1 音声処理のため (だけ) の MATLAB 入門

音声データを用いて MATLAB の基本機能を概観する。

1.1 音声データの準備

まず、1 秒程度の音声 (サウンド) ファイルを用意する。Windows Vista の場合は、サウンドレコーダーで MATLAB で直接扱える形式では保存できない。したがって、別のソフトウェアをインストールする必要がある。audacity などフリーのソフトウェアで十分である (XP であれば、サウンドレコーダーで録音できる)。

演習 1 何か適当に 1 秒間しゃべった音声を PC で録音せよ。PCM 16.0 kHz, 16 ビット、モノラルで RIFF 形式 (.wav という拡張子がつく形式) で保存すること。audacity など、高機能なソフトウェアでは、このような設定で保存することができる。

1.2 音声データの読込と再生

音声ファイルを MATLAB に読み込む。MATLAB を起動して、**カレントディレクトリ** のところに、録音したファイルをドラッグアンドドロップしてコピーしておく。このファイルを `xxx.wav` とすると、**コマンドウィンドウ** に次のようにタイプする。ただし、`>>` はプロンプトである。入力する必要はない。

ソースコード 1

```
>> [y, fs, nbits] = wavread('xxx.wav');
```

ここで、`wavread` は引数にファイル名の文字列を取りそのファイルのデータを MATLAB に読み込む関数である。この式は、`wavread` が 3 つの返り値を持ち、1 番目の返り値を変数 `y`、2 番目の返り値を変数 `fs`、3 番目の返り値を変数 `nbits` に代入することを表す。

これらの値は**ワークスペース** を見るとわかる。また、

```
>> fs

ans =
        16000
```

というように **コマンドウィンドウ** に値を確認したい変数名 (ここでは `fs`) を入力することでも見ることができる。これは、MATLAB で `fs` という単一の変数からなる式を計算した結果の表示である。

`wavread` の返り値が何を意味するかは、`help` 関数で知ることができる。

```
>> help wavread
```

```
  WAVREAD   Microsoft WAVE (``.wav'') サウンドファイルの読み込み

  Y = WAVREAD(FILE) は、文字列 FILE で指定された WAVE ファイルを読み込み、サ
  ンプリングされたデータを Y に出力します。拡張子が指定されていない場合は、
  ``.wav'' を付け加えます。振幅値は、範囲 [-1,+1] です。

  [Y,FS,NBITS] = WAVREAD(FILE) は、ヘルツ単位でサンプリングレート (FS) を出
  力し、ファイル内のデータを符号化するために使うサンプルあたりのビット数
  (NBITS) を出力します。
(以下略)
```

この説明からわかるように上記の**ソースコード** 1 の場合、音声ファイル xxx.wav に含まれるデータは変数 y 、サンプリングレートは変数 fs 、符号化のためのビット数は nbits に格納される。

演習 2 自分で録音したファイルを wavread を使って MATLAB に読み込み、fs と nbits の値を確認せよ。

このデータを再生するための関数も MATLAB には用意されている。関数名がわからないときに機能に関するキーワードから関数を検索する関数が MATLAB には用意されている。

例えば、次のように**コマンドウィンドウ**に入力する。

```
>> lookfor 再生
```

すると、関数の説明文に「再生」という文字列を含む関数名の一覧が表示される。

```
movie                    - 記録されたムービーフレームを再生
movieview                - 再生ボタン付きの MATLAB ムービーを表示
soundview                - リプレイボタンと共に、音声の表示と再生を行います。
sound                    - ベクトルを音声として再生
soundsc                  - 音声としてのベクトルのスケーリングと再生
wavplay                  - Windows のオーディオ出力デバイスを使って音を再生
implay                   - ムービー、ビデオ、イメージ列を再生
```

ここでは、sound を使ってみる。

```
>> sound(y, fs)
```

録音した内容が再生されるはずである。

1.3 波形の表示

音声データを観察するためには、聞くだけでは不十分である。音声を目で見て観察できるようにする (視覚化する) 方法はいくつも考えられてきた。

```
>> plot(y)
```

とすると y の波形が表示される。これは音 y の変位を時間軸にそって直線でつないだものを描画したものである。

y 軸は変位の大きさ、x 軸は時間をあらわす。

描画のために開かれた Figure ウィンドウの + の拡大鏡ツールを選択し、ウィンドウ内で右クリックするとポップアップメニューが表示される。そのズームオプションから、水平方向のズームを選択する。そのようにしてウィンドウ内で左クリックを続けるとズームインしてゆき、拡大された波形が表示される。

このように表示されるデータ y は MATLAB ではどのような値になっているかを見る。ワークスペースの y のところをダブルクリックする。すると、変数エディタというウィンドウが表示される。この 1 列目に実数が並んでいることがわかる。変数エディタのスクロールバーを下に動かすと、列の最後がどこかわかる。最後のデータの左側の数字がこのデータの要素の数をあらわす。例えば、その数字が 44000 だとして説明を進める。

このように MATLAB では、変数 y に複数の数の組を特に宣言することなしに割り当てることができる。この例のように列方向に複数のデータを持つ数の組を列ベクトル (列ベクトル) とよぶ。また、この列ベクトルは、44000×1 行列であるともいう。MATLAB ではこのように、ベクトルも行列も区別なく変数に割り当てることができる。

演習 3 lookfor 関数を用いて、ベクトルの長さを出力する関数を調べよ。(以下、この関数を本資料では、func0102 と表記する。各自、正しい関数名に置き換えて実行すること)

前述のサンプリングレートは、1 秒間のデータ数をあらわす。したがってベクトルの長さをサンプリングレートで割ると、データが何秒間なのかがわかる。

```
>> func0102(y)/fs

ans =
    2.7500
```

この場合は、2.75 秒である。

演習 4 各自録音したファイルの長さ (単位は秒) を計算せよ。

```
>> plot(y)
```

という形式で表示させたときは、グラフの x 軸がデータの数となっていた。見やすくするために、x 軸の単位を秒にする方法を考える。

help plot とすればわかるように、plot 関数は、plot(x, y) という形式で x 軸、y 軸の数値を指定することができる。今回の場合、開始して 1 秒のところを 1.0 となるようなベクトルを作って、それを x とすればよい。1 秒間に 16000 個データがあるということは、1 つ目のデータの開始時刻を 0 とすると、2 つ目のデータの開始時刻は、1/16000 である。3 つ目のデータの開始時刻は 2/16000 である。このようにして、$0, 1/16000, 2/16000, 3/16000, \ldots$ というベクトルを作ればよい。

MATLAB では、このような一定間隔の数列を簡単に作ることができる。例えば、1 から 1 つ刻みで 10 までの数列を作るには、

```
>> x = 1:10

x =
     1     2     3     4     5     6     7     8     9    10
```

というふうに : を使うことで簡単に実現できる。

また、この数列の要素全体を、例えば 10 で割り算したいときには、繰り返し計算をしなくても、

```
>> x = (1:10)/10

x =
  Columns 1 through 6
    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000

  Columns 7 through 10
    0.7000    0.8000    0.9000    1.0000
```

と一度に計算できる。

演習 5 y と同じ長さを持ち、適切な値を持つベクトル x を作成し、plot(x, y) を実行して、x 軸の単位が秒になるグラフを描画せよ。

1.4 ベクトルとしてのデータの操作

変数 y は複数の値を持つ。この値のうち、一部の値だけ利用したいときには、要素番号で指定することができる。例えば、

```
>> y(1)
```

と入力すると、y の最初の値が出力される。この要素番号のことをインデックスとよぶ。MATLAB は java などと異なり、最初のインデックスは 1 であることを注意しなければならない。また、インデックスとしてベクトルを指定することもできる。

```
>> yy = y(1:10)
```

という式では、y の 1 番目から 10 番目の要素の値からなるベクトルが変数 yy に代入され、出力される。なお、

```
>> yy = y(1:10);
```

というように式の後に ; を付けると、値は出力されなくなる。値が多いときなどは、出力に時間がかかるので、; を付けるのを忘れないように。

この機能を利用すると、データから簡単に必要な部分だけを取り出すことができる。

演習 6 各自のデータの 0.1 秒の時点から 0.5 秒間のデータを変数 y1 に代入せよ。また、そのデータの波形を表示したり、再生し、正しく取り出されていることを確認せよ。

インデックスにベクトルを指定する場合、その値は順番でなくてもよい。例えば、

```
>> x1 = 1:10;
>> x1([1 5 2 3])

ans =
     1     5     2     3
```

MATLAB では [ ] で囲むことによって、ベクトルを作ることができる。上記の例では、行方向にデータが複数あるので、行ベクトルである。

刻み幅が 1 でない数列を作る場合には、

```
>> x1 = 1:2:11

x1 =
     1     3     5     7     9    11
```

というように、刻み幅 (この場合 2) を指定し、: を 2 つ使って数列を定義すればよい。

刻み幅は整数でなくてもよい。また、正の数でなくてもよい。

演習 7 インデックスに指定するベクトルを工夫して、各自のデータを反転したデータを作成せよ。反転とは、例えば、データが、[1 5 2 4] であれば、[4 2 5 1] のようにする操作のことである。反転できたら、そのデータの波形を表示したり、再生し、正しく反転できていることを確認せよ。

演習 8 演習 5 と同様のことを刻み幅を工夫して数列を作成することによって実現せよ。(系列全体を割り算しなくてもできる)

なお、[ ] を用いて、行ベクトルを作るときには、

```
>> r1 = [1; 5; 2; 4]

r1 =
     1
     5
     2
     4
```

というように、; を用いる。

演習 9 列ベクトルは行ベクトルを「転置」することでも作ることができる。行列の「転置」とはどのような操作かを思い出し、lookfor 関数で「転置」を行う関数を調べて、行ベクトルを転置することで列ベクトルを作ってみよ。

ベクトルを使って、より大きなベクトルを作ることができる

```
>> x1=1:3

x1 =
     1     2     3

>> x2 = [x1 x1 x1]

x2 =
     1     2     3     1     2     3     1     2     3
```

このようにすれば簡単に繰り返しデータを作ることができる。

演習 10 各自録音したデータの音のある部分を 0.2 秒間取り出し、y2 に代入せよ。この y2 を 4 回繰り返したデータを作成し、y2x4 に代入せよ。そのデータの波形を表示したり、再生し、正しく反転できていることを確認せよ。

演習 11 **演習** 10 と同じことが、関数 repmat を利用して実現できる。help 関数で repmat の使い方を調べ、repmat を用いて、**演習** 10 と同じことを実現せよ。

1.5 演算

変位に一定値を掛けると音が大きくなる。割り算のときと同様に、系列全体に掛け算することができる。

```
>> x1=1:3

x1 =
     1     2     3

>> x1*2

ans =
     2     4     6
```

演習 12 各自録音したデータに 1.1 などいくつかの適当な値を掛けてデータを作成せよ。そのデータの波形を表示したり、再生してどのようになるか確かめよ。

y に余り大きな数を掛けると、最大値が 1 より大きくなったり、最小値が -1 より小さくなったりすることがある。help sound で sound のヘルプを見ればわかるが、sound は「Y の値は、範囲-1.0 < = y <= 1.0 と仮定します。この範囲外の値は切り取られます。」と書かれている。したがって、1.0 より大きな値や -1.0 より小さい値の部分は正しく再生されず、結果として音が歪んでしまう。

歪ませることなく、最大限音を大きくすることを考える。

演習 13 y の最大の要素を返す関数を探し、最大値を maxy に代入せよ。また、最小の要素を返す関数を探し、最小値を miny に代入せよ。

演習 14 次の MATLAB スクリプト (スクリプトとは、MATLAB の式が複数続いたものとする) はどういう計算しているか help 関数などを利用して理解せよ。ただし**演習** 13 で探した最大の要素を返す関数を func0112 とする。

```
>> d=func0112(abs([maxy miny]))

d =
    0.7789

>> y=y/d;
>> sound(y,fs)
```

演習 14 と同様のことをする関数が用意されている。soundsc である。

演習 15 演習 12 で作成した様々なファイルを soundsc 関数で再生し、どのような結果が得られるか確かめよ。

soundsc は実は MATLAB で書かれたプログラムである。ソースコードは

```
>> type soundsc
```

とすれば表示される。

演習 16 soundsc がどのような計算をおこなっているか確かめよ。

1.6 音の生成

最も基本的な音である正弦波 (サイン波) を作成する。440Hz の正弦波は、時間を t とすると $\sin(2\pi ft)$ (ただし、$f = 440$) とあらわせる。MATLAB では、多くの関数が、行列を引数として、単一の数字と同じ感覚で計算できる。その機能を用いると、周波数が 440Hz の正弦波をサンプリングレートが 16000 で 1 秒間生成するスクリプトは次のように書ける。(pi は 初期値として π が与えられた変数である。)

```
>> Fs=16000;
>> t=0:1/Fs:1;
>> y=sin(2*pi*440*t);
>> sound(y, Fs);
>> plot(t,y)
```

このように、数式とほとんど同じ感覚でプログラミングできる。

演習 17 様々な周波数の正弦波を生成して再生してみよ。

この波形の一部だけを表示するためには、表示したい範囲のインデックスを指定すればよい。例えば、1001 点目から 1000 点分表示するには、次のようにすればよい。

```
>> t1=1001;
>> t2=t1+1000;
>> plot(y(t1:t2))
```

演習 18 上の例では、x 軸の単位が点数となってしまう。x 軸の単位が秒となるようにプロットせよ。

演習 19 適当な波形を 4 周期分プロットせよ。

1.7 音の重ね合わせ

正弦波を足し合わせることで、様々な音を作ることができる。次の例は、440Hz の正弦波 y440 と 660 Hz の正弦波 y660 を足し合わせて y を作る。

```
>> Fs=16000;
>> t=0:1/Fs:1;
>> y440=sin(2*pi*440*t);
>> y660=sin(2*pi*660*t);
>> y=y440+y660;
>> plot(y(1:200));
```

プロットを見ればわかるように、y は最も変位の大きいところでは、2 に近い値を取る。**1.5** に記述したように、sound 関数は 1 以上の値を再生できない。したがって、soundsc 関数を使うか、最大の変位を 1 以下になるようにしなければならないことに留意すること。

MATLAB では、正弦波以外にも、波を生成する関数が用意されている。

演習 20 MATLAB の関数を用いていろいろな波を発生し、それらをプロットして、どのような特徴があるか調べよ。また、それらを再生して、どのような音なのかを確認せよ。(ヒント: 「lookfor 発生」 と入力すればいろいろな関数が見つかる。)

生成した信号をはじめとする MATLAB 内のデータはファイルとして書き出すことができる。

演習 21 wavwrite 関数の使い方を調べて自分で作成、加工したデータをファイルとして保存せよ。

課題

以下の課題をパワーポイントファイルに作成し、授業支援システムのま Navi の第 2 回目の「準備課題」のところでアップロードすること。パワーポイントファイルは、18 ポイント以上の大きさの文字を使って作成すること。締切は 4/20(火) 午前 5 時とします (月曜日の夜中です)。自動で締め切ってしまいますので、十分に注意して下さい。

1. **演習** 5 を解答せよ。(図を書け)
2. **演習** 8 を解答せよ。(MATLAB スクリプトを書け)
3. **演習** 11 を解答せよ。(MATLAB スクリプトを書け)
4. **演習** 12 を解答せよ。(MATLAB スクリプトを書け)
5. この資料を読んで、わからなかった点があれば、わからなかった点が解消されるような回答が期待できるような具体的な質問を考えて書くこと。