1　波形データの生成

まず、1 次元のディジタルデータの代表として、音データを取り上げる。最も単純な音のひとつに純音がある。純音は正弦波で表される音である。物理などの本を見ると、純音は次のような式で表されることがわかる。

$$y = A\sin(2\pi f t) \tag{1}$$

ここで、y は音圧、A は振幅、f は周波数、t は時刻である。

この式に基づいて、MATLAB で音のディジタルデータを作成するプログラムを作成する。

実際のデータを作成するためには、上記の式の変数について固有の値を決めなければならない。A, f は 1 つの値 (スカラーとよぶ) を定めればよい。しかし、t については、例えば、1 秒間のデータを作成するときには、開始時刻を 0 (秒) とすると、時刻 0 秒から 1 秒まで変化する。式で書くと $0 \leq t \leq 1$ となる。物理的な世界では、t は連続的に変化する。

この連続的な量をコンピュータで扱う最も一般的な方法は、均等な微小な間隔の値の列として表現することである。例えば、1/10000 秒の間隔とすると、$0 \leq t \leq 1$ という範囲は、$0, 1/10000, 2/10000, 3/10000, \ldots, 9999/10000, 1$ という　(1)　個の数の列で表されるのである。

練習 1　下線部 (1) を埋めよ。

この 1/10000 をサンプリング周期と呼ぶ。また、この周期に対応する周波数をサンプリング周波数と呼ぶ。

MATLAB で 440Hz の音を 1 秒間分生成してみる。コマンドウィンドウに次のように入力する。(先頭に数字がある場合は行番号をあらわす。また、>> はプロンプトである。したがって両方とも入力する必要はない)

ソースコード 1: 正弦波の生成

```
1 >> t=0:1/8000:1;
2 >> f=440;
3 >> a=0.8;
4 >> y=a*sin(2*pi*f*t);
```

1 行目では、時刻を表わす数列を作成している。: は、MATLAB ではいろいろな意味を持つ記号であるが、ここでは、数列を生成するために利用されている。

: がどのようにはたらくかを確かめてみる。

```
>> n=0:10

n =

     0     1     2     3     4     5     6     7     8     9    10

>>
```

このように、開始値 : 終了値 という形で指定する (この例の場合、開始値は 0、終了値は 10) と、0 から 10 までの整数列が生成され、n に代入される。(n は変数である。MATLAB では、変数自体には型がないため、数列であっても行列であっても = で代入できる)

先ほどの t の例は : が 2 つ使われている。この場合は、開始値 : 間隔 : 終了値 となる。

例えば、

```
>> m=-1:0.5:2

m =

   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000

>>
```

のようになる。

行の最後の ; は、値の表示を抑制する。

```
1 >> a
2
3 a =
4
5     0.8000
6
7 >> a;
8 >>
```

このように、変数をプロンプトに入力すると値が表示される。しかし、 7 行目のように ; をつけると何も表示されない。

ソースコード 1 の 4 行目は、式 (1) を MATLAB でプログラミングしている。ほとんど式と同じ形で表現できることに注目してほしい。この pi は、MATLAB であらかじめ用意されている変数で π の値を持つ。

sin は、三角関数の sin である。MATLAB で用意されている関数については、help コマンドで説明を見ることができる。

```
>> help sin
  SIN   ラジアン単位の正弦

  SIN(X)は、 Xの要素の正弦値を出力します。

  参考 asin, sind.

    ヘルプブラウザ内の参照ページ
       doc sin

>>
```

help の説明では大文字になっているが、実際に使うときには小文字でないといけないことに注意すること。この説明では、引数に関して X の要素の と書かれている。また、**ソースコード 1** の 4 行目を見ればわかるように、この sin は引数に数列をとれる。

```
>> sin(0:0.1:1)

ans =

  Columns 1 through 7

         0    0.0998    0.1987    0.2955    0.3894    0.4794    0.5646

  Columns 8 through 11

    0.6442    0.7174    0.7833    0.8415

>>
```

ans は、直前の計算結果が自動的に代入される変数である。このように、sin は、数列を引数に与えると、計算結果も数列となる。このように、 MATLAB の多くの関数は数列や行列を引数に取ることができる。

ソースコード 1 の 4 行目では、t は数列である。その前の 2*pi*f はスカラーであるため、2*pi*f*t は、数列 t の各々の要素 (全ての要素) を $2\pi f$ 倍する。この $2\pi f$ 倍された数列に対して sin を計算し、その結果の数列

を a 倍している。このような数列は、MATLAB ではベクトルとして扱う。

作成した音データのベクトルを出力するための関数が sound である。**ソースコード 1** で作成した y は次のようにして出力する。

```
>> sound(y, 8000)
>>
```

ラの音が 1 秒間聞こえるはずである。

練習 2 880Hz の cos 波を 1 秒間生成せよ。また音を出力して確認せよ。

練習 3 次の条件の sin 波を生成し、音を出力して確認せよ。

1. 262Hz, 0.5 秒間
2. 440Hz, 2 秒間

練習 4 サンプリング周波数 16KHz で 523Hz の cos 波を 1 秒間生成せよ。またその音を出力して確認せよ。

2　1 次元データのプロット

音を生成したり、加工したり、録音した場合には、もちろん、音を再生・出力して確認すべきである。しかし、音は聞こえ方が人によってかなり異なるため、聞く以外の方法でも確認した方がよい。

まず、時間に対する音圧の変化のグラフで確認する方法を取り上げる。(時間波形のプロットと呼ばれることが多い。)

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

グラフは表示されるが、これではほとんどよくわからないだろう。

そこで拡大してみる。グラフが表示されている **Figure 1** というウィンドウの+ が中に書かれている虫眼鏡アイコンを選択し、グラフ上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。そのポップアップメニューからズームオプションを選択し、水平方向のズームを選ぶ。その後は、適宜、左クリックをすると、上手く操作できれば、sin 波のグラフを見ることができるだろう。

このとき、グラフの縦軸 (y 軸) は、音圧 (変位ともいう) をあらわしている。しかし、グラフの横軸 (x 軸) は、時刻ではなく、データの番号をあらわしている。グラフの概形を見るだけなら、このプロット方法でも十分である。しかし、データのどの時刻でどうなっているかを知るためには、横軸に時刻が表示されるようにプロットすべきである。

```
>> plot(t,y)
```

このようにすると、横軸は時刻になる。

いちいち拡大ツールを使うのは面倒なので、データの一部をプロットする方法を紹介する。

MATLAB では、ベクトルの要素を次のように指定する。

```
>> y(1)

ans =

     0

>>
```

このように () で指定する。ちなみに、java などとは違って、インデクスは 0 でなく 1 から始まる。この例は、y の最初の要素を表示している。

インデクスをベクトルで指定することもできる。

```
>> y(1:10)

ans =

  Columns 1 through 7

         0    0.2710    0.5099    0.6886    0.7858    0.7902    0.7010

  Columns 8 through 10

    0.5290    0.2945    0.0251

>>
```

これは y の最初から 10 点の値を出力している。このように y の一部を簡単に取り出せる。したがって、次のように y の一部をプロットできる。

```
>> plot(y(1:10))
```

横軸を対応する時刻にするためには、次のようにすればよい。

```
>> r=1:10;
>> plot(t(r),y(r))
```

練習 5 ソースコード 1 の y について次の問に答えよ。ただし、横軸は時刻となるようにプロットすること。

1. 最初から 50ms をプロットせよ。
2. 最初から 1 周期分をプロットせよ。

3 時間波形の重ね合わせ

音というデータでは、同時に複数の音を重ねることができる。複数の音が同時に鳴るのは、時間波形の足し算で実現できる。

ソースコード 2: 時間波形の重ね合わせ

```
>> fs=8000;
>> t=0:1/fs:1;
>> a=0.3;
>> y523=a*sin(2*pi*523*t);
>> y660=a*sin(2*pi*660*t);
>> y784=a*sin(2*pi*784*t);
>> sound(y523+y660+y784,fs)
>> r=1:100;
>> yy=y523+y660+y784;
>> plot(t,yy)
```

プロットを見るとわかるように、新しく生成された時間波形の振幅は各々の波の振幅よりも大きな値になっていることがわかる。

周波数がもう少し近い音を重ね合わせてみる。

```
>> y438=a*sin(2*pi*438*t);
>> y442=a*sin(2*pi*442*t);
>> sound(y438+y442,fs)
```

2 つの音ではなく、1 つの音が振幅を変化させて鳴っているように聞こえるだろう。これはいわゆる「うなり」という現象である。

振幅が同じで周波数が異なる正弦波を足し合わせることを数式にすると次のようになる。(周波数は f_1, f_2 とする)

$$yy = a\sin(2\pi f_1 t) + a\sin(2\pi f_2 t) \tag{2}$$

練習 6 yy を変形して、次のような形にせよ。

$$yy = a_{yy}\cos(2\pi f_b t)\sin(2\pi f_a t) \tag{3}$$

変形途中の式も書くこと。

また、耳で聞いて、うなりは 1 秒間に何回強弱が起きているかを観測せよ。なぜ、その回数になるかが説明できる式を考えて、その式に基づいて説明せよ。

練習 7 2 つの音の周波数をいろいろと変化させて、うなりとして聞こえる場合と 2 つの音に聞こえる場合にはどのような違いがあるかを調べよ。どのようにして調べたかを述べよ。

式 (3) を MATLAB で計算するためには次のようにする。

```
>> yy=ayy*cos(2*pi*fb*t).*sin(2*pi*fa*t);
>> sound(yy,fs)
```

`ayy, fb, fa` などは各自の計算結果を用いて指定せよ。

練習 8 `.*` という演算子がどのような計算をおこなうか、

```
>> help .*
```

として確かめよ。

式 (3) の $a_{yy}\cos(2\pi f_b t)$ の部分は振幅を時間の関数によって変化させる、というふうにとらえられる。このように、信号を振幅を時間の関数で変化させることを振幅変調と呼ぶ。

音を重ね合わせて音を生成すると元の振幅より大きくなる。`help` コマンドを用いるとわかるが、`sound` 関数は、再生できるデータの変位に範囲があり、その範囲外の値は、範囲の最大・最小値とされてしまう (クリップされるという)。

音がクリップされて再生されると歪んでしまう。例えば、次のようにすると音は歪んでしまう。

```
>> sound(20*sin(2*pi*440*t),fs)
```

歪ませないためには、`sound` に与えるデータを範囲に入るようにしなければならない。つまり、波形を変化させずに最大値が 1.0 を越えないようにし、最小値は -1.0 より小さくならないようにしなければならない。(このように、元の値の意味を変えずに、値を制限に応じて変換することを正規化と呼ぶ)

MATLAB でこのような正規化するためには次のようにする。

ソースコード 3: `sound` 関数のための正規化

```
1 >> ymin=_____(y);              % y の最小値を求めて ymin に代入
2 >> ymax=_____(y);              % y の最大値を求めて ymax に代入
3 >> d=_____(____([ymin ymax]))% ymin と ymax の絶対値が大きい方を d に代入
4 >> y=y/d;                      % y を d で割る(正規化する)
5 >> sound(y,fs)                 % 歪まないことを確認
```

MATLAB で用意されている関数にどのようなものがあるかを調べるには、`lookfor` コマンドが便利である。`lookfor` コマンドは、次のように使う。

```
>> lookfor 最大
```

練習 9 **ソースコード 3** の空欄を適宜埋めて完成させよ。

なお、**ソースコード 3** と同様の処理をする関数が MATLAB には用意されている。`soundsc` である。MATLAB で用意されている関数には、sin のように組み込み関数もあるが、`soundsc` は MATLAB で記述されている。どのようになっているかは、`type` コマンドで確認できる。

```
>> type soundsc
```

4 時間波形の連結

音楽で「ド、レ、ミ」というフレーズを生成するときや、例えば、音声で「アツイ (暑い)」と発話するときには、「ド」が終了した時刻の次に「レ」がくる、というように時間波形が順々に連結されたようになる。

ソースコード 1 のように生成した時間波形は、行ベクトル (行列の行方向にのびるベクトル) になっている。

MATLAB で行ベクトルを連結するには次のようにする。

```
1  >> v1=[0 1 2]
2
3  v1 =
4
5       0     1     2
6
7  >> v2=[3 4]
8
9  v2 =
10
11      3     4
12
13 >> v3=[5 6 7 8]
14
15 v3 =
16
17      5     6     7     8
18
19 >> v=[v1 v2 v3]
20
21 v =
22
23      0     1     2     3     4     5     6     7     8
24
25 >>
```

19 行目のように MATLAB では `[]` を使って行ベクトルを並べると、その順に連結される。

練習 10 次のソースコードを埋めて、ドレミというフレーズを生成し出力してみよ。

```
>> do=--------------------------------;     % ドを 0.5秒間生成し do に代入
>> re=--------------------------------;     % レを 0.5秒間生成し re に代入
>> mi=--------------------------------;     % ミを 0.5秒間生成し mi に代入
>> y=-------------------;                   % ドレミを生成し y に代入
>> soundsc(y, fs);                         % 出力して確認
```

5 音声データの読み込み

まず、1 秒程度の音声 (サウンド) ファイルを用意する。Windows Vista/7 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)。

練習 11 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。

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。

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

`wavread` は音声ファイルのデータを MATLAB に読み込む関数である。この音声ファイルのデータは `sound` 関数で出力できる。

練習 12 `help` コマンドで `wavread` の返却値の説明を調べて、`wavread` で MATLAB に読み込んだ音声データを、`sound` 関数を利用して正しく出力せよ。

練習 13 `wavread` で読み込んだデータをプロットせよ。ただし、横軸の単位が時間 (秒) になるようにプロットせよ。

練習 14 `wavread` で読み込んだデータをプロットせよ。ただし、開始時刻から 0.3 秒のところから 125ms 間だけ取り出してプロットすること。横軸は 0.3 秒から開始して、単位が時間になるようにすること。

`wavread` で読み込んだデータは、**ソースコード 1** と同様に `sound` で出力できる。しかし、少し異なる部分がある。

読み込んだデータの最初の部分を表示させてみる。

```
>> y(1:10)

ans =

    0.0032
    0.0012
    0.0010
   -0.0006
   -0.0002
    0.0023
    0.0007
   -0.0003
    0.0010
    0.0018

>>
```

実際の値は各自異なるので無視してほしい。ここで注目すべきは、データが縦方向にのびていることである。**ソースコード 1** のように : を用いて数列を作成すると行ベクトルになるが、`wavread` で読み込んだデータは列ベクトルになる。

ある (1 次元) データが行ベクトルか列ベクトルかを確認するのには、`size` 関数が便利である。

```
>> size(y)

ans =

        7001           1

>> size(t)

ans =

           1        8001

>>
```

size 関数は、データのサイズを返却する関数である。help コマンドで確認すればわかるが、1 つ目の返り値は列方向の長さ (行数) で、2 つ目の返り値は行方向の長さ (列数) となる。

録音した音に対して、自分で生成した音を重ねたり、振幅変調をかけられる。しかし、+ は行列の足し算であり、.* は要素ごとのかけ算なので、size が同じでないと演算が実行できない。

長さが異なる場合には、短かい方に合わせて部分を取り出すのが簡単である。ベクトルの方向が違う場合 (行ベクトルと列ベクトルの場合) は、例えば行ベクトルを列ベクトルに変換しなければならない。

練習 15 自分で録音した音に、適当な正弦波を重ねてみよ。(行ベクトルを列ベクトルに変換するためには、線形代数での行列の基本操作である転置が利用できる。) エラーなく重ねることができたら、プロットして意図通り処理できているかを確認せよ。プロットに問題なければ、正規化して sound 関数で出力するか、soundsc 関数で出力してどのような音になったか確認せよ。

練習 16 自分で録音した音を、適当な正弦波で振幅変調してみよ。結果をプロットせよ。また、プロットして問題がなければ、出力してどのような音になったか確認せよ。

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 9/25(土) 午後 8 時とします。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。音を生成・加工する課題については、音データもパワーポイントのスライドに貼り付けるようにすること。

1. **練習 1** を解答せよ。
2. **練習 6** を解答せよ。(途中の式もパワーポイントにきちんと書くこと)
3. **練習 7** を解答せよ。
4. **練習 10** を解答せよ。
5. **練習 14** を解答せよ。
6. **練習 16** を解答せよ。
7. 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。(これに関しては、スライドではなく、別のファイルに作成してアップロードせよ)