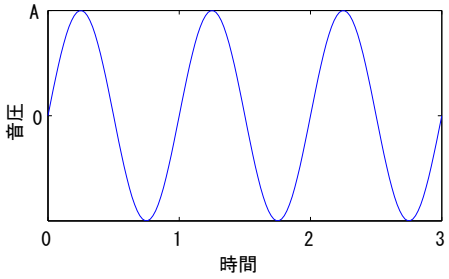
1　波形データの生成

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。物理の文献を見ると、純音は次のような式で表されることがわかる。

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

ここで、y は音圧、A は振幅、f は周波数、t は時間である。時間を横軸にグラフを描くと次のようになる。

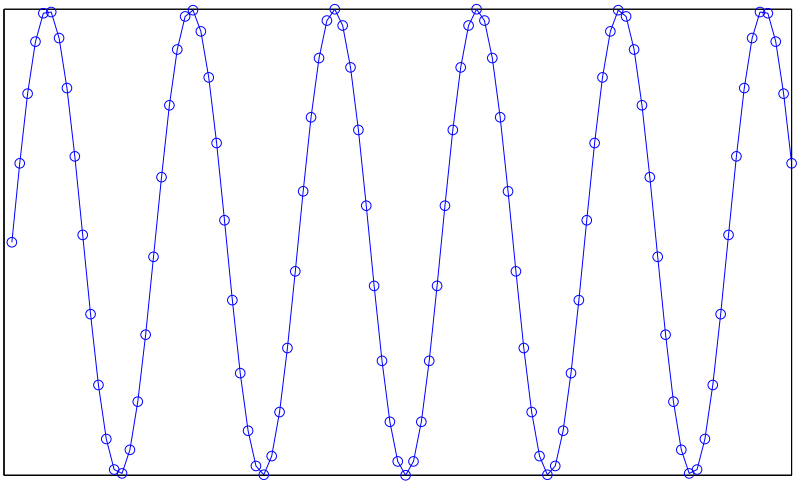


グラフは 2 次元であるが、y という一つの値が時間によって変化するので 1 次元データとよぶ。

この式に基づいて、MATLAB で音のディジタルデータを生成するプログラムを作成する。データを生成するためには、まず、式 (1) の変数の値を決めなければならない。A, f は、それぞれ 1 つの値を定めればよい。例えば、$A = 1$、$f = 50$ などである。このように大きさのみを持つ量のことをスカラーとよぶ。

一方、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。つまり一つの値ではない。

物理的な世界では、t は連続的に変化する。このような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、1/8000 秒の間隔とすると、$0 \leq t \leq 1$ という範囲は、$0, 1/8000, 2/8000, 3/8000, \ldots, 7999/8000, 1$ という数の列で表される。この 1/8000 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。このようにとびとびの値で表すことを離散的であるという。つまり、コンピュータの中では正弦波は、次のグラフの丸印のところだけで表現されている。



このようなデータをプログラミングするときには、t をひとまとまりで扱うと便利である。そのような場合、数学的にはベクトルとして扱うことがある。

MATLAB で 440Hz の音を 1 秒間生成してみる。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、MATLAB が表示するプロンプトである。したがって試すときには、両方とも入力しなくてよい)

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);
```

このように、MATLAB はベクトルを用いた計算を非常に簡単に行え、ほとんど数式と同じ形でプログラミングできる。プログラムを説明する。1 行目では、時間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。

この : の典型的な使い方を示す。

```
>> 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ことに注意)。

ソースコード 1 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。間隔を指定した : の例を示す。

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

m =

   -1.0000   -0.5000         0    0.5000    1.0000    1.5000    2.0000
```

指定された通りに 0.5 刻み間隔の数列が生成される。

行の最後の ; は、値の表示を抑制する。例を示す。

```
1 >> a=1
2
3 a =
4
5      1
6
7 >> b=2;
8 >> c=a+b;
9 >>
```

1 行目のように、代入など計算を行うと値が表示される。しかし、 7 行目のように ; をつけると何も表示されない。8 行目のように計算したときも、その式に ; をつけると結果は表示されない。

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。数式とほとんど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 の説明では関数名が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せよ。この説明では、引数に関して 「X の要素の」と書かれている。**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。引数に数列をとった場合の例を示す。

```
>> 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
```

3 行目の ans は、直前の計算結果が自動的に代入される変数である。この例のように、sin は、数列を引数に与えると、計算結果は引数の数列の sin の値を持つ数列となる。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。このことで、MATLAB は数式とほぼ同じ形でプログラミングできるのである。

ソースコード 1 の 4 行目では、t は数列である。t にかけられている 2*pi*f はスカラーである。したがって、2*pi*f*t は、数列 t の各々の要素をそれぞれ $2\pi f$ 倍する。全体が $2\pi f$ 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。このように MATLAB では数列をベクトルとして扱う。

作成した音データ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。

2　1 次元データの可視化

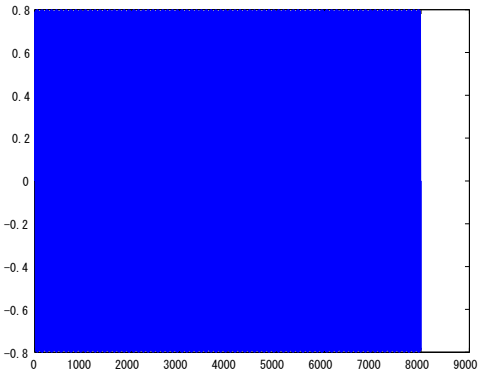
音を生成したり、加工したり、録音した場合には、もちろん、音を再生・出力して確認すべきである。しかし、音は聞こえ方が人によってかなり異なるし、プログラムに失敗していたら、聞こえる音にならなかったり、デバイスに悪影響を与えるようなデータになることもある。したがって、聞く以外の方法でも確認した方がよい。普通には見ることができないデータを見えるようにすることを可視化という。

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。)

ソースコード 2: 1 引数の `plot`

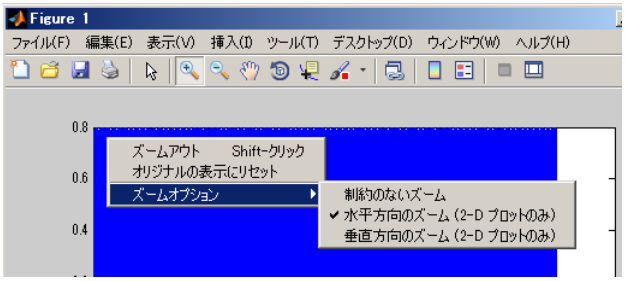
```
>> plot(y)
```

次のようなグラフが表示される。

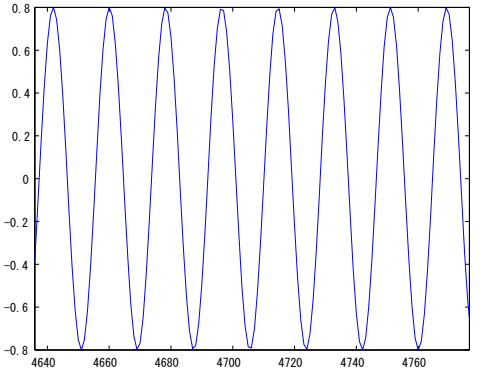


これではほとんどよくわからないだろう。

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、次のグラフのように、典型的な sin 波のグラフを見られるだろう。



これらのグラフでは、縦軸 (y 軸) は、音圧 (変位ともいう) をあらわしている。しかし、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。

ソースコード 3: 2 引数の plot

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

このようにすると、横軸は時刻になる。

グラフの詳細を観察するために、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。

ソースコード 4: インデクスによるベクトルの要素の参照

```
>> 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。

インデクスは数列でも指定できる。

```
>> 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 MATLAB では複数の値を同時に指定できるので、y の一部を簡単に取り出せる。この機能を使うと、次のように簡単に 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))
```

このように t と y に関して、同じインデクスの部分を取り出せば対応する時刻で表示される。

3　時間波形の重ね合わせ

音は、和音を出したり、複数の楽器で合奏したり、セリフの背景に音楽 (BGM) を流したりするように、同時に複数の音を重ねることができる。複数の音を同時に鳴らすのは、時間波形の足し算で実現できる。

ソースコード 5: 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。

ソースコード 6: 時間波形の重ね合わせ—うなり

```
>> a=0.4;
>> 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)

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

この式は次のように変形できる。

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

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。正しく計算できていれば、同じ結果が得られる。

`.*` という演算子がどのような計算をおこなうかは、`help` 関数で確認できる。

```
>> help .*
```

このように入力して表示された結果を、よく探すと次のような説明が表示される。

```
 .*   配列乗算

 X.*Y は、要素毎の積です。X と Y は、どちらかがスカラでなければ、同じ
 大きさでなければなりません。スカラは、配列要素すべてに掛けることが
 できます。
```

要素ごとの積とはどのような計算かは、次のようにすれば確認できる。

```
>> x=[1 2 3];
>> y=[4 5 6];
>> x.*y

ans =

     4    10    18
```

x.*y の結果は、x(1)*y(1) のように対応する値をかけた値になっている。

式 (3) の $a_{yy}\cos(2\pi f_b t)$ の部分を式 (1) の A の部分に対応すると考えると、振幅を時間の関数によって変化させている、ととらえられる。このように、信号の振幅を時間の関数で変化させることを振幅変調と呼ぶ。

音を重ね合わせて生成された音は、元のそれぞれの音の振幅より大きくなることがある。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。

ソースコード 7: 振幅の大きな音の再生

```
>> 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 では、自動的にこのような処理をして出力する関数が用意されている。soundsc である。soundsc を使うと音は歪まない。

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
```

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。

ソースコード 5 の y523 と y660 を連結して再生するには、次のようにすればよい。

```
>> soundsc([y523 y660], fs)
```

5　読み込んだ音声データの加工

まず、音声ファイルを MATLAB に読み込む方法を説明する。MATLAB を起動して、**カレントディレクトリ** のところに、音声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[1]。**コマンドウィンドウ** に次のようにタイプする。

ソースコード 8: wavread 関数による音声ファイルの読み込み

```
>> [y, fs, nbits] = wavread('vibra8.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。

wavread で読み込んだデータは、**ソースコード 1** で作成したデータと同様に 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
   -0.0000
         0
   -0.0001
    0.0001
   -0.0006
   -0.0016
   -0.0017
   -0.0016
   -0.0014
```

ソースコード 1 で作成したデータとは違って、データが縦方向にのびていることがわかる。つまり、**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ことがわかる。

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 =

       26000           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。+ や .* を利用すればよい。

前述の vibra.wav のデータに正弦波を足してみる。

[1] 音声ファイルは授業支援システムからダウンロードしておくこと

```
>> [y,fs,nbits]=wavread('vibra8.wav');
>> t=0:1/fs:1;
>> f=440;
>> a=0.1;
>> ysin=a*sin(2*pi*f*t);
>> ymix=y+ysin
??? エラー ==> plus
行列の次元は一致しなければなりません。
```

上述のように、ファイルから読み込んだデータと : を元に作成したデータでは、ベクトルの方向が違うのでエラーが出てしまう。したがって、どちらかを変換してやらなければならない。

ここでは、行ベクトルの ysin を列ベクトルに変換することで、方向をあわせてみる。行ベクトルと列ベクトルの相互変換は行列演算の転置である。転置を行う関数があれば、簡単に変換できることになる。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 転置
ctranspose                      -- '    複素共役転置
transpose                       -- .'   転置。
```

この場合は、どちらでも同じ結果になるので、’ を使ってみる。

```
>> ysin_t=ysin';
>> size(ysin_t)

ans =

        8001           1

>> size(ysin)

ans =

           1        8001
```

転置されて、行ベクトルが列ベクトルになったことがわかる。

そこで、ysin_t を y に足してみる。

```
>> ymix=y+ysin_t
??? エラー ==> plus
行列の次元は一致しなければなりません。
>> size(y)

ans =

       26000           1

>> size(ysin_t)

ans =

        8001           1
```

方向は同じでも、size が異なると+で足すことはできない。size が異なる場合には、短かい方に合わせて長い方の一部を取り出すのが簡単である。

ソースコード 9: 録音したデータと生成したデータの重ね合わせ

```
>> ymix=y(7000:15000)+ysin_t;
>> soundsc(ymix,fs)
```

無事、音を重ねることができた。二つの音が重なって鳴っていることが確認できるはずである。