなバグ	14
付録 A	ソースプログラム	16
A.1	fft4gRead.java	16
A.2	ReadWave.java	41

# 第1章 はじめに

## 1.1 目標

波動方程式

$$\frac{1}{c^2}u_{tt} = u_{xx} \quad (t > 0, x \in (0, 1))$$

と初期条件

$$u(x, 0) = \phi(x) \quad (x \in [0, 1]),$$

$$u_t(x, 0) = \psi(x) \quad (x \in [0, 1])$$

それといずれかの境界条件

$$(\text{Dirichlet 境界条件}) \quad u(0, t) = u(1, t) = 0 \quad (t > 0)$$

$$(\text{Neumann 境界条件}) \quad u_x(0, t) = u_x(1, t) = 0 \quad (t > 0)$$

を課した初期値境界値問題の解の音を聴くことを最初の目標としましたが、これは出来ていません。

今回行ったのは実際の楽器の音を FFT で分析することです。

## 第2章 実際の音に対するDFT

### 2.1 Fourier 級数

離散 Fourier 変換を説明する為の準備として Fourier 級数の復習をする。  
適度に滑らかな周期  $T$  の関数  $u : \mathbb{R} \rightarrow \mathbb{C}$  は次のようにフーリエ級数展開できる。

$$u(t) = \sum_{n=-\infty}^{\infty} c_n \exp\left(\frac{2\pi i n t}{T}\right), \quad i = \sqrt{-1}$$

ただし、 $c_n$  は次式で定められるフーリエ係数とする。

$$c_n = \frac{1}{T} \int_0^T u(t) \exp\left(-\frac{2\pi i n t}{T}\right) dt.$$

### 2.2 離散 Fourier 変換

$u$  が周期  $T$  の関数であるとする。  $N \in \mathbb{N}$  を固定して

$$\omega = \omega_N := \exp\left(\frac{2\pi i}{N}\right)$$

とおく。  $\omega$  は 1 の原始  $N$  乗根である。さらに  $j \in \mathbb{Z}$  に対して

$$t_j := j \frac{T}{N}, \quad u_j := u(t_j)$$

とおく。  $u_j$  は周期  $N$  の周期数列である。フーリエ係数  $c_n$  の定義式の定積分を台形公式で近似したものを離散フーリエ係数  $C_n$  とおくと、

$$C_n = \frac{1}{T} \sum_{j=0}^{N-1} u(t_j) \exp\left(\frac{-2\pi i n}{T} t_j\right) \frac{T}{N} = \frac{1}{N} \sum_{j=0}^{N-1} \omega^{-jn} u_j$$

が成り立つ。  $C_n$  も  $n$  について周期  $N$  の数列である。  $\{u_j\} \mapsto \{C_n\}$  に変換することを離散 Fourier 変換 (Discrete Fourier Transform, DFT) と言う。

逆に

$$u_n = \sum_{j=0}^{N-1} C_n \omega^{jn}$$

と言う式も成り立つ。  $\{C_n\} \mapsto \{u_j\}$  に変換することを逆離散 Fourier 変換と言う。

台形公式とは

$$\int_0^T U(t)dt \doteq \left( \frac{1}{2}U(0) + \sum_{j=1}^{N-1} U(t_j) + \frac{1}{2}U(T) \right) \frac{T}{N}$$

と近似するものである。これは小区間  $[t_j, t_{j+1}]$  で次の近似をしたものである。

$$\int_{t_j}^{t_{j+1}} U(t)dt \doteq \frac{1}{2} (U(t_j) + U(t_{j+1})) \frac{T}{N}$$

周期関数では  $U(0) = U(T)$  なので、台形公式は

$$\int_0^T U(t)dt \doteq \sum_{j=0}^{N-1} U(t_j) \frac{T}{N}$$

となる。周期関数の1周期の積分は台形公式で非常に高精度に求まることが知られている。

## 2.3 ヴァイオリンの音の波形

Windows のサウンドレコーダーを使用し録音したデータ、サンプリング周波数 44.1kHz・量子化ビット数 16 ビット・ステレオのヴァイオリンのドの音を使う。

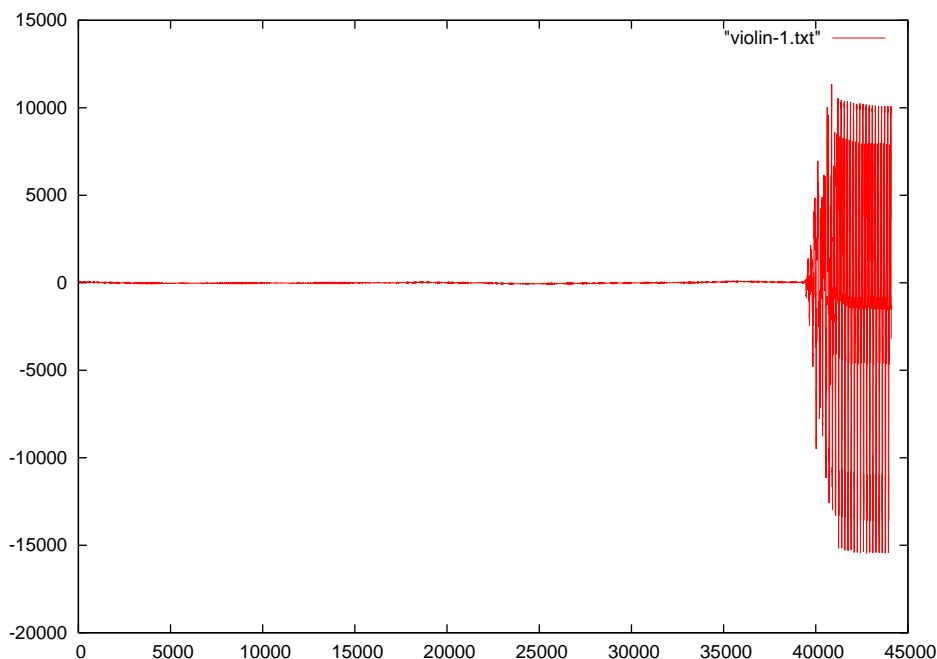
サンプリング周波数 = 1 秒間に何回データを記録するか。

この場合は 1 秒間に 44100 個のデータが入っている。

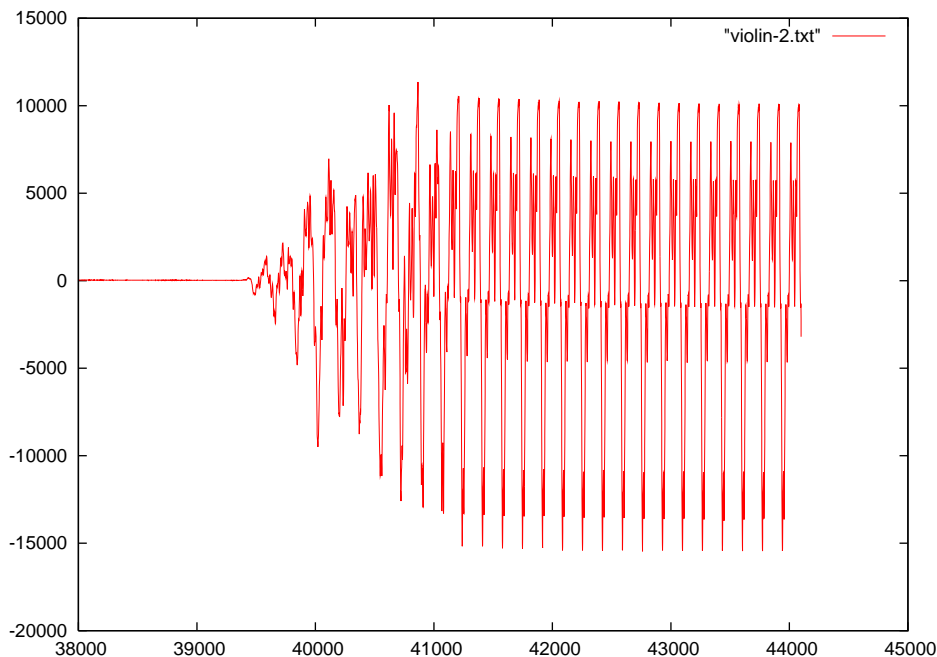
量子化ビット数 = 測定に使用する目盛りの細かさ。

この場合は  $-2^{15} \sim 2^{15} - 1$  の 65536 段階で表現されている。

最初の 1 秒間の波形を表示する。



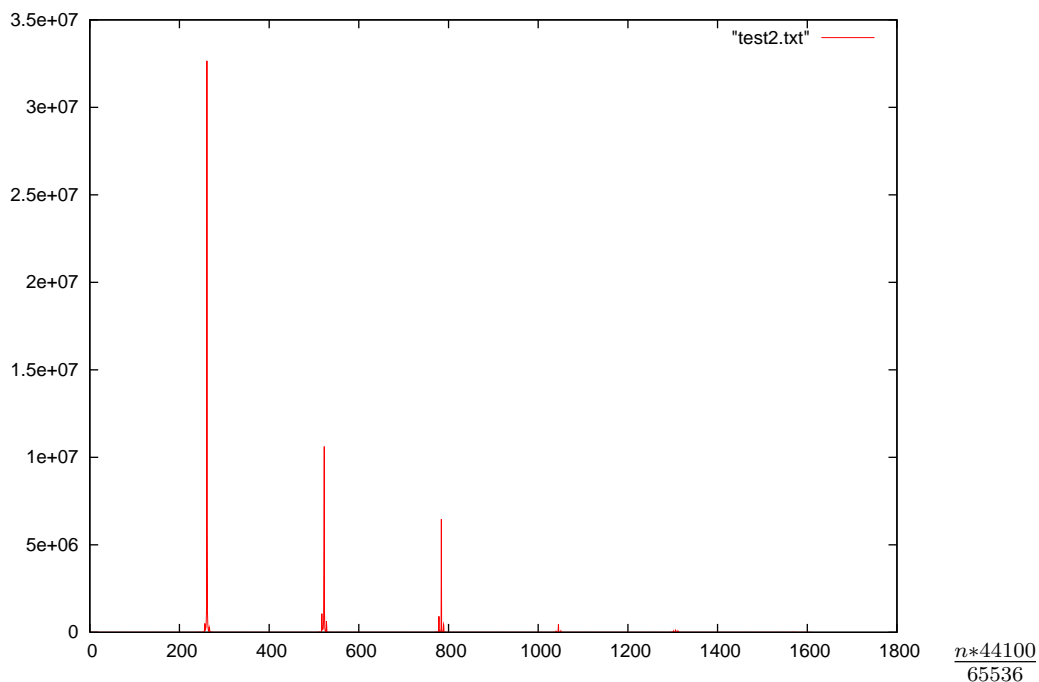
この図を見れば最初の方は振幅が小さいことが分かる。これは録音し始めて実際に弾くまでの間だと思われる。よってその部分を削除してみる。



すると振幅が 39500 番目ぐらいから大きくなっているのが分かる。ゆえに、39500 ~ のデータを使って DFT することにする。

## 2.4 ヴァイオリンの音に対する DFT

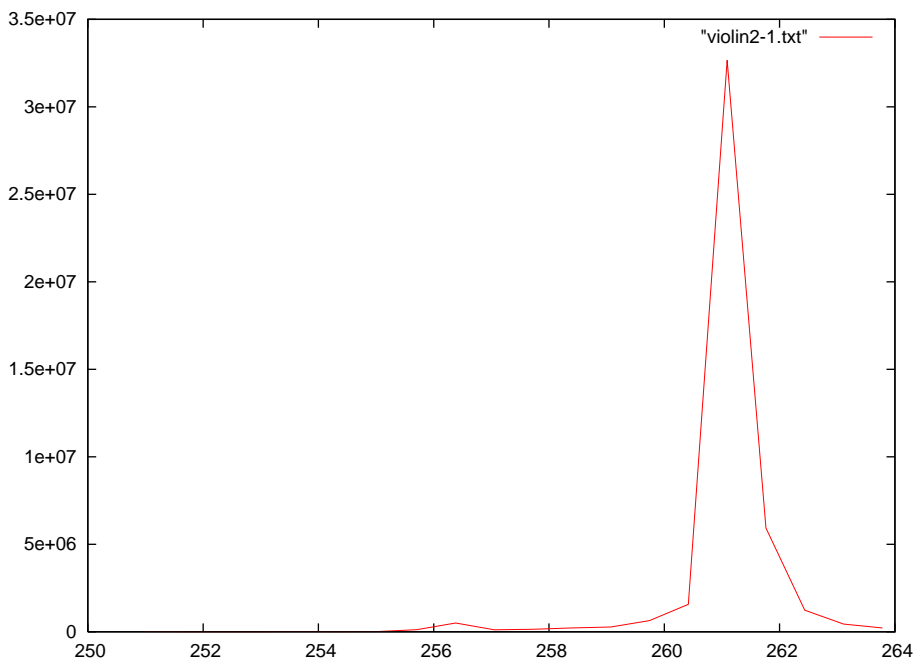
プログラムの都合上<sup>1</sup>、音が鳴り始めてから 65536 個のデータ ( $T = \frac{65536}{44100}$  秒分のデータ) を DFT し、2400 個分の  $|C_n|^2$  をグラフ化したものが次の図である。



<sup>1</sup>大浦 FFT では、データの個数が 2 のべき乗でないといけない。

$C_1$  は  $f_1 = 1/T = \frac{44100}{65536}$  Hz に対応している。 $|C_n|$  は、 $n = 388$  のときに最大となっているが、対応する周波数は、 $388f_1 = 388 \cdot \frac{44100}{65536} = 261.090\dots$  Hz である。グラフの横軸には、直接周波数を読み取れるように  $\frac{n * 44100}{65536}$  の値を表示してある。

この図のピークのあたりを拡大してみる。



はっきり約 261 Hz でピークを迎えていることが分かる。これはドの音の周波数 (261Hz) に相当する。よって、DFT すれば音が分からなくても高さが読み取れる。

## 第3章 大浦FFTの書き換え

### 3.1 はじめに

今回音の分析をするにあたって、Ooura's Mathematical Software Packages (大浦 [1]) にある「汎用 FFT (高速 フーリエ/コサイン/サイン 変換) パッケージ」を使用しています。これはCで書かれているのでJavaに変換しています。

ここではその際の書き換えの手順を追っていきます。

### 3.2 C Javaへの移植

まずは大まかにJavaへ変換していきます。CとJavaはかなり同じに書けるのでこの作業は楽です。ここからは例を挙げながら説明していきます。

```
1 void makewt(int nw, int *ip, double *w)
2 {
3     void bitrv2(int n, int *ip, double *a);
4     int j, nwh;
5     double delta, x, y;
6
7     ip[0] = nw;
8     ip[1] = 1;
9     if (nw > 2) {
10        nwh = nw >> 1;
11        delta = atan(1.0) / nwh;
12        w[0] = 1;
13        w[1] = 0;
14        w[nwh] = cos(delta * nwh);
15        w[nwh + 1] = w[nwh];
16        if (nwh > 2) {
17            for (j = 2; j < nwh; j += 2) {
18                x = cos(delta * j);
19                y = sin(delta * j);
20                w[j] = x;
21                w[j + 1] = y;
22                w[nw - j] = y;
23                w[nw - j + 1] = x;
24            }
25            bitrv2(nw, ip + 2, w);
26        }
27    }
28 }
```

これは元のプログラムの一部です、修正が必要な行は1・3・11・14・18・19・25行目になります。



```

1 private static void makewt(int nw, int [] ip, double [] w)
2 {
3     int j, nwh;
4     double delta, x, y;
5
6     ip[0] = nw;
7     ip[1] = 1;
8     if (nw > 2) {
9         nwh = nw >> 1;
10        delta = Math.atan(1.0) / nwh;
11        w[0] = 1;
12        w[1] = 0;
13        w[nwh] = Math.cos(delta * nwh);
14        w[nwh + 1] = w[nwh];
15        if (nwh > 2) {
16            for (j = 2; j < nwh; j += 2) {
17                x = Math.cos(delta * j);
18                y = Math.sin(delta * j);
19                w[j] = x;
20                w[j + 1] = y;
21                w[nw - j] = y;
22                w[nw - j + 1] = x;
23            }
24            bitrv2a(nw, ip, w);
25        }
26    }
27 }

```

と、このような形に書き直します。

よく見ると Java24 行目の部分がかかなり変わっていることに気づくと思います。元は ip+2 となっていて、これは配列の番号をずらすと言う作業なのですが、Java では使えません。よって、新しく関数を定義する必要があります。

元の bitrv2 というのは

```

1 void bitrv2(int n, int *ip, double *a)
2 {
3     int j, j1, k, k1, l, m, m2;
4     double xr, xi, yr, yi;
5
6     ip[0] = 0;
7     l = n;
8     m = 1;
9     while ((m << 3) < l) {
10        l >>= 1;
11        for (j = 0; j < m; j++) {
12            ip[m + j] = ip[j] + l;
13        }
14        m <<= 1;
15    }
16    m2 = 2 * m;
17    if ((m << 3) == 1) {
18        for (k = 0; k < m; k++) {
19            for (j = 0; j < k; j++) {
20                j1 = 2 * j + ip[k];
21                k1 = 2 * k + ip[j];
22                xr = a[j1];

```

```

23         xi = a[j1 + 1];
24         yr = a[k1];
25         yi = a[k1 + 1];
26         中略

```

と言うものです。ip以外の部分は同じなので、元の bitrv2 をコピーして使います。その際名前を変えないといけないのでここでは bitrv2a とします。ipの部分に2を加えるという作業を6・12・20・21行目に行います。

```

1  private static void bitrv2a(int n, int [] ip, double [] a)
2  {
3      int j, j1, k, k1, l, m, m2;
4      double xr, xi, yr, yi;
5
6      ip[2] = 0;
7      l = n;
8      m = 1;
9      while ((m << 3) < l) {
10         l >>= 1;
11         for (j = 0; j < m; j++) {
12             ip[m + j + 2] = ip[j + 2] + 1;
13         }
14         m <<= 1;
15     }
16     m2 = 2 * m;
17     if ((m << 3) == 1) {
18         for (k = 0; k < m; k++) {
19             for (j = 0; j < k; j++) {
20                 j1 = 2 * j + ip[k + 2];
21                 k1 = 2 * k + ip[j + 2];
22                 xr = a[j1];
23                 xi = a[j1 + 1];
24                 yr = a[k1];
25                 yi = a[k1 + 1];
26                 中略

```

この様に新しく関数を定義すれば問題なく動きます。  
しかし、+の部分を修正していくと問題に当たります。

```

1  void rdft(int n, int isgn, double *a, int *ip, double *w)
2  {
3      void makewt(int nw, int *ip, double *w);
4      void makect(int nc, int *ip, double *c);
5      void bitrv2(int n, int *ip, double *a);
6      void cftfsub(int n, double *a, double *w);
7      void cftbsub(int n, double *a, double *w);
8      void rftfsub(int n, double *a, int nc, double *c);
9      void rftbsub(int n, double *a, int nc, double *c);
10     int nw, nc;
11     double xi;
12
13     nw = ip[0];
14     if (n > (nw << 2)) {
15         nw = n >> 2;
16         makewt(nw, ip, w);
17     }

```

```

18     nc = ip[1];
19     if (n > (nc << 2)) {
20         nc = n >> 2;
21         makeect(nc, ip, w + nw);
22     }
23     中略

```

21 行目です。+の値が数値じゃないので今までのようにやるとエラーが起こります。そこで今回は少し変えて修正します。

元の makeect は

```

1 void makeect(int nc, int *ip, double *c)
2 {
3     int j, nch;
4     double delta;
5
6     ip[1] = nc;
7     if (nc > 1) {
8         nch = nc >> 1;
9         delta = atan(1.0) / nch;
10        c[0] = cos(delta * nch);
11        c[nch] = 0.5 * c[0];
12        for (j = 1; j < nch; j++) {
13            c[j] = 0.5 * cos(delta * j);
14            c[nc - j] = 0.5 * sin(delta * j);
15        }
16    }
17 }

```

wの部分に nw を加えると言う操作を行うのですが、元のプログラムには w がありません。この場合は c が w に対応しているので c に nw を加えることになります。

```

1 private static void makeect2(int nc, int [] ip, double [] c, int nw)
2 {
3     int j, nch;
4     double delta;
5
6     ip[1] = nc;
7     if (nc > 1) {
8         nch = nc >> 1;
9         delta = Math.atan(1.0) / nch;
10        c[nw] = Math.cos(delta * nch);
11        c[nch + nw] = 0.5 * c[nw];
12        for (j = 1; j < nch; j++) {
13            c[j + nw] = 0.5 * Math.cos(delta * j);
14            c[nc - j + nw] = 0.5 * Math.sin(delta * j);
15        }
16    }
17 }

```

ここでいつもと違うのが1行目、ひとつ定義が増えています。+の部分の数値を読むようにしてあります。こうする事によりエラーがなくなります。

実際に使うときも

```

1 private static void rdft(int n, int isgn, double [] a, int [] ip, double [] w)
2 {
3     int nw, nc;
4     double xi;
5
6     nw = ip[0];
7     if (n > (nw << 2)) {
8         nw = n >> 2;
9         makewt(nw, ip, w);
10    }
11    nc = ip[1];
12    if (n > (nc << 2)) {
13        nc = n >> 2;
14        makect2(nc, ip, w, nw);
15    }
16    中略

```

14 行目のように+の値の部分を一箇所増やして使います。  
この様な作業を全てに行っていけば Java への書き換えが出来ます。

### 3.3 テストプログラムの実行

今まで書き直してきましたが、それが正しくかつエラーが無いかは分かりません。そこで元のプログラムにあったテストプログラムを Java でも実行してみます。

```

1 private static void putdata(int nini, int nend, double [] a, double rnd)
2 {
3     int j;
4
5     for(j = nini; j <= nend; j++){
6         a[j] = rnd;
7     }
8 }
9
10 private static double errorcheck(int nini, int nend, double scale, double [] a, double rnd)
11 {
12     int j;
13     double err, e;
14     err = 0;
15
16     for(j = nini; j <= nend; j++){
17         e = rnd - a[j] * scale;
18         if(Math.abs(e) >= err){
19             err = Math.abs(e);
20         }
21     }
22     return err;
23 }
24
25 public static void main(String[] args){
26     int n, NMAX, NMAXSQRT;
27     double err, rnd, p;
28     int [] ip;
29     double [] a, w, t;

```

```

30
31     NMAX = 8192;
32     NMAXSQRT = 64;
33
34     ip = new int [NMAXSQRT + 2];
35     a = new double [NMAX + 1];
36     w = new double [NMAX * 5 / 4];
37     t = new double [NMAX / 2 + 1];
38
39     n = 1024;
40     p = Math.random();
41     rnd = ((p * 7141 + 54773 ) % 259200) * (1.0/259200.0);
42
43     ip[0] = 0;
44
45     putdata(0, n-1, a, rnd);
46     cdft(n, 1, a, ip, w);
47     cdft(n, -1, a, ip, w);
48     err = errorcheck(0, n-1, 2.0/n, a, rnd);
49     System.out.printf("cdft err = %g\n", err);
50
51     putdata(0, n-1, a, rnd);
52     rdft(n, 1, a, ip, w);
53     rdft(n, -1, a, ip, w);
54     err = errorcheck(0, n-1, 2.0/n, a, rnd);
55     System.out.printf("rdft err = %g\n", err);
56
57     putdata(0, n-1, a, rnd);
58     ddct(n, 1, a, ip, w);
59     ddct(n, -1, a, ip, w);
60     a[0] *= 0.5;
61     err = errorcheck(0, n-1, 2.0/n, a, rnd);
62     System.out.printf("ddct err = %g\n", err);
63
64     putdata(0, n-1, a, rnd);
65     ddst(n, 1, a, ip, w);
66     ddst(n, -1, a, ip, w);
67     a[0] *= 0.5;
68     err = errorcheck(0, n-1, 2.0/n, a, rnd);
69     System.out.printf("ddst err = %g\n", err);
70
71     putdata(0, n, a, rnd);
72     a[0] *= 0.5;
73     a[n] *= 0.5;
74     dfct(n, a, t, ip, w);
75     a[0] *= 0.5;
76     a[n] *= 0.5;
77     dfct(n, a, t, ip, w);
78     err = errorcheck(0, n, 2.0/n, a, rnd);
79     System.out.printf("dfct err = %g\n", err);
80
81     putdata(1, n-1, a, rnd);
82     dfst(n, a, t, ip, w);
83     dfst(n, a, t, ip, w);
84     err = errorcheck(1, n-1, 2.0/n, a, rnd);
85     System.out.printf("dfst err = %g\n", err);
86 }

```

内容はランダムで値をとり、各種変換を行いもとに戻したときに、一番誤差の大きい値を表示するものです。

```
1  cdft err = 0.00000
2  rdft err = 0.00000
3  ddct err = 2.22045e-16
4  ddst err = 2.49800e-16
5  dfct err = 0.00000
6  dfst err = 2.22045e-16
```

結果は上記のようになりほぼ誤差が無いことが分かります。  
よってC Java と移植できました。

## 第4章 音の読み込み方

### 4.1 はじめに

今回音のデータを読み込む時に桂田先生作の ReadWave.java (A.2 参照) というプログラムを使用しています。ここではプログラムの説明をしていきます。

### 4.2 使い方

次のようにコンパイル・実行します。

```
knoppix$ javac ReadWave.java
knoppix$ java ReadWave wave ファイル名
```

こうすると wave ファイルのデータがターミナル書き出されます。このままだと読み込んだデータを使いつらいので

```
knoppix$ java ReadWave wave ファイル名 > 適当な txt ファイル名
```

と実行します。こうする事によりデータが txt ファイルに書き出され、グラフ化することが楽になります。このデータに番号をつけて表示したものが2章最初の図となります。

### 4.3 小さなバグ

最初はデータが正しくないということが起こりました。これは Java の Byte が符号付きであることが原因でした。

```
74 int nBytesWritten = line.write(abData, 0, nBytesRead);
75 for (int i = 0; i < nBytesRead; i += 4) {
76     short left, right;
77     left = (short)(abData[i] | (abData[i+1] << 8));
78     right = (short)(abData[i+2] | (abData[i+3] << 8));
79     System.out.println("" + left + " " + right);
80 }
81 中略
```

元々はこの様なプログラムでしたが、このままでは値が合いません。次のように4・5行目に書き加えることにより問題は解決しました。

```
74 int nBytesWritten = line.write(abData, 0, nBytesRead);
75 for (int i = 0; i < nBytesRead; i += 4) {
76     short left, right;
77     left = (short)(abData[i] & 0xff | (abData[i+1] << 8));
78     right = (short)(abData[i+2] & 0xff | (abData[i+3] << 8));
79     System.out.println("" + left + " " + right);
80 }
81 中略
```



# 付録A ソースプログラム

## A.1 fft4gRead.java

このプログラムは、Ooura's Mathematical Software Packages にある「汎用 FFT (高速 フーリエ/コサイン/サイン 変換) パッケージ」を Java に変換して使用しています。

次のようにしてコンパイル・実行します。

```
knoppix$ javac fft4gRead.java
knoppix$ java ft4gRead Wave ファイル名 > 適当な txt ファイル名
```

```
1  import java.io.*;
2  import java.io.File;
3  import javax.sound.sampled.*;
4
5  public class fft4gRead {
6      private static void cdft(int n, int isgn, double [] a, int [] ip, double [] w)
7      {
8          if (n > (ip[0] << 2)) {
9              makewt(n >> 2, ip, w);
10         }
11         if (n > 4) {
12             if (isgn >= 0) {
13                 bitrv2a(n, ip, a);
14                 cftfsub(n, a, w);
15             } else {
16                 bitrv2conj2(n, ip, a);
17                 cftbsub(n, a, w);
18             }
19         } else if (n == 4) {
20             cftfsub(n, a, w);
21         }
22     }
23
24     private static void rdft(int n, int isgn, double [] a, int [] ip, double [] w)
25     {
26         int nw, nc;
27         double xi;
28
29         nw = ip[0];
30         if (n > (nw << 2)) {
31             nw = n >> 2;
32             makewt(nw, ip, w);
33         }
34         nc = ip[1];
35         if (n > (nc << 2)) {
36             nc = n >> 2;
```

```

37     makeect2(nc, ip, w, nw);
38 }
39 if (isgn >= 0) {
40     if (n > 4) {
41         bitrv2a(n, ip, a);
42         cftfsub(n, a, w);
43         rftfsub2(n, a, nc, w, nw);
44     } else if (n == 4) {
45         cftfsub(n, a, w);
46     }
47     xi = a[0] - a[1];
48     a[0] += a[1];
49     a[1] = xi;
50 } else {
51     a[1] = 0.5 * (a[0] - a[1]);
52     a[0] -= a[1];
53     if (n > 4) {
54         rftbsub2(n, a, nc, w, nw);
55         bitrv2a(n, ip, a);
56         cftbsub(n, a, w);
57     } else if (n == 4) {
58         cftfsub(n, a, w);
59     }
60 }
61 }
62
63 private static void ddct(int n, int isgn, double [] a, int [] ip, double [] w)
64 {
65     int j, nw, nc;
66     double xr;
67
68     nw = ip[0];
69     if (n > (nw << 2)) {
70         nw = n >> 2;
71         makewt(nw, ip, w);
72     }
73     nc = ip[1];
74     if (n > nc) {
75         nc = n;
76         makeect2(nc, ip, w, nw);
77     }
78     if (isgn < 0) {
79         xr = a[n - 1];
80         for (j = n - 2; j >= 2; j -= 2) {
81             a[j + 1] = a[j] - a[j - 1];
82             a[j] += a[j - 1];
83         }
84         a[1] = a[0] - xr;
85         a[0] += xr;
86         if (n > 4) {
87             rftbsub2(n, a, nc, w, nw);
88             bitrv2a(n, ip, a);
89             cftbsub(n, a, w);
90         } else if (n == 4) {
91             cftfsub(n, a, w);
92         }
93     }
94     dctsub2(n, a, nc, w, nw);

```

```

95     if (isgn >= 0) {
96         if (n > 4) {
97             bitrv2a(n, ip, a);
98             cftfsub(n, a, w);
99             rftfsub2(n, a, nc, w, nw);
100        } else if (n == 4) {
101            cftfsub(n, a, w);
102        }
103        xr = a[0] - a[1];
104        a[0] += a[1];
105        for (j = 2; j < n; j += 2) {
106            a[j - 1] = a[j] - a[j + 1];
107            a[j] += a[j + 1];
108        }
109        a[n - 1] = xr;
110    }
111 }
112
113 private static void ddst(int n, int isgn, double [] a, int [] ip, double [] w)
114 {
115     int j, nw, nc;
116     double xr;
117
118     nw = ip[0];
119     if (n > (nw << 2)) {
120         nw = n >> 2;
121         makewt(nw, ip, w);
122     }
123     nc = ip[1];
124     if (n > nc) {
125         nc = n;
126         makect2(nc, ip, w, nw);
127     }
128     if (isgn < 0) {
129         xr = a[n - 1];
130         for (j = n - 2; j >= 2; j -= 2) {
131             a[j + 1] = -a[j] - a[j - 1];
132             a[j] -= a[j - 1];
133         }
134         a[1] = a[0] + xr;
135         a[0] -= xr;
136         if (n > 4) {
137             rftbsub2(n, a, nc, w, nw);
138             bitrv2a(n, ip, a);
139             cftbsub(n, a, w);
140         } else if (n == 4) {
141             cftfsub(n, a, w);
142         }
143     }
144     dstsub2(n, a, nc, w, nw);
145     if (isgn >= 0) {
146         if (n > 4) {
147             bitrv2a(n, ip, a);
148             cftfsub(n, a, w);
149             rftfsub2(n, a, nc, w, nw);
150         } else if (n == 4) {
151             cftfsub(n, a, w);
152         }

```

```

153         xr = a[0] - a[1];
154         a[0] += a[1];
155         for (j = 2; j < n; j += 2) {
156             a[j - 1] = -a[j] - a[j + 1];
157             a[j] -= a[j + 1];
158         }
159         a[n - 1] = -xr;
160     }
161 }
162
163 private static void dfct(int n, double [] a, double [] t, int [] ip, double [] w)
164 {
165     int j, k, l, m, mh, nw, nc;
166     double xr, xi, yr, yi;
167
168     nw = ip[0];
169     if (n > (nw << 3)) {
170         nw = n >> 3;
171         makewt(nw, ip, w);
172     }
173     nc = ip[1];
174     if (n > (nc << 1)) {
175         nc = n >> 1;
176         makect2(nc, ip, w, nw);
177     }
178     m = n >> 1;
179     yi = a[m];
180     xi = a[0] + a[n];
181     a[0] -= a[n];
182     t[0] = xi - yi;
183     t[m] = xi + yi;
184     if (n > 2) {
185         mh = m >> 1;
186         for (j = 1; j < mh; j++) {
187             k = m - j;
188             xr = a[j] - a[n - j];
189             xi = a[j] + a[n - j];
190             yr = a[k] - a[n - k];
191             yi = a[k] + a[n - k];
192             a[j] = xr;
193             a[k] = yr;
194             t[j] = xi - yi;
195             t[k] = xi + yi;
196         }
197         t[mh] = a[mh] + a[n - mh];
198         a[mh] -= a[n - mh];
199         dctsub2(m, a, nc, w, nw);
200         if (m > 4) {
201             bitrv2a(m, ip, a);
202             cftfsub(m, a, w);
203             rftfsub2(m, a, nc, w, nw);
204         } else if (m == 4) {
205             cftfsub(m, a, w);
206         }
207         a[n - 1] = a[0] - a[1];
208         a[1] = a[0] + a[1];
209         for (j = m - 2; j >= 2; j -= 2) {
210             a[2 * j + 1] = a[j] + a[j + 1];

```

```

211         a[2 * j - 1] = a[j] - a[j + 1];
212     }
213     l = 2;
214     m = mh;
215     while (m >= 2) {
216         dctsub2(m, t, nc, w, nw);
217         if (m > 4) {
218             bitrv2a(m, ip, t);
219             cftfsub(m, t, w);
220             rftfsub2(m, t, nc, w, nw);
221         } else if (m == 4) {
222             cftfsub(m, t, w);
223         }
224         a[n - 1] = t[0] - t[1];
225         a[1] = t[0] + t[1];
226         k = 0;
227         for (j = 2; j < m; j += 2) {
228             k += 1 << 2;
229             a[k - 1] = t[j] - t[j + 1];
230             a[k + 1] = t[j] + t[j + 1];
231         }
232         l <<= 1;
233         mh = m >> 1;
234         for (j = 0; j < mh; j++) {
235             k = m - j;
236             t[j] = t[m + k] - t[m + j];
237             t[k] = t[m + k] + t[m + j];
238         }
239         t[mh] = t[m + mh];
240         m = mh;
241     }
242     a[1] = t[0];
243     a[n] = t[2] - t[1];
244     a[0] = t[2] + t[1];
245 } else {
246     a[1] = a[0];
247     a[2] = t[0];
248     a[0] = t[1];
249 }
250 }
251
252 private static void dfst(int n, double [] a, double [] t, int [] ip, double [] w)
253 {
254     int j, k, l, m, mh, nw, nc;
255     double xr, xi, yr, yi;
256
257     nw = ip[0];
258     if (n > (nw << 3)) {
259         nw = n >> 3;
260         makewt(nw, ip, w);
261     }
262     nc = ip[1];
263     if (n > (nc << 1)) {
264         nc = n >> 1;
265         makect2(nc, ip, w, nw);
266     }
267     if (n > 2) {
268         m = n >> 1;

```

```

269     mh = m >> 1;
270     for (j = 1; j < mh; j++) {
271         k = m - j;
272         xr = a[j] + a[n - j];
273         xi = a[j] - a[n - j];
274         yr = a[k] + a[n - k];
275         yi = a[k] - a[n - k];
276         a[j] = xr;
277         a[k] = yr;
278         t[j] = xi + yi;
279         t[k] = xi - yi;
280     }
281     t[0] = a[mh] - a[n - mh];
282     a[mh] += a[n - mh];
283     a[0] = a[m];
284     dstsub2(m, a, nc, w, nw);
285     if (m > 4) {
286         bitrv2a(m, ip, a);
287         cftfsub(m, a, w);
288         rftfsub2(m, a, nc, w, nw);
289     } else if (m == 4) {
290         cftfsub(m, a, w);
291     }
292     a[n - 1] = a[1] - a[0];
293     a[1] = a[0] + a[1];
294     for (j = m - 2; j >= 2; j -= 2) {
295         a[2 * j + 1] = a[j] - a[j + 1];
296         a[2 * j - 1] = -a[j] - a[j + 1];
297     }
298     l = 2;
299     m = mh;
300     while (m >= 2) {
301         dstsub2(m, t, nc, w, nw);
302         if (m > 4) {
303             bitrv2a(m, ip, t);
304             cftfsub(m, t, w);
305             rftfsub2(m, t, nc, w, nw);
306         } else if (m == 4) {
307             cftfsub(m, t, w);
308         }
309         a[n - 1] = t[1] - t[0];
310         a[1] = t[0] + t[1];
311         k = 0;
312         for (j = 2; j < m; j += 2) {
313             k += l << 2;
314             a[k - 1] = -t[j] - t[j + 1];
315             a[k + 1] = t[j] - t[j + 1];
316         }
317         l <<= 1;
318         mh = m >> 1;
319         for (j = 1; j < mh; j++) {
320             k = m - j;
321             t[j] = t[m + k] + t[m + j];
322             t[k] = t[m + k] - t[m + j];
323         }
324         t[0] = t[m + mh];
325         m = mh;
326     }

```

```

327         a[1] = t[0];
328     }
329     a[0] = 0;
330 }
331
332 private static void makewt(int nw, int [] ip, double [] w)
333 {
334     int j, nwh;
335     double delta, x, y;
336
337     ip[0] = nw;
338     ip[1] = 1;
339     if (nw > 2) {
340         nwh = nw >> 1;
341         delta = Math.atan(1.0) / nwh;
342         w[0] = 1;
343         w[1] = 0;
344         w[nwh] = Math.cos(delta * nwh);
345         w[nwh + 1] = w[nwh];
346         if (nwh > 2) {
347             for (j = 2; j < nwh; j += 2) {
348                 x = Math.cos(delta * j);
349                 y = Math.sin(delta * j);
350                 w[j] = x;
351                 w[j + 1] = y;
352                 w[nw - j] = y;
353                 w[nw - j + 1] = x;
354             }
355             bitrv2a(nw, ip, w);
356         }
357     }
358 }
359
360 private static void makect(int nc, int [] ip, double [] c)
361 {
362     int j, nch;
363     double delta;
364
365     ip[1] = nc;
366     if (nc > 1) {
367         nch = nc >> 1;
368         delta = Math.atan(1.0) / nch;
369         c[0] = Math.cos(delta * nch);
370         c[nch] = 0.5 * c[0];
371         for (j = 1; j < nch; j++) {
372             c[j] = 0.5 * Math.cos(delta * j);
373             c[nc - j] = 0.5 * Math.sin(delta * j);
374         }
375     }
376 }
377
378 private static void bitrv2(int n, int [] ip, double [] a)
379 {
380     int j, j1, k, k1, l, m, m2;
381     double xr, xi, yr, yi;
382
383     ip[0] = 0;
384     l = n;

```

```

385     m = 1;
386     while ((m << 3) < 1) {
387         l >>= 1;
388         for (j = 0; j < m; j++) {
389             ip[m + j] = ip[j] + 1;
390         }
391         m <<= 1;
392     }
393     m2 = 2 * m;
394     if ((m << 3) == 1) {
395         for (k = 0; k < m; k++) {
396             for (j = 0; j < k; j++) {
397                 j1 = 2 * j + ip[k];
398                 k1 = 2 * k + ip[j];
399                 xr = a[j1];
400                 xi = a[j1 + 1];
401                 yr = a[k1];
402                 yi = a[k1 + 1];
403                 a[j1] = yr;
404                 a[j1 + 1] = yi;
405                 a[k1] = xr;
406                 a[k1 + 1] = xi;
407                 j1 += m2;
408                 k1 += 2 * m2;
409                 xr = a[j1];
410                 xi = a[j1 + 1];
411                 yr = a[k1];
412                 yi = a[k1 + 1];
413                 a[j1] = yr;
414                 a[j1 + 1] = yi;
415                 a[k1] = xr;
416                 a[k1 + 1] = xi;
417                 j1 += m2;
418                 k1 -= m2;
419                 xr = a[j1];
420                 xi = a[j1 + 1];
421                 yr = a[k1];
422                 yi = a[k1 + 1];
423                 a[j1] = yr;
424                 a[j1 + 1] = yi;
425                 a[k1] = xr;
426                 a[k1 + 1] = xi;
427                 j1 += m2;
428                 k1 += 2 * m2;
429                 xr = a[j1];
430                 xi = a[j1 + 1];
431                 yr = a[k1];
432                 yi = a[k1 + 1];
433                 a[j1] = yr;
434                 a[j1 + 1] = yi;
435                 a[k1] = xr;
436                 a[k1 + 1] = xi;
437             }
438             j1 = 2 * k + m2 + ip[k];
439             k1 = j1 + m2;
440             xr = a[j1];
441             xi = a[j1 + 1];
442             yr = a[k1];

```



```

443         yi = a[k1 + 1];
444         a[j1] = yr;
445         a[j1 + 1] = yi;
446         a[k1] = xr;
447         a[k1 + 1] = xi;
448     }
449 } else {
450     for (k = 1; k < m; k++) {
451         for (j = 0; j < k; j++) {
452             j1 = 2 * j + ip[k];
453             k1 = 2 * k + ip[j];
454             xr = a[j1];
455             xi = a[j1 + 1];
456             yr = a[k1];
457             yi = a[k1 + 1];
458             a[j1] = yr;
459             a[j1 + 1] = yi;
460             a[k1] = xr;
461             a[k1 + 1] = xi;
462             j1 += m2;
463             k1 += m2;
464             xr = a[j1];
465             xi = a[j1 + 1];
466             yr = a[k1];
467             yi = a[k1 + 1];
468             a[j1] = yr;
469             a[j1 + 1] = yi;
470             a[k1] = xr;
471             a[k1 + 1] = xi;
472         }
473     }
474 }
475 }
476
477 private static void bitrv2conj(int n, int [] ip, double [] a)
478 {
479     int j, j1, k, k1, l, m, m2;
480     double xr, xi, yr, yi;
481
482     ip[0] = 0;
483     l = n;
484     m = 1;
485     while ((m << 3) < l) {
486         l >>= 1;
487         for (j = 0; j < m; j++) {
488             ip[m + j] = ip[j] + l;
489         }
490         m <<= 1;
491     }
492     m2 = 2 * m;
493     if ((m << 3) == 1) {
494         for (k = 0; k < m; k++) {
495             for (j = 0; j < k; j++) {
496                 j1 = 2 * j + ip[k];
497                 k1 = 2 * k + ip[j];
498                 xr = a[j1];
499                 xi = -a[j1 + 1];
500                 yr = a[k1];

```

```

501         yi = -a[k1 + 1];
502         a[j1] = yr;
503         a[j1 + 1] = yi;
504         a[k1] = xr;
505         a[k1 + 1] = xi;
506         j1 += m2;
507         k1 += 2 * m2;
508         xr = a[j1];
509         xi = -a[j1 + 1];
510         yr = a[k1];
511         yi = -a[k1 + 1];
512         a[j1] = yr;
513         a[j1 + 1] = yi;
514         a[k1] = xr;
515         a[k1 + 1] = xi;
516         j1 += m2;
517         k1 -= m2;
518         xr = a[j1];
519         xi = -a[j1 + 1];
520         yr = a[k1];
521         yi = -a[k1 + 1];
522         a[j1] = yr;
523         a[j1 + 1] = yi;
524         a[k1] = xr;
525         a[k1 + 1] = xi;
526         j1 += m2;
527         k1 += 2 * m2;
528         xr = a[j1];
529         xi = -a[j1 + 1];
530         yr = a[k1];
531         yi = -a[k1 + 1];
532         a[j1] = yr;
533         a[j1 + 1] = yi;
534         a[k1] = xr;
535         a[k1 + 1] = xi;
536     }
537     k1 = 2 * k + ip[k];
538     a[k1 + 1] = -a[k1 + 1];
539     j1 = k1 + m2;
540     k1 = j1 + m2;
541     xr = a[j1];
542     xi = -a[j1 + 1];
543     yr = a[k1];
544     yi = -a[k1 + 1];
545     a[j1] = yr;
546     a[j1 + 1] = yi;
547     a[k1] = xr;
548     a[k1 + 1] = xi;
549     k1 += m2;
550     a[k1 + 1] = -a[k1 + 1];
551 }
552 } else {
553     a[1] = -a[1];
554     a[m2 + 1] = -a[m2 + 1];
555     for (k = 1; k < m; k++) {
556         for (j = 0; j < k; j++) {
557             j1 = 2 * j + ip[k];
558             k1 = 2 * k + ip[j];

```

```

559         xr = a[j1];
560         xi = -a[j1 + 1];
561         yr = a[k1];
562         yi = -a[k1 + 1];
563         a[j1] = yr;
564         a[j1 + 1] = yi;
565         a[k1] = xr;
566         a[k1 + 1] = xi;
567         j1 += m2;
568         k1 += m2;
569         xr = a[j1];
570         xi = -a[j1 + 1];
571         yr = a[k1];
572         yi = -a[k1 + 1];
573         a[j1] = yr;
574         a[j1 + 1] = yi;
575         a[k1] = xr;
576         a[k1 + 1] = xi;
577     }
578     k1 = 2 * k + ip[k];
579     a[k1 + 1] = -a[k1 + 1];
580     a[k1 + m2 + 1] = -a[k1 + m2 + 1];
581 }
582 }
583 }
584
585 private static void cftfsub(int n, double [] a, double [] w)
586 {
587     int j, j1, j2, j3, l;
588     double x0r, x0i, x1r, x1i, x2r, x2i, x3r, x3i;
589
590     l = 2;
591     if (n > 8) {
592         cft1st(n, a, w);
593         l = 8;
594         while ((l << 2) < n) {
595             cftmdl(n, l, a, w);
596             l <<= 2;
597         }
598     }
599     if ((l << 2) == n) {
600         for (j = 0; j < l; j += 2) {
601             j1 = j + 1;
602             j2 = j1 + 1;
603             j3 = j2 + 1;
604             x0r = a[j] + a[j1];
605             x0i = a[j + 1] + a[j1 + 1];
606             x1r = a[j] - a[j1];
607             x1i = a[j + 1] - a[j1 + 1];
608             x2r = a[j2] + a[j3];
609             x2i = a[j2 + 1] + a[j3 + 1];
610             x3r = a[j2] - a[j3];
611             x3i = a[j2 + 1] - a[j3 + 1];
612             a[j] = x0r + x2r;
613             a[j + 1] = x0i + x2i;
614             a[j2] = x0r - x2r;
615             a[j2 + 1] = x0i - x2i;
616             a[j1] = x1r - x3i;

```

```

617         a[j1 + 1] = x1i + x3r;
618         a[j3] = x1r + x3i;
619         a[j3 + 1] = x1i - x3r;
620     }
621 } else {
622     for (j = 0; j < l; j += 2) {
623         j1 = j + 1;
624         x0r = a[j] - a[j1];
625         x0i = a[j + 1] - a[j1 + 1];
626         a[j] += a[j1];
627         a[j + 1] += a[j1 + 1];
628         a[j1] = x0r;
629         a[j1 + 1] = x0i;
630     }
631 }
632 }
633
634 private static void cftbsub(int n, double [] a, double [] w)
635 {
636     int j, j1, j2, j3, l;
637     double x0r, x0i, x1r, x1i, x2r, x2i, x3r, x3i;
638
639     l = 2;
640     if (n > 8) {
641         cft1st(n, a, w);
642         l = 8;
643         while ((l << 2) < n) {
644             cftmdl(n, l, a, w);
645             l <<= 2;
646         }
647     }
648     if ((l << 2) == n) {
649         for (j = 0; j < l; j += 2) {
650             j1 = j + 1;
651             j2 = j1 + 1;
652             j3 = j2 + 1;
653             x0r = a[j] + a[j1];
654             x0i = -a[j + 1] - a[j1 + 1];
655             x1r = a[j] - a[j1];
656             x1i = -a[j + 1] + a[j1 + 1];
657             x2r = a[j2] + a[j3];
658             x2i = a[j2 + 1] + a[j3 + 1];
659             x3r = a[j2] - a[j3];
660             x3i = a[j2 + 1] - a[j3 + 1];
661             a[j] = x0r + x2r;
662             a[j + 1] = x0i - x2i;
663             a[j2] = x0r - x2r;
664             a[j2 + 1] = x0i + x2i;
665             a[j1] = x1r - x3i;
666             a[j1 + 1] = x1i - x3r;
667             a[j3] = x1r + x3i;
668             a[j3 + 1] = x1i + x3r;
669         }
670     } else {
671         for (j = 0; j < l; j += 2) {
672             j1 = j + 1;
673             x0r = a[j] - a[j1];
674             x0i = -a[j + 1] + a[j1 + 1];

```

```

675         a[j] += a[j1];
676         a[j + 1] = -a[j + 1] - a[j1 + 1];
677         a[j1] = x0r;
678         a[j1 + 1] = x0i;
679     }
680 }
681 }
682
683 private static void cft1st(int n, double [] a, double [] w)
684 {
685     int j, k1, k2;
686     double wk1r, wk1i, wk2r, wk2i, wk3r, wk3i;
687     double x0r, x0i, x1r, x1i, x2r, x2i, x3r, x3i;
688
689     x0r = a[0] + a[2];
690     x0i = a[1] + a[3];
691     x1r = a[0] - a[2];
692     x1i = a[1] - a[3];
693     x2r = a[4] + a[6];
694     x2i = a[5] + a[7];
695     x3r = a[4] - a[6];
696     x3i = a[5] - a[7];
697     a[0] = x0r + x2r;
698     a[1] = x0i + x2i;
699     a[4] = x0r - x2r;
700     a[5] = x0i - x2i;
701     a[2] = x1r - x3i;
702     a[3] = x1i + x3r;
703     a[6] = x1r + x3i;
704     a[7] = x1i - x3r;
705     wk1r = w[2];
706     x0r = a[8] + a[10];
707     x0i = a[9] + a[11];
708     x1r = a[8] - a[10];
709     x1i = a[9] - a[11];
710     x2r = a[12] + a[14];
711     x2i = a[13] + a[15];
712     x3r = a[12] - a[14];
713     x3i = a[13] - a[15];
714     a[8] = x0r + x2r;
715     a[9] = x0i + x2i;
716     a[12] = x2i - x0i;
717     a[13] = x0r - x2r;
718     x0r = x1r - x3i;
719     x0i = x1i + x3r;
720     a[10] = wk1r * (x0r - x0i);
721     a[11] = wk1r * (x0r + x0i);
722     x0r = x3i + x1r;
723     x0i = x3r - x1i;
724     a[14] = wk1r * (x0i - x0r);
725     a[15] = wk1r * (x0i + x0r);
726     k1 = 0;
727     for (j = 16; j < n; j += 16) {
728         k1 += 2;
729         k2 = 2 * k1;
730         wk2r = w[k1];
731         wk2i = w[k1 + 1];
732         wk1r = w[k2];

```

```

733     wk1i = w[k2 + 1];
734     wk3r = wk1r - 2 * wk2i * wk1i;
735     wk3i = 2 * wk2i * wk1r - wk1i;
736     x0r = a[j] + a[j + 2];
737     x0i = a[j + 1] + a[j + 3];
738     x1r = a[j] - a[j + 2];
739     x1i = a[j + 1] - a[j + 3];
740     x2r = a[j + 4] + a[j + 6];
741     x2i = a[j + 5] + a[j + 7];
742     x3r = a[j + 4] - a[j + 6];
743     x3i = a[j + 5] - a[j + 7];
744     a[j] = x0r + x2r;
745     a[j + 1] = x0i + x2i;
746     x0r -= x2r;
747     x0i -= x2i;
748     a[j + 4] = wk2r * x0r - wk2i * x0i;
749     a[j + 5] = wk2r * x0i + wk2i * x0r;
750     x0r = x1r - x3i;
751     x0i = x1i + x3r;
752     a[j + 2] = wk1r * x0r - wk1i * x0i;
753     a[j + 3] = wk1r * x0i + wk1i * x0r;
754     x0r = x1r + x3i;
755     x0i = x1i - x3r;
756     a[j + 6] = wk3r * x0r - wk3i * x0i;
757     a[j + 7] = wk3r * x0i + wk3i * x0r;
758     wk1r = w[k2 + 2];
759     wk1i = w[k2 + 3];
760     wk3r = wk1r - 2 * wk2r * wk1i;
761     wk3i = 2 * wk2r * wk1r - wk1i;
762     x0r = a[j + 8] + a[j + 10];
763     x0i = a[j + 9] + a[j + 11];
764     x1r = a[j + 8] - a[j + 10];
765     x1i = a[j + 9] - a[j + 11];
766     x2r = a[j + 12] + a[j + 14];
767     x2i = a[j + 13] + a[j + 15];
768     x3r = a[j + 12] - a[j + 14];
769     x3i = a[j + 13] - a[j + 15];
770     a[j + 8] = x0r + x2r;
771     a[j + 9] = x0i + x2i;
772     x0r -= x2r;
773     x0i -= x2i;
774     a[j + 12] = -wk2i * x0r - wk2r * x0i;
775     a[j + 13] = -wk2i * x0i + wk2r * x0r;
776     x0r = x1r - x3i;
777     x0i = x1i + x3r;
778     a[j + 10] = wk1r * x0r - wk1i * x0i;
779     a[j + 11] = wk1r * x0i + wk1i * x0r;
780     x0r = x1r + x3i;
781     x0i = x1i - x3r;
782     a[j + 14] = wk3r * x0r - wk3i * x0i;
783     a[j + 15] = wk3r * x0i + wk3i * x0r;
784 }
785 }
786
787 private static void cftmdl(int n, int l, double [] a, double [] w)
788 {
789     int j, j1, j2, j3, k, k1, k2, m, m2;
790     double wk1r, wk1i, wk2r, wk2i, wk3r, wk3i;

```

```

791     double x0r, x0i, x1r, x1i, x2r, x2i, x3r, x3i;
792
793     m = 1 << 2;
794     for (j = 0; j < l; j += 2) {
795         j1 = j + 1;
796         j2 = j1 + 1;
797         j3 = j2 + 1;
798         x0r = a[j] + a[j1];
799         x0i = a[j + 1] + a[j1 + 1];
800         x1r = a[j] - a[j1];
801         x1i = a[j + 1] - a[j1 + 1];
802         x2r = a[j2] + a[j3];
803         x2i = a[j2 + 1] + a[j3 + 1];
804         x3r = a[j2] - a[j3];
805         x3i = a[j2 + 1] - a[j3 + 1];
806         a[j] = x0r + x2r;
807         a[j + 1] = x0i + x2i;
808         a[j2] = x0r - x2r;
809         a[j2 + 1] = x0i - x2i;
810         a[j1] = x1r - x3i;
811         a[j1 + 1] = x1i + x3r;
812         a[j3] = x1r + x3i;
813         a[j3 + 1] = x1i - x3r;
814     }
815     wk1r = w[2];
816     for (j = m; j < l + m; j += 2) {
817         j1 = j + 1;
818         j2 = j1 + 1;
819         j3 = j2 + 1;
820         x0r = a[j] + a[j1];
821         x0i = a[j + 1] + a[j1 + 1];
822         x1r = a[j] - a[j1];
823         x1i = a[j + 1] - a[j1 + 1];
824         x2r = a[j2] + a[j3];
825         x2i = a[j2 + 1] + a[j3 + 1];
826         x3r = a[j2] - a[j3];
827         x3i = a[j2 + 1] - a[j3 + 1];
828         a[j] = x0r + x2r;
829         a[j + 1] = x0i + x2i;
830         a[j2] = x2i - x0i;
831         a[j2 + 1] = x0r - x2r;
832         x0r = x1r - x3i;
833         x0i = x1i + x3r;
834         a[j1] = wk1r * (x0r - x0i);
835         a[j1 + 1] = wk1r * (x0r + x0i);
836         x0r = x3i + x1r;
837         x0i = x3r - x1i;
838         a[j3] = wk1r * (x0i - x0r);
839         a[j3 + 1] = wk1r * (x0i + x0r);
840     }
841     k1 = 0;
842     m2 = 2 * m;
843     for (k = m2; k < n; k += m2) {
844         k1 += 2;
845         k2 = 2 * k1;
846         wk2r = w[k1];
847         wk2i = w[k1 + 1];
848         wk1r = w[k2];

```

```

849     wk1i = w[k2 + 1];
850     wk3r = wk1r - 2 * wk2i * wk1i;
851     wk3i = 2 * wk2i * wk1r - wk1i;
852     for (j = k; j < l + k; j += 2) {
853         j1 = j + 1;
854         j2 = j1 + 1;
855         j3 = j2 + 1;
856         x0r = a[j] + a[j1];
857         x0i = a[j + 1] + a[j1 + 1];
858         x1r = a[j] - a[j1];
859         x1i = a[j + 1] - a[j1 + 1];
860         x2r = a[j2] + a[j3];
861         x2i = a[j2 + 1] + a[j3 + 1];
862         x3r = a[j2] - a[j3];
863         x3i = a[j2 + 1] - a[j3 + 1];
864         a[j] = x0r + x2r;
865         a[j + 1] = x0i + x2i;
866         x0r -= x2r;
867         x0i -= x2i;
868         a[j2] = wk2r * x0r - wk2i * x0i;
869         a[j2 + 1] = wk2r * x0i + wk2i * x0r;
870         x0r = x1r - x3i;
871         x0i = x1i + x3r;
872         a[j1] = wk1r * x0r - wk1i * x0i;
873         a[j1 + 1] = wk1r * x0i + wk1i * x0r;
874         x0r = x1r + x3i;
875         x0i = x1i - x3r;
876         a[j3] = wk3r * x0r - wk3i * x0i;
877         a[j3 + 1] = wk3r * x0i + wk3i * x0r;
878     }
879     wk1r = w[k2 + 2];
880     wk1i = w[k2 + 3];
881     wk3r = wk1r - 2 * wk2r * wk1i;
882     wk3i = 2 * wk2r * wk1r - wk1i;
883     for (j = k + m; j < l + (k + m); j += 2) {
884         j1 = j + 1;
885         j2 = j1 + 1;
886         j3 = j2 + 1;
887         x0r = a[j] + a[j1];
888         x0i = a[j + 1] + a[j1 + 1];
889         x1r = a[j] - a[j1];
890         x1i = a[j + 1] - a[j1 + 1];
891         x2r = a[j2] + a[j3];
892         x2i = a[j2 + 1] + a[j3 + 1];
893         x3r = a[j2] - a[j3];
894         x3i = a[j2 + 1] - a[j3 + 1];
895         a[j] = x0r + x2r;
896         a[j + 1] = x0i + x2i;
897         x0r -= x2r;
898         x0i -= x2i;
899         a[j2] = -wk2i * x0r - wk2r * x0i;
900         a[j2 + 1] = -wk2i * x0i + wk2r * x0r;
901         x0r = x1r - x3i;
902         x0i = x1i + x3r;
903         a[j1] = wk1r * x0r - wk1i * x0i;
904         a[j1 + 1] = wk1r * x0i + wk1i * x0r;
905         x0r = x1r + x3i;
906         x0i = x1i - x3r;

```



```

907         a[j3] = wk3r * x0r - wk3i * x0i;
908         a[j3 + 1] = wk3r * x0i + wk3i * x0r;
909     }
910 }
911 }
912
913 private static void rftfsub(int n, double [] a, int nc, double [] c)
914 {
915     int j, k, kk, ks, m;
916     double wkr, wki, xr, xi, yr, yi;
917
918     m = n >> 1;
919     ks = 2 * nc / m;
920     kk = 0;
921     for (j = 2; j < m; j += 2) {
922         k = n - j;
923         kk += ks;
924         wkr = 0.5 - c[nc - kk];
925         wki = c[kk];
926         xr = a[j] - a[k];
927         xi = a[j + 1] + a[k + 1];
928         yr = wkr * xr - wki * xi;
929         yi = wkr * xi + wki * xr;
930         a[j] -= yr;
931         a[j + 1] -= yi;
932         a[k] += yr;
933         a[k + 1] -= yi;
934     }
935 }
936
937 private static void rftbsub(int n, double [] a, int nc, double [] c)
938 {
939     int j, k, kk, ks, m;
940     double wkr, wki, xr, xi, yr, yi;
941
942     a[1] = -a[1];
943     m = n >> 1;
944     ks = 2 * nc / m;
945     kk = 0;
946     for (j = 2; j < m; j += 2) {
947         k = n - j;
948         kk += ks;
949         wkr = 0.5 - c[nc - kk];
950         wki = c[kk];
951         xr = a[j] - a[k];
952         xi = a[j + 1] + a[k + 1];
953         yr = wkr * xr + wki * xi;
954         yi = wkr * xi - wki * xr;
955         a[j] -= yr;
956         a[j + 1] = yi - a[j + 1];
957         a[k] += yr;
958         a[k + 1] = yi - a[k + 1];
959     }
960     a[m + 1] = -a[m + 1];
961 }
962
963 private static void dctsub(int n, double [] a, int nc, double [] c)
964 {

```

```

965     int j, k, kk, ks, m;
966     double wkr, wki, xr;
967
968     m = n >> 1;
969     ks = nc / n;
970     kk = 0;
971     for (j = 1; j < m; j++) {
972         k = n - j;
973         kk += ks;
974         wkr = c[kk] - c[nc - kk];
975         wki = c[kk] + c[nc - kk];
976         xr = wki * a[j] - wkr * a[k];
977         a[j] = wkr * a[j] + wki * a[k];
978         a[k] = xr;
979     }
980     a[m] *= c[0];
981 }
982
983 private static void dstsub(int n, double [] a, int nc, double [] c)
984 {
985     int j, k, kk, ks, m;
986     double wkr, wki, xr;
987
988     m = n >> 1;
989     ks = nc / n;
990     kk = 0;
991     for (j = 1; j < m; j++) {
992         k = n - j;
993         kk += ks;
994         wkr = c[kk] - c[nc - kk];
995         wki = c[kk] + c[nc - kk];
996         xr = wki * a[k] - wkr * a[j];
997         a[k] = wkr * a[k] + wki * a[j];
998         a[j] = xr;
999     }
1000     a[m] *= c[0];
1001 }
1002
1003 /* 追加 */
1004 private static void bitrv2a(int n, int [] ip, double [] a)
1005 {
1006     int j, j1, k, k1, l, m, m2;
1007     double xr, xi, yr, yi;
1008
1009     ip[2] = 0;
1010     l = n;
1011     m = 1;
1012     while ((m << 3) < l) {
1013         l >>= 1;
1014         for (j = 0; j < m; j++) {
1015             ip[m + j + 2] = ip[j + 2] + 1;
1016         }
1017         m <<= 1;
1018     }
1019     m2 = 2 * m;
1020     if ((m << 3) == 1) {
1021         for (k = 0; k < m; k++) {
1022             for (j = 0; j < k; j++) {

```

```

1023         j1 = 2 * j + ip[k + 2];
1024         k1 = 2 * k + ip[j + 2];
1025         xr = a[j1];
1026         xi = a[j1 + 1];
1027         yr = a[k1];
1028         yi = a[k1 + 1];
1029         a[j1] = yr;
1030         a[j1 + 1] = yi;
1031         a[k1] = xr;
1032         a[k1 + 1] = xi;
1033         j1 += m2;
1034         k1 += 2 * m2;
1035         xr = a[j1];
1036         xi = a[j1 + 1];
1037         yr = a[k1];
1038         yi = a[k1 + 1];
1039         a[j1] = yr;
1040         a[j1 + 1] = yi;
1041         a[k1] = xr;
1042         a[k1 + 1] = xi;
1043         j1 += m2;
1044         k1 -= m2;
1045         xr = a[j1];
1046         xi = a[j1 + 1];
1047         yr = a[k1];
1048         yi = a[k1 + 1];
1049         a[j1] = yr;
1050         a[j1 + 1] = yi;
1051         a[k1] = xr;
1052         a[k1 + 1] = xi;
1053         j1 += m2;
1054         k1 += 2 * m2;
1055         xr = a[j1];
1056         xi = a[j1 + 1];
1057         yr = a[k1];
1058         yi = a[k1 + 1];
1059         a[j1] = yr;
1060         a[j1 + 1] = yi;
1061         a[k1] = xr;
1062         a[k1 + 1] = xi;
1063     }
1064     j1 = 2 * k + m2 + ip[k + 2];
1065     k1 = j1 + m2;
1066     xr = a[j1];
1067     xi = a[j1 + 1];
1068     yr = a[k1];
1069     yi = a[k1 + 1];
1070     a[j1] = yr;
1071     a[j1 + 1] = yi;
1072     a[k1] = xr;
1073     a[k1 + 1] = xi;
1074 }
1075 } else {
1076     for (k = 1; k < m; k++) {
1077         for (j = 0; j < k; j++) {
1078             j1 = 2 * j + ip[k + 2];
1079             k1 = 2 * k + ip[j + 2];
1080             xr = a[j1];

```

```

1081         xi = a[j1 + 1];
1082         yr = a[k1];
1083         yi = a[k1 + 1];
1084         a[j1] = yr;
1085         a[j1 + 1] = yi;
1086         a[k1] = xr;
1087         a[k1 + 1] = xi;
1088         j1 += m2;
1089         k1 += m2;
1090         xr = a[j1];
1091         xi = a[j1 + 1];
1092         yr = a[k1];
1093         yi = a[k1 + 1];
1094         a[j1] = yr;
1095         a[j1 + 1] = yi;
1096         a[k1] = xr;
1097         a[k1 + 1] = xi;
1098     }
1099 }
1100 }
1101 }
1102
1103 private static void bitrv2conj2(int n, int [] ip, double [] a)
1104 {
1105     int j, j1, k, k1, l, m, m2;
1106     double xr, xi, yr, yi;
1107
1108     ip[2] = 0;
1109     l = n;
1110     m = 1;
1111     while ((m << 3) < l) {
1112         l >>= 1;
1113         for (j = 0; j < m; j++) {
1114             ip[m + j + 2] = ip[j + 2] + 1;
1115         }
1116         m <<= 1;
1117     }
1118     m2 = 2 * m;
1119     if ((m << 3) == 1) {
1120         for (k = 0; k < m; k++) {
1121             for (j = 0; j < k; j++) {
1122                 j1 = 2 * j + ip[k + 2];
1123                 k1 = 2 * k + ip[j + 2];
1124                 xr = a[j1];
1125                 xi = -a[j1 + 1];
1126                 yr = a[k1];
1127                 yi = -a[k1 + 1];
1128                 a[j1] = yr;
1129                 a[j1 + 1] = yi;
1130                 a[k1] = xr;
1131                 a[k1 + 1] = xi;
1132                 j1 += m2;
1133                 k1 += 2 * m2;
1134                 xr = a[j1];
1135                 xi = -a[j1 + 1];
1136                 yr = a[k1];
1137                 yi = -a[k1 + 1];
1138                 a[j1] = yr;

```

```

1139         a[j1 + 1] = yi;
1140         a[k1] = xr;
1141         a[k1 + 1] = xi;
1142         j1 += m2;
1143         k1 -= m2;
1144         xr = a[j1];
1145         xi = -a[j1 + 1];
1146         yr = a[k1];
1147         yi = -a[k1 + 1];
1148         a[j1] = yr;
1149         a[j1 + 1] = yi;
1150         a[k1] = xr;
1151         a[k1 + 1] = xi;
1152         j1 += m2;
1153         k1 += 2 * m2;
1154         xr = a[j1];
1155         xi = -a[j1 + 1];
1156         yr = a[k1];
1157         yi = -a[k1 + 1];
1158         a[j1] = yr;
1159         a[j1 + 1] = yi;
1160         a[k1] = xr;
1161         a[k1 + 1] = xi;
1162     }
1163     k1 = 2 * k + ip[k + 2];
1164     a[k1 + 1] = -a[k1 + 1];
1165     j1 = k1 + m2;
1166     k1 = j1 + m2;
1167     xr = a[j1];
1168     xi = -a[j1 + 1];
1169     yr = a[k1];
1170     yi = -a[k1 + 1];
1171     a[j1] = yr;
1172     a[j1 + 1] = yi;
1173     a[k1] = xr;
1174     a[k1 + 1] = xi;
1175     k1 += m2;
1176     a[k1 + 1] = -a[k1 + 1];
1177 }
1178 } else {
1179     a[1] = -a[1];
1180     a[m2 + 1] = -a[m2 + 1];
1181     for (k = 1; k < m; k++) {
1182         for (j = 0; j < k; j++) {
1183             j1 = 2 * j + ip[k + 2];
1184             k1 = 2 * k + ip[j + 2];
1185             xr = a[j1];
1186             xi = -a[j1 + 1];
1187             yr = a[k1];
1188             yi = -a[k1 + 1];
1189             a[j1] = yr;
1190             a[j1 + 1] = yi;
1191             a[k1] = xr;
1192             a[k1 + 1] = xi;
1193             j1 += m2;
1194             k1 += m2;
1195             xr = a[j1];
1196             xi = -a[j1 + 1];

```

```

1197         yr = a[k1];
1198         yi = -a[k1 + 1];
1199         a[j1] = yr;
1200         a[j1 + 1] = yi;
1201         a[k1] = xr;
1202         a[k1 + 1] = xi;
1203     }
1204     k1 = 2 * k + ip[k + 2];
1205     a[k1 + 1] = -a[k1 + 1];
1206     a[k1 + m2 + 1] = -a[k1 + m2 + 1];
1207 }
1208 }
1209 }
1210
1211 private static void makect2(int nc, int [] ip, double [] c, int nw)
1212 {
1213     int j, nch;
1214     double delta;
1215
1216     ip[1] = nc;
1217     if (nc > 1) {
1218         nch = nc >> 1;
1219         delta = Math.atan(1.0) / nch;
1220         c[nw] = Math.cos(delta * nch);
1221         c[nch + nw] = 0.5 * c[nw];
1222         for (j = 1; j < nch; j++) {
1223             c[j + nw] = 0.5 * Math.cos(delta * j);
1224             c[nc - j + nw] = 0.5 * Math.sin(delta * j);
1225         }
1226     }
1227 }
1228
1229 private static void rftfsub2(int n, double [] a, int nc, double [] c, int nw)
1230 {
1231     int j, k, kk, ks, m;
1232     double wkr, wki, xr, xi, yr, yi;
1233
1234     m = n >> 1;
1235     ks = 2 * nc / m;
1236     kk = 0;
1237     for (j = 2; j < m; j += 2) {
1238         k = n - j;
1239         kk += ks;
1240         wkr = 0.5 - c[nc - kk + nw];
1241         wki = c[kk + nw];
1242         xr = a[j] - a[k];
1243         xi = a[j + 1] + a[k + 1];
1244         yr = wkr * xr - wki * xi;
1245         yi = wkr * xi + wki * xr;
1246         a[j] -= yr;
1247         a[j + 1] -= yi;
1248         a[k] += yr;
1249         a[k + 1] -= yi;
1250     }
1251 }
1252
1253 private static void rftbsub2(int n, double [] a, int nc, double [] c, int nw)
1254 {

```

```

1255     int j, k, kk, ks, m;
1256     double wkr, wki, xr, xi, yr, yi;
1257
1258     a[1] = -a[1];
1259     m = n >> 1;
1260     ks = 2 * nc / m;
1261     kk = 0;
1262     for (j = 2; j < m; j += 2) {
1263         k = n - j;
1264         kk += ks;
1265         wkr = 0.5 - c[nc - kk + nw];
1266         wki = c[kk + nw];
1267         xr = a[j] - a[k];
1268         xi = a[j + 1] + a[k + 1];
1269         yr = wkr * xr + wki * xi;
1270         yi = wkr * xi - wki * xr;
1271         a[j] -= yr;
1272         a[j + 1] = yi - a[j + 1];
1273         a[k] += yr;
1274         a[k + 1] = yi - a[k + 1];
1275     }
1276     a[m + 1] = -a[m + 1];
1277 }
1278
1279 private static void dctsub2(int n, double [] a, int nc, double [] c, int nw)
1280 {
1281     int j, k, kk, ks, m;
1282     double wkr, wki, xr;
1283
1284     m = n >> 1;
1285     ks = nc / n;
1286     kk = 0;
1287     for (j = 1; j < m; j++) {
1288         k = n - j;
1289         kk += ks;
1290         wkr = c[kk + nw] - c[nc - kk + nw];
1291         wki = c[kk + nw] + c[nc - kk + nw];
1292         xr = wki * a[j] - wkr * a[k];
1293         a[j] = wkr * a[j] + wki * a[k];
1294         a[k] = xr;
1295     }
1296     a[m] *= c[nw];
1297 }
1298
1299 private static void dstsub2(int n, double [] a, int nc, double [] c, int nw)
1300 {
1301     int j, k, kk, ks, m;
1302     double wkr, wki, xr;
1303
1304     m = n >> 1;
1305     ks = nc / n;
1306     kk = 0;
1307     for (j = 1; j < m; j++) {
1308         k = n - j;
1309         kk += ks;
1310         wkr = c[kk + nw] - c[nc - kk + nw];
1311         wki = c[kk + nw] + c[nc - kk + nw];
1312         xr = wki * a[k] - wkr * a[j];

```

```

1313         a[k] = wkr * a[k] + wki * a[j];
1314         a[j] = xr;
1315     }
1316     a[m] *= c[nw];
1317 }
1318 /* ここまで */
1319
1320 private static final int EXTERNAL_BUFFER_SIZE = 128000;
1321 public static void main(String[] args){
1322     int frameSize;
1323     int sampleSizeInBits;
1324     int channels;
1325     float sampleRate;
1326     boolean isStereo;
1327     boolean isBigEndian;
1328     short [] a1, a2;
1329     int n, M, count, point;
1330     int [] ip;
1331     double wa, m;
1332     double [] a, b, w;
1333     a1 = new short [1280000];
1334     a2 = new short [1280000];
1335     M = 65536;
1336     count = 0;
1337     point = 39500;
1338     ip = new int [M+1];
1339     a = new double [M+1];
1340     b = new double [M+1];
1341     w = new double [M+1];
1342     if (args.length == 0) System.exit(0);
1343     try {
1344         File soundFile = new File(args[0]);
1345         AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(soundFile);
1346         AudioFormat audioFormat = audioInputStream.getFormat();
1347
1348         DataLine.Info info = new DataLine.Info(SourceDataLine.class, audioFormat);
1349         SourceDataLine line = (SourceDataLine) AudioSystem.getLine(info);
1350         line.open(audioFormat);
1351         line.start();
1352
1353         channels = audioFormat.getChannels();
1354         isStereo = (channels == 2);
1355         isBigEndian = audioFormat.isBigEndian();
1356         frameSize = audioFormat.getFrameSize();
1357         sampleSizeInBits = audioFormat.getSampleSizeInBits();
1358         sampleRate = audioFormat.getSampleRate();
1359         System.out.println("#original file: " + args[0]);
1360         System.out.println("#number of channels: " + channels);
1361         System.out.println("#sampling rate: " + sampleRate);
1362         System.out.println("#number of bits per sample: " + sampleSizeInBits);
1363         System.out.println("#FrameSize: " + frameSize);
1364         System.out.println("#isBigEndian: " + isBigEndian);
1365         if (audioFormat.getEncoding() == AudioFormat.Encoding.PCM_SIGNED) {
1366             System.out.println("#PCM Signed");
1367         }
1368         else if (audioFormat.getEncoding() == AudioFormat.Encoding.PCM_UNSIGNED) {
1369             System.out.println("#PCM Unsigned!!!");
1370             System.exit(0);

```



```

1371     }
1372     else {
1373         System.out.println("#NO PCM!!");
1374         System.exit(0);
1375     }
1376
1377     int nBytesRead = 0;
1378     byte[] abData = new byte[EXTERNAL_BUFFER_SIZE];
1379     if (isStereo) {
1380         if (sampleSizeInBits == 16) {
1381             while (nBytesRead != -1) {
1382                 nBytesRead = audioInputStream.read(abData, 0, abData.length);
1383                 if (nBytesRead >= 0) {
1384                     int nBytesWritten = line.write(abData, 0, nBytesRead);
1385                     for (int i = 0; i < nBytesRead; i += 4) {
1386                         short left, right;
1387                         ++count;
1388                         left = (short)(abData[i] & 0xff | (abData[i+1] << 8));
1389                         right = (short)(abData[i+2] & 0xff | (abData[i+3] << 8));
1390                         a1[count] = left;
1391                         a2[count] = right;
1392                     }
1393                 }
1394             }
1395         }
1396     }
1397     else {
1398         while (nBytesRead != -1) {
1399             nBytesRead = audioInputStream.read(abData, 0, abData.length);
1400             if (nBytesRead >= 0) {
1401                 int nBytesWritten = line.write(abData, 0, nBytesRead);
1402                 for (int i = 0; i < nBytesRead; i += 2) {
1403                     short left, right;
1404                     ++count;
1405                     left = (short)(abData[i] & 0xff);
1406                     right = (short)(abData[i+1] & 0xff);
1407                     a1[count] = left;
1408                     a2[count] = right;
1409                 }
1410             }
1411         }
1412     }
1413     else {
1414         if (sampleSizeInBits == 16) {
1415             while (nBytesRead != -1) {
1416                 nBytesRead = audioInputStream.read(abData, 0, abData.length);
1417                 if (nBytesRead >= 0) {
1418                     int nBytesWritten = line.write(abData, 0, nBytesRead);
1419                     for (int i = 0; i < nBytesRead; i += 2) {
1420                         short c = (short)(abData[i] & 0xff | (abData[i+1] << 8));
1421                         ++count;
1422                         a1[count] = c;
1423                     }
1424                 }
1425             }
1426         }
1427     }
1428     else {
1429         while (nBytesRead != -1) {

```

```

1429         nBytesRead = audioInputStream.read(abData, 0, abData.length);
1430         if (nBytesRead >= 0) {
1431             int nBytesWritten = line.write(abData, 0, nBytesRead);
1432             for (int i = 0; i < nBytesRead; i++) {
1433                 short c;
1434                 ++count;
1435                 c = (short)(abData[i] & 0xff);
1436                 a1[count] = c;
1437             }
1438         }
1439     }
1440 }
1441 }
1442 for(n = 0;n <= M;n++){
1443     a[n] = (double)(a1[n+point]);
1444 }
1445 rdft(M, 1, a, ip, w);
1446 for(n = 1;n <= M/2-1;n++){
1447     wa = (2.0/M) * (2.0/M) * (a[2*n] * a[2*n] + a[2*n+1] * a[2*n+1]);
1448     m = (double)(n * 44100) / (double)(M);
1449     System.out.printf("%f %f\n", m, Math.abs(wa));
1450 }
1451 line.drain();
1452 line.close();
1453
1454     System.exit(0);
1455 } catch (Exception e) {
1456     e.printStackTrace();
1457     System.exit(1);
1458 }
1459 }
1460 }

```

## A.2 ReadWave.java

```

1 //
2 // ReadWave.java
3 // 2008/1/30 初めて作成
4 // 2008/2/13 一木君にバグを指摘される。Byte は符号付きだった。
5 // 2008/2/15 やはりビット演算だけで記述することにした。
6 //
7
8 import java.io.File;
9 import javax.sound.sampled.*;
10
11 public class ReadWave {
12     private static final int EXTERNAL_BUFFER_SIZE = 128000;
13     public static void main(String[] args) {
14         int frameSize; //
15         int sampleSizeInBits; // 音声データの数値のビット数 (16または8)
16         int channels; // チャンネルの数 (2がステレオ, 1がモノラル)
17         float sampleRate; // サンプリングレート (1秒間の...)
18         boolean isStereo; // ステレオか
19         boolean isBigEndian; // ビッグエンディアンか (上位バイトが先か)
20         if (args.length == 0) System.exit(0);
21         try {

```

```

22 // File クラスのインスタンスを生成する
23 File soundFile = new File(args[0]);
24 // オーディオ入力ストリームを取得する
25 AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(soundFile);
26 // オーディオフォーマットを取得する
27 AudioFormat audioFormat = audioInputStream.getFormat();
28
29 // データラインの情報オブジェクトを生成する
30 DataLine.Info info = new DataLine.Info(SourceDataLine.class, audioFormat);
31 // 指定されたデータライン情報に一致するラインを取得する
32 SourceDataLine line = (SourceDataLine) AudioSystem.getLine(info);
33 // 指定されたオーディオ形式でラインを開きます
34 line.open(audioFormat);
35 // ラインでのデータ入出力を可能にします
36 line.start();
37
38 // データの形式を読み取る
39 channels = audioFormat.getChannels();
40 isStereo = (channels == 2);
41 isBigEndian = audioFormat.isBigEndian();
42 frameSize = audioFormat.getFrameSize();
43 sampleSizeInBits = audioFormat.getSampleSizeInBits();
44 sampleRate = audioFormat.getSampleRate();
45 System.out.println("#original file: " + args[0]);
46 System.out.println("#number of channels: " + channels);
47 System.out.println("#sampling rate: " + sampleRate);
48 System.out.println("#number of bits per sample: " + sampleSizeInBits);
49 System.out.println("#FrameSize: " + frameSize);
50 System.out.println("#isBigEndian: " + isBigEndian);
51 if (audioFormat.getEncoding() == AudioFormat.Encoding.PCM_SIGNED) {
52     System.out.println("#PCM Signed");
53 }
54 else if (audioFormat.getEncoding() == AudioFormat.Encoding.PCM_UNSIGNED) {
55     System.out.println("#PCM Unsigned!!!");
56     System.exit(0);
57 }
58 else {
59     System.out.println("#NO PCM!!!");
60     System.exit(0);
61 }
62
63 // 音声データを読み取り、鳴らして、数値を表示
64 int nBytesRead = 0;
65 byte[] abData = new byte[EXTERNAL_BUFFER_SIZE];
66 if (isStereo) {
67     if (sampleSizeInBits == 16) {
68         // 16 ビットステレオ (フツー)
69         while (nBytesRead != -1) {
70             // オーディオストリームからデータを読み込みます
71             nBytesRead = audioInputStream.read(abData, 0, abData.length);
72             if (nBytesRead >= 0) {
73                 // オーディオデータをミキサーに書き込みます
74                 int nBytesWritten = line.write(abData, 0, nBytesRead);
75                 for (int i = 0; i < nBytesRead; i += 4) {
76                     short left, right;
77                     left = (short)(abData[i] & 0xff | (abData[i+1] << 8));
78                     right = (short)(abData[i+2] & 0xff | (abData[i+3] << 8));
79                     System.out.println("" + left + " " + right);

```

```

80         }
81     }
82 }
83 }
84 else { // sampleSizeInBits == 8
85     // 8ビットステレオ (フツー)
86     while (nBytesRead != -1) {
87         // オーディオストリームからデータを読み込みます
88         nBytesRead = audioInputStream.read(abData, 0, abData.length);
89         if (nBytesRead >= 0) {
90             // オーディオデータをミキサーに書き込みます
91             int nBytesWritten = line.write(abData, 0, nBytesRead);
92             for (int i = 0; i < nBytesRead; i += 2) {
93                 short left, right;
94                 left = (short)(abData[i] & 0xff);
95                 right = (short)(abData[i+1] & 0xff);
96                 System.out.println("" + left + " " + right);
97             }
98         }
99     }
100 }
101 }
102 else {
103     // モノラル
104     if (sampleSizeInBits == 16) {
105         // 16ビットモノラル
106         while (nBytesRead != -1) {
107             // オーディオストリームからデータを読み込みます
108             nBytesRead = audioInputStream.read(abData, 0, abData.length);
109             if (nBytesRead >= 0) {
110                 // オーディオデータをミキサーに書き込みます
111                 int nBytesWritten = line.write(abData, 0, nBytesRead);
112                 for (int i = 0; i < nBytesRead; i += 2) {
113                     short c;
114                     c = (short)(abData[i] & 0xff | (abData[i+1] << 8));
115                     System.out.println("" + c);
116                 }
117             }
118         }
119     }
120     else { // sampleSizeInBits == 8
121         // 8ビットモノラル
122         while (nBytesRead != -1) {
123             // オーディオストリームからデータを読み込みます
124             nBytesRead = audioInputStream.read(abData, 0, abData.length);
125             if (nBytesRead >= 0) {
126                 // オーディオデータをミキサーに書き込みます
127                 int nBytesWritten = line.write(abData, 0, nBytesRead);
128                 for (int i = 0; i < nBytesRead; i++) {
129                     short c;
130                     c = (short)(abData[i] & 0xff);
131                     System.out.println("" + c);
132                 }
133             }
134         }
135     }
136 }
137 // ラインからキューに入っているデータを排出します

```

```
138     line.drain();
139     // ラインを閉じます
140     line.close();
141
142     System.exit(0);
143 } catch (Exception e) {
144     e.printStackTrace();
145     System.exit(1);
146 }
147 }
148 }
```

## 参考文献

- [1] 大浦 拓哉, Ooura's Mathematical Software Packages, <http://www.kurims.kyoto-u.ac.jp/~ooura/index-j.html>
- [2] 三井 康之, Java による波動方程式の数値解析, 2001 年度桂田研卒業研究, <http://www.math.meiji.ac.jp/~mk/labo/pdf/2001-mitsui.pdf>
- [3] 田中 宏和, Java で HelloWorld, <http://www.hellohiro.com/>
- [4] 奥村 晴彦,  $\text{\LaTeX}2\epsilon$  美文書作成入門, 技術評論社
- [5] 戸川 隼人, 演習と応用 Java, サイエンス社