メモ

桂田 祐史

2008年1月30日

卒研のWWWページ http://www.math.meiji.ac.jp/~mk/labo/2007/

1 全般的な状況

- 音を扱う人達 (I君,KT君) 向けには、以下に示すように、Waveファイルの読み込みとFFTで手助け。
- O君 MitsuiWorld で動画ファイル出来るようになる？
  そうしたらどんどん作って熱方程式博物館とか波動方程式博物館が出来そう。
- KM君円盤領域上の関数の可視化出来るようになる？GLSCを越えることができるか。
  http://www.math.meiji.ac.jp/~mk/labo/2007/heat2d-disk.tar.gz にあるCプログラム (可視化にgnuplotを使った) をJavaに移植したい。
  有限要素法で計算した結果の可視化用のプログラムも出来ると思うのだが、これは時間切れか。Zバッファを理解する方が良いか。
- SW 近似少なくとも陽解法ならば、汎用性の高いプログラムが出来る？KB君の仕事に期待するが、陰解法をする時間は残っていないか。
- N君 MitsuiWorld の可視化のパラメーターを対話的に変更してというか、「手で持って眺めまわす」ことが出来るようになる？
  これは使い勝手をかなり上げると期待している。

2 Java で FFT (2)

2.1 WAVE ファイルを読む

これは手助けすることにした。

Waveファイルを読み込んで、音を鳴らしながら、音声データを数値表示するプログラムReadWave.java[1] を作成した。Waveファイルの読み込みをする部分は、Javaで Hello World!

[1] http://www.math.meiji.ac.jp/~mk/labo/2007/ReadWave.java

サウンド編[2] にある HelloWorldSound.java[3] を参考にした。オーディオ・データの形式は、Java のクラス・ライブラリィが調べてくれるので、「尋ねる」だけである。音のビット・データを Byte の配列として読み込むわけだが、それをどのように数値として解釈するかは、色々場合分けが必要で結構面倒である (定番の、16 ビット・ステレオ、符号付き PCM データに限れば簡単だと思うが…)。

```
javac ReadWave.java
```

とコンパイルする。解析したい Wave ファイルのパス名を与えて実行する。例えば

```
java ReadWave piano.wav
```

とすると、piano.wav を入力として仕事をする。数値データの出力は膨大なものとなるので、次のようにリダイレクトするのが良いかもしれない。

```
java ReadWave piano.wav > piano.txt
```

チェック用に piano.wav[4] を ReadWave で処理した結果 piano.txt[5] を載せておく。

ソース・プログラムは案外長いが、(i) 16 ビット・ステレオ、(ii) 8 ビット・ステレオ、(iii) 16 ビット・モノラル、(iv) 8 ビット・モノラル、と 4 つに場合分けしてあるからである (本当はこれにエンディアンの違いと、符号の有無の違いで、さらに場合分けする必要があるが、それは無視した)。

16 ビット・ステレオの場合 (リトル・エンディアン, 符号付き)

```
    nBytesRead = audioInputStream.read(abData, 0, abData.length);

    if (nBytesRead >= 0) {
       for (int i = 0; i < nBytesRead; i += 4) {
           short left, right;
           left  = (short)(abData[i  ] | (abData[i+1] << 8));
           right = (short)(abData[i+2] | (abData[i+3] << 8));
           System.out.println("" + left + " " + right);
       }
    }
```

2.1.1 ReadWave.java

```
1  //
2  // ReadWave.java
3  //   2008/1/30
4  //   2008/2/13 NoOwEBByte tB
5  //
```

[2] http://www.hellohiro.com/sound.htm
[3] http://www.hellohiro.com/src/HelloWorldSound.java
[4] http://www.math.meiji.ac.jp/~mk/labo/2007/piano.wav
[5] http://www.math.meiji.ac.jp/~mk/labo/2007/piano.txt

```
 6
 7  import java.io.File;
 8  import javax.sound.sampled.*;
 9
10  public class ReadWave {
11    private static final int EXTERNAL_BUFFER_SIZE = 128000;
12    public static void main(String[] args) {
13      int frameSize;       //
14      int sampleSizeInBits; // f[^lrbg (168)
15      int channels;        // `l (2XeI, 1m)
16      float sampleRate;    // TvO[g (1b...)
17      boolean isStereo;    // XeI
18      boolean isBigEndian; // rbOGfBA (oCg)
19      if (args.length == 0) System.exit(0);
20      try {
21        // FileNXCX^X
22        File soundFile = new File(args[0]);
23        // I[fBIXg[
24        AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(soundFile);
25        // I[fBItH[}bg
26        AudioFormat audioFormat = audioInputStream.getFormat();
27
28        // f[^CIuWFNg
29        DataLine.Info info = new DataLine.Info(SourceDataLine.class,audioFormat);
30        // wf[^CvC
31        SourceDataLine line = (SourceDataLine) AudioSystem.getLine(info);
32        // wI[fBI`CJ
33        line.open(audioFormat);
34        // Cf[^o\
35        line.start();
36
37        // f[^`
38        channels = audioFormat.getChannels();
39        isStereo = (channels == 2);
40        isBigEndian = audioFormat.isBigEndian();
41        frameSize = audioFormat.getFrameSize();
42        sampleSizeInBits = audioFormat.getSampleSizeInBits();
43        sampleRate = audioFormat.getSampleRate();
44        System.out.println("#original file: " + args[0]);
45        System.out.println("#number of channels: " + channels);
46        System.out.println("#sampling rate: " + sampleRate);
47        System.out.println("#number of bits per sample: " + sampleSizeInBits);
48        System.out.println("#FrameSize: " + frameSize);
49        System.out.println("#isBigEndian: " + isBigEndian);
50        if (audioFormat.getEncoding() == AudioFormat.Encoding.PCM_SIGNED) {
51    System.out.println("#PCM Signed");
52        }
53        else if (audioFormat.getEncoding() == AudioFormat.Encoding.PCM_UNSIGNED) {
54    System.out.println("#PCM Unsigned!!!");
55    System.exit(0);
56        }
57        else {
58    System.out.println("#NO PCM!!");
59    System.exit(0);
60        }
61
62        // f[^AAl\
63        int nBytesRead = 0;
```

```
64        byte[] abData = new byte[EXTERNAL_BUFFER_SIZE];
65        if (isStereo) {
66    if (sampleSizeInBits == 16) {
67                // 16rbgXeI (tc[)
68        while (nBytesRead != -1) {
69    // I[fBIXg[f[^
70    nBytesRead = audioInputStream.read(abData, 0, abData.length);
71    if (nBytesRead >= 0) {
72        // I[fBIf[^~LT[
73        int nBytesWritten = line.write(abData, 0, nBytesRead);
74        for (int i = 0; i < nBytesRead; i += 4) {
75    short left, right;
76    left = abData[i]; if (left < 0) left += 256;
77    left  = (short)(left | (abData[i+1] << 8));
78    right = abData[i+2]; if (right < 0) right += 256;
79    right = (short)(right | (abData[i+3] << 8));
80    System.out.println("" + left + " " + right);
81        }
82    }
83        }
84    }
85    else { // sampleSizeInBits == 8
86                // 8rbgXeI (tc[)
87        while (nBytesRead != -1) {
88    // I[fBIXg[f[^
89    nBytesRead = audioInputStream.read(abData, 0, abData.length);
90    if (nBytesRead >= 0) {
91        // I[fBIf[^~LT[
92        int nBytesWritten = line.write(abData, 0, nBytesRead);
93        for (int i = 0; i < nBytesRead; i += 2) {
94    short left, right;
95    left = abData[i];    if (left < 0)  left  += 256;
96    right = abData[i+1]; if (right < 0) right += 256;
97    System.out.println("" + left + " " + right);
98        }
99    }
100       }
101   }
102       }
103       else {
104   // m
105   if (sampleSizeInBits == 16) {
106               // 16rbgm
107       while (nBytesRead != -1) {
108   // I[fBIXg[f[^
109   nBytesRead = audioInputStream.read(abData, 0, abData.length);
110   if (nBytesRead >= 0) {
111       // I[fBIf[^~LT[
112       int nBytesWritten = line.write(abData, 0, nBytesRead);
113       for (int i = 0; i < nBytesRead; i += 2) {
114   short c = abData[i];
115   if (c < 0) c += 256;
116   c = (short)(c | (abData[i+1] << 8));
117   System.out.println("" + c);
118       }
119   }
120       }
121   }
```

```
122     else { // sampleSizeInBits == 8
123                 // 8rbgm
124         while (nBytesRead != -1) {
125     // I[fBIXg[f[^
126     nBytesRead = audioInputStream.read(abData, 0, abData.length);
127     if (nBytesRead >= 0) {
128         // I[fBIf[^~LT[
129         int nBytesWritten = line.write(abData, 0, nBytesRead);
130         for (int i = 0; i < nBytesRead; i++) {
131     short c = abData[i];
132     if (c < 0) c += 256;
133     System.out.println("" + c);
134         }
135     }
136         }
137     }
138         }
139         // CL[f[^ro
140         line.drain();
141         // C
142         line.close();
143 
144         System.exit(0);
145       } catch (Exception e) {
146         e.printStackTrace();
147         System.exit(1);
148       }
149     }
150  }
```

2.2 Java 用の FFT ライブラリィ

大浦版 FFT の Java への移植はそれなりに面倒そうなので (と言っても、一晩仕事だと思うけれど)、他の可能性も模索することに。

有名 (?) な **FFTPACK** の Java への移植である **jfftpack** を使ってみるか。

2.2.1 jfftpack を試す

P. N. Swartztaruber の FFTPACK[6] は古くからあるライブラリィだが、Java への移植である **jfftpack** (by Baoshe Zhang `baoshe.zhang@uleth.ca`) がある。

オリジナルの FFTPACK については、次の解説を書いた。

- 「FFT についてのメモ」[7]

- FFTPACK[8]

jfftpack をコンパイルして作った jfftpack.jar[9] を載せておく。

次のテスト・プログラムを試そう。

[6] `http://www.netlib.org/fftpack/`
[7] `http://www.math.meiji.ac.jp/~mk/labo/text/fft-lecture.pdf`
[8] `http://www.math.meiji.ac.jp/~mk/labo/text/fftpack-index.pdf`
[9] `http://www.math.meiji.ac.jp/~mk/labo/2007/jfftpack.jar`

```
/*
 * testfft.java
 *   FFTPACK の Java版 jfftpack (http://www.netlib.org/fftpack/ で入手可)
 *   http://www.math.meiji.ac.jp/~mk/labo/2007/jfftpack.jar に載せる
 *
 *   jfftpack.jar の位置を環境変数 CLASSPATH で指定する。
 *   (1) Windows では set CLASSPATH=%CLASSPATH%;どこか\jfftpack.jar
 *   (2) Linux では export CLASSPATH=$CLASSPATH:どこか/jfftpack.jar
 *
 *   あるいは java コマンドの -classpath オプションを使う。
 *     java -classpath どこか/jfftpack.jar:. testfft
 *
 */

import ca.uol.aig.fftpack.*;

public class testfft {
    public static void main(String args[]) {
int i,n;
double dx,x;
double [] a;
n = 1024;
a = new double [n];
RealDoubleFFT myfft = new RealDoubleFFT(n);
// n等分点上の関数値を求める
dx = 2 * Math.PI / n;
for (i = 0; i < n; i++) {
    x = i * dx;
    a[i] = 1+2*Math.cos(x)+3*Math.sin(x)+4*Math.cos(2*x);
}
// 離散Fourier変換
myfft.ft(a);
// Fourier係数を求める
a[0] /= n;
for (i = 1; i < n; i++)
    a[i] = a[i] / (n / 2.0);
// 1,2,3,4が現われますやら…
for (i = 0; i < 10; i++)
    System.out.printf("i=%d: %f\n", i, Math.abs(a[i]));
//     System.out.println("i=" + i + ":" + Math.abs(a[i]));
    }
}
```

1. testjfftpack-unix.tar.gz[10], — Knoppix 用
2. testjfftpack-win.lzh[11] — Windows 用

Knoppix の場合

```
knoppix$ tar xzf testjfftpack-unix.tar.gz
knoppix$ cd testjfftpack-unix
knoppix$ export CLASSPATH=/home/knoppix/testjfftpack-unix/jfftpack.jar:.

あるいは export CLASSPATH=‘pwd‘:.

knoppix$ javac testfft.java
knoppix$ java testfft
i=0: 1.000000
i=1: 2.000000
i=2: 3.000000
i=3: 4.000000
i=4: 0.000000
i=5: 0.000000
i=6: 0.000000
i=7: 0.000000
i=8: 0.000000
i=9: 0.000000
knoppix$
```

注 (2008/2/6 分かったこと): CLASSPATH を設定する以外に、javac や java に -classpath jfftpack.jar:. と指定するという手もある (短縮形 -cp jfftpack.jar:. も使える)。

2.2.2 どうやって jfftpack.jar を作るか

ca.uol.aig.fftpack という package 名であることに注意。

```
mkdir -p ca/uol/aig/fftpack              (ディレクトリィを用意)
cp -p どこか/*.java ca/uol/aig/fftpack   (ソース・プログラムを用意)
javac ca/uol/aig/fftpack/*.java          (ソース・プログラムをコンパイル)
jar -cvf どこか/fftpack.jar .            (jar ファイルを作る)
jar -tvf どこか/fftpack.jar              (jar ファイルの中身を確認する)
```

[10] http://www.math.meiji.ac.jp/~mk/labo/2007/testjfftpack-unix.tar.gz
[11] http://www.math.meiji.ac.jp/~mk/labo/2007/testjfftpack-win.lzh

2.3 これからどうする

これで Wave ファイルから PCM データを読み込むこと、またそれを離散 Fourier 変換する手段は確保できた。

逆に適当に (例えば波動方程式を解いて) 作った数値データから Wave ファイルを作ったりとか、周波数の解析をするとか、まだ準備することはあるが、色々試せる段階に来たのではないだろうか。

3 MitsuiWorld の座標変換

(作業中…これは学生にやってもらいたいぞ)

`xmin`, `xmax`, `ymin`, `ymax`, `zmin`, `zmax` を x_*, x^*, y_*, y^*, z_*, z^* とする。

$$\Omega := [x_*, x^*] \times [y_*, y^*] \times [z_*, z^*]$$

$\varphi_J \colon \Omega \to \mathbf{R}^3$ を