特別研究 I: MATLAB 演習（2005 年度版）

担当：北原 鉄朗*

本演習では，プログラム言語 MATLAB の使い方を学んだのち，それを用いた音響信号処理の基礎の基礎についても学ぶ．ここでの目標は以下の通りである．

1. MATLAB で一通りのプログラムを書けるようになる．
2. MATLAB で簡単な聴覚心理学の実験のためのデータを作れるようになる．
3. MATLAB で簡単な音響信号処理を行えるようになる．ここでは，基本周波数推定を取り上げる．

なお，本演習に関する情報は，http://winnie.kuis.kyoto-u.ac.jp/~kitahara/local/matlab/（アクセス制限あり）に掲載する．内容の変更等の通知はここへの掲示をもって行うので，こまめに確認すること．

1 MATLAB とは

1.1 MATLAB の特徴

MATLAB は，プログラム言語の 1 つであり，次の特徴を持つ．

- プリミティブなデータ型が行列である．
- 行列計算や FFT などの最適化された（組み込み関数による）計算は高速である．
- インタプリタ言語のため，上の処理以外は遅い．
- 信号処理，統計解析，シミュレーションなどのための関数が豊富に用意されている．

1.2 MATLAB の利用

MATLAB のインストール

本演習では，情報学研究科が提供しているライセンスを利用して，各受講生が使用するノート PC（Windows）にインストールすることを想定している．詳しくは上記 URL を参照のこと．

MATLAB の起動方法（Windows の場合）

通常のアプリケーションを起動するのと同様に，スタートメニューからアイコンを選ぶか，デスクトップ上にアイコンがある場合には，それをダブルクリックすればよい．

*奥乃研究室，kitahara@kuis.kyoto-u.ac.jp

MATLAB の起動方法（UNIX の場合）

コマンドラインから

```
%matlab
```

とすると，MATLAB の統合環境が立ち上がる．少しでも軽くしたいときは，

```
%matlab -nodesktop
```

とすればよい（ターミナル内で MATLAB が立ち上がる）．

MATLAB 起動後は

MATLAB を起動すると，

```
>>
```

というプロンプトが表示されるので，ここにコマンドを入力する．ちなみに，終了するコマンドは，`quit` である．

2 MATLABプログラミング

2.1 MATLABプログラミングの最初の一歩

四則演算など

```
>> 5 + 3

ans =

     8

>> 5 - 3

ans =

     2

>> 5 * 3

ans =

    15

>> 5 / 3

ans =

    1.6667

>> 5 \ 3

ans =

    0.6

>> 5 ^ 3

ans =

   125

>> mod(5, 3)

ans =

     2
```

複素数

```
>> 1 - 3 * i

ans =

   1.0000 - 3.0000i
```

変数

```
>> a = 3;
>> b = 2;
>> a + b

ans =

     5
```

MATLAB では，変数を宣言せずに使うことができる．
MATLAB では，文末にセミコロンをつけると計算結果の表示が抑制される．

ワークスペース

ワークスペースとは，MATLAB 実行中にメモリに格納されている変数群である．whos コマンドで現在のワークグループを確認することができる．

```
>> whos
  Name        Size        Bytes  Class
  a           1x1             8  double array
  b           1x1             8  double array
  ans         1x1             8  double array
```

ベクトル

```
>> x = [1  2  3]

x =

     1     2     3

>> x = [1, 2, 3]

x =

     1     2     3

>> x = [1; 2; 3]

x =

     1
     2
     3

>> x = [1  2  3]'

x =

     1
     2
     3
```

アポストロフィは，転置を表す.

コロンを使ったベクトル表現

```
>> x = 1 : 10

x =

     1     2     3     4     5     6     7     8     9    10

>> x = (1 : 10)'

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

>> x = 1 : 2 : 10

x =
     1     3     5     7     9

>> x = (1 : 5) * 2

x =
     2     4     6     8    10
```

行列

```
>> A = [1  2  3; 4  5  6]

A =

     1     2     3
     4     5     6

>> A = [1  2  3; 4  5  6]'

A =
     1     4
     2     5
     3     6
```

行列同士の演算

```
>> A = [1 2; 3 4];
>> B = [5 6; 7 8];
>> A + B

ans =

     6     8
    10    12

>> A * B

ans =

    19    22
    43    50

>> A .* B

ans =

     5    12
    21    32
```

MATLAB では， * / \ ^ で通常の行列演算を表すのに対し， .* ./ .\ .^ で要素同士の演算を表す. 加減算は通常の行列演算も要素同士の演算も等価なため，.+ .- はない.

例題 1 次の連立方程式を解くことを考える．

$$\begin{cases} x + 4y = 1 \\ 2x + 3y = -1 \end{cases}$$

行列で表すと，

$$\begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\therefore \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

```
>> [1  4; 2  3] \  [1;  -1]
ans =

   -1.4000
    0.6000
```

演習 1 自分で連立方程式を考え，MATLAB を使って解け．

行列計算に便利な関数

行列計算に便利な関数をいくつか列挙する．詳しくは後述のヘルプを参照のこと．

- `zeros`(m，n)：要素がすべて 0 である $m \times n$ 型行列を作成する．
- `ones`(m，n)：要素がすべて 1 である $m \times n$ 型行列を作成する．
- `repmat`(A，m，n)：行列 A を縦方向に m 回，横方向に n 回繰り返した行列を作成する．
- `find`(A)：行列 A の非ゼロ要素のインデックスを出力する．通常は `find(X>100)` のように用いる．
- `sum`(A)：行列 A に対して，列毎に合計値を求める．
- `mean`(A)：行列 A に対して，列毎に平均値を求める．
- `std`(A)：行列 A に対して，列毎に標準偏差を求める．

m ファイル

MATLAB では，以下のような内容のテキストファイルをつくり，拡張子に m をつけて保存することで，新たな関数として実行できる．

（ファイル名：ave.m）

```
function x = ave(a, b, c)

x = (a + b + c) / 3;
```

```
>> ave(6, 7, 8)

ans =
     7
```

if 文・for 文・while 文

```
if a > 0
  command1;
elseif a == 0
  command2;
else
  command3;
end
```

```
for i = 1 : 10
  x = x + i * i;
  fprintf(1, '%d\n', x);
end
```

```
while result > 0
  result = command(a, b, c);
end
```

論理和には&，論理積には|，否定には~を用いる．また，ベクトル内の全要素に対して論理和，論理積を求める any，all が用意されている．

2.2 グラフィックス

関数をグラフにする

```
>> x = 0 : pi / 100 : 2 * pi;
>> y = sin(x);
>> plot(x, y);
```

関数をグラフにする 2

```
>> x = 0 : pi / 10 : 2 * pi;
>> y = sin(x);
>> plot(x, y, 'o');
```

違うウィンドウに描画する

```
>> figure(2)
```

とすると，指定した番号（ここでは 2）を ID とするグラフィックス用ウィンドウが立ち上がり，次の描画命令以降は，そのウィンドウに描画する．

すでに描画したグラフに重ねて描画する

通常，描画コマンドを実行すると，現在描画されているグラフは消されるが，

```
>> hold on
```

とすると，消さずに上から描画する．

例：

```
>> figure 1
>> x = 0 : pi / 100 : 2 * pi;
>> y = sin(x);
>> plot(x, y);
>> hold on
>> x = 0 : pi / 10 : 2 * pi;
>> y = sin(x);
>> plot(x, y);
>> hold off
```

2.3 MATLAB らしいプログラミング

前述の通り，MATLAB は，行列計算は速いが，for 文などは遅い．すなわち，他の言語では for 文で書くような処理を，以下に行列計算に落とし込むかが，高速プログラミングのみそとなる．

次の 2 つのプログラムの実行時間を比較しよう

```
function S = test1(n)

S = 0;
for i = 1 : n
  S = S + i * i;
end
```

```
function S = test2(n)

a = 1 : n;
S = sum(a .* a);
```

実行時間の計測には，tic コマンドと toc コマンドを使用する．

```
>> tic; test1(1000), toc
>> tic; test2(1000), toc
```

演習 2 次のプログラムと等価で高速なプログラムを考えよ．

```
for i = 1 : m
  for j = 1 : n
    A(i, j) = i * j;
  end
end
```

2.4 文字列操作

MATLAB における文字列の扱い

MATLAB では，文字列は char 型の $1 \times n$ 型行列（横ベクトル）として扱われる．シングルクォーテーション（ダブルクォーテーションではないことに注意）で囲むと，文字列として扱われる．

```
>> S = 'hello';
>> whos
  Name        Size        Bytes  Class
  S           1x5            10  char array
```

文字列処理に用いる関数

文字列処理でよく用いられる関数を列挙する．詳しくはヘルプを参照のこと．

- `fprintf`(id，S)：ファイル ID が id のファイルに文字列 S を書き込む．id を 1 とすると標準出力となる．C 言語と同じ．
- `ischar`(S)：変数 S が文字列かどうかを調べる．
- `isempty`(S)：変数 S が空（0×0 型）かどうかを調べる．
- `strcmp`(S_1，S_2)：2 つの文字列 S_1, S_2が等しいかどうかを調べる．
- `strfind`(S，p)：文字列 S においてパターン p がはじめに出力するインデックスを求める．
- `num2str`(x)：変数 x を文字列に変換する．
- `int2str`(x)：変数 x を整数とみなして文字列に変換する．
- `str2num`(x)：文字列として表された数字 x を数値に変換する．

文字列の結合

文字列の正体は横ベクトルなので，通常のベクトルの結合と同じようにすればよい．

```
>> S1 = 'Hello';
>> S2 = 'World';
>> S = [S1  S2]
S =
HelloWorld

>> S = [S1  ' ' S2]
S =
Hello World
```

テキストファイルの読み込み

ファイルオープンには fopen，1 行の読み込みには fgetl または fgets を用いる．

例題 2 テキストファイルの中身を表示する関数を作成してみよう．

（ファイル名：mytype.m）

```
function mytype(filename)

fid = fopen(filename, 'r');
while 1
  line = fgetl(fid);
  if ~ischar(line)
    break;
  end
  fprintf(1, [line  '\n']);
end
```

演習 3 テキストファイルの行数と文字数をカウントする関数を作成せよ．

トークンの取り出し

トークンとは文字列処理の最小単位を表し，ここでは空白で区切られるものとする．トークンは，以下の strtok 関数で取り出すことができる．

```
>> S = 'I have a pen';
>> [token, rest] = strtok(S)
token =
I

rest =
 have a pen

>> [token, rest2] = strtok(rest)
token =
have

rest =
 a pen
```

例題 3 以下のように (x, y, z) の 3 次元座標を列挙したファイルがあるとする．これを読み込む関数を作成しよう．

```
0.0000   0.0000   0.0000
-1.2300  -3.2222  1.3376
3.2443   2.4433   5.0004
```

（ファイル名：readxyz.m）

```
function [x, y, z] = readxyz(filename)

fid = fopen(filename, 'r');
i = 1;
while 1
  line = fgetl(fid);
  if ~ischar(line)
    break;
  end
  [token, line] = strtok(line);
  x(i) = str2num(token);
  [token, line] = strtok(line);
  y(i) = str2num(token);
  [token, line] = strtok(line);
  z(i) = str2num(token);
  i = i + 1;
end
```

演習 4 上記の数値列を 1 つの行列とみなし，1 つの行列に読み込む関数を作成せよ．その際，任意のサイズの行列に対応できるようにせよ．

演習 5 上の例題で示したプログラムはエラー処理をしていないので，不正なファイル（たとえば数値に変換できない文字列が含まれているなど）が読み込まれたときのエラーメッセージがわかりにくくなっている．エラー処理の記述方法を自分で調べ，適切なエラーメッセージが表示されるよう改良せよ．

演習 6 テキストファイルの行数と文字数を求める関数に，単語数（トークン数）を求める機能を追加せよ．

2.5 GUIの作成

MATLABでは，簡単にGUIを作成するためのツールとして「GUIビルダー」というものが用意されているが，ここでは，それを使わないGUI作成法を紹介する．

ウインドウの作成

ウインドウの作成には，figure関数を使用する．

```
>> figure('Name', 'GUITest', 'Position', [100 100 640 480], ...
       'Tag', 'GUITest', 'Risize', 'off');
```

figure関数の書式は，

figure(*PropertyName1*，*PropertyValue1*，*PropertyName2*，*PropertyName2*，...)

となっており，引数でプロパティを設定する．以下に，主なプロパティのみ列挙する．

名前	取りうる値	説明
Name	文字列	ウインドウのタイトルバーに表示される文字列
Position	[int int int int]	ウインドウの位置とサイズ
q		（左下のx座標，左下のy座標，横の長さ，縦の長さ）
		※座標系は，画面の左下が原点であることに注意
Tag	文字列	プログラム上でGUI部品を特定するための識別子
Resize	'on' / 'off'	ウインドウのサイズ変更を許可するか否か

GUI部品の配置

GUI部品の配置には，uicontrol関数を用いる．書式は，figureと同じく，

uicontrol(*PropertyName1*，*PropertyValue1*，*PropertyName2*，*PropertyName2*，...)

となっており，以下のプロパティを利用できる（主なもののみ）．

名前	取りうる値	説明
Style	文字列	GUI部品の種類を以下の文字列で指定する．
		pushbutton, togglebutton, radiobutton, checkbox,
		edit, text, slider, frame, listbox, popupmenu
Position	[int int int int]	figure関数と同じ
Tag	文字列	figure関数と同じ
String	文字列	ボタンの上などに表示される文字列
Value	GUI部品の種類に依存	GUI部品の初期値
Callback	MATLABステートメント	ボタンが押されたりしたときのアクションを記述

上記のものの他に，FontName，FontSizeなどさまざまなものが用意されている．詳しくは，MATLABのオンラインヘルプなどを参照されたい．

例題 4 1つのスライダとエディトボックスからなり，スライドを動かしたら，スライダの値が表示されるGUIプログラムをつくりたい．まず，画面のデザインの部分のプログラムを書いてみよう．

（ファイル名：GUITest.m）

```
function GUITest

fig = figure('Name', 'GUITest', 'Position', [100 100 300 100], ...
   'Tag', 'GUITest', 'Resize', 'off');

slider = uicontrol('Style', 'slider', 'Position', [30 30 170 30], ...
   'Tag', 'slider1', 'Callback', 'update');

edt = uicontrol('Style', 'edit', 'Position', [220 30 50 30], ...
   'Tag', 'edt1');
```

このプログラムを実行すると，ウインドウが表示され，その上にスライダとエディトボックスが配置される．スライダが操作されると，update.m ファイルを探しにいき，update 関数を実行する．update 関数の設計は次節にて行う．

演習 7 上記の例題を参考に，自分でさまざまな GUI を設計せよ．必要に応じてオンラインヘルプなどを参照すること．

イベントアクションの記述

ボタンが押されるなどの，GUI 部品に対するなんらかの操作が発生することを**イベント**といい，イベントが発生したときに何らかの処理を引き起こすことを**コールバック**という．ここでは，コールバックの記述法について述べる．

コールバックの基本は，GUI 部品の Callback プロパティにコールバックの内容を記述することである．通常は，コールバックの内容を 1 つの関数として記述し，Callback プロパティにはその関数名を書く．たとえば，例題 2 では，スライダの Callback プロパティが update となっているので，update という名の関数をつくり，そこにスライダが操作されたときの処理を書けばよい．

（ファイル名：update.m）

```
function update

% ここにスライダが動かされたときの動作を記述
```

GUI プログラムでは，GUI 部品のプロパティの値を取得・設定しなければならないことが多い．これには，次の 3 つの関数を使用する．

- findobj

  書式：*objhandle* = findobj('Tag', *TagValue*)

  Tag プロパティが *TagValue* である GUI オブジェクトの**ハンドル**を *objhandle* に代入する．ハンドルは，MATLAB 内部で使用するオブジェクトの識別番号のようなもので，次の set，get で使用する．

- set

  書式：set(*objhandle*，*PropertyName*，*PropertyValue*)

  ハンドルが *objhandle* である GUI オブジェクトのプロパティを設定する．

- get

  書式：x = get(*objhandle*，*PropertyName*)

  ハンドルが *objhandle* である GUI オブジェクトのプロパティを取得する．

例題 5 例題 2 の GUI プログラムを完成させよう．

（ファイル名：update.m）

```
function update

slider = findobj('Tag', 'slider1');
edt = findobj('Tag', 'edt1');
newvalue = get(slider, 'Value');
set(edt1, 'String', num2str(newvalue));
```

演習 8 演習 3 で作成した GUI にコールバックを記述し，簡単な対話型プログラムを完成させよ．

2.6 ヘルプ

```
>> help fft

 FFT Discrete Fourier transform.
    FFT(X) is the discrete Fourier transform (DFT) of vector X.  For
    matrices, the FFT operation is applied to each column. For N-D
    arrays, the FFT operation operates on the first non-singleton
    dimension.
                  ......
```

```
>> helpdesk
```

3　音響信号処理の基礎の基礎

3.1　簡単な音響信号の合成

正弦波

定常的な信号は，基本的に正弦波の合成によりつくられる．基本周波数を f，振幅を A とすると，次の式で表される：

$$y = A\sin(2\pi f t).$$

例題 6 440Hz（中央のラ）の音を正弦波でつくってみよう．

```
>> t = 0 : 1/44100 : 5;
>> y = 0.9 * sin(2 * pi * 440 * t);
>> wavwrite(y, 44100, 16, 'test1.wav');
```

このような正弦波のみによりつくられた音は，**純音**と呼ばれる．

演習 9 262Hz（中央のド）の音を正弦波でつくってみよう．

正弦波の足し合わせによる複合音の合成

基本周波数の整数倍の正弦波を足し合わせると，1 つの音に聞こえる性質がある．

例題 7 440Hz の音に 2 次倍音，3 次倍音を加えてみよう．

```
>> t = 0 : 1/44100 : 5;
>> y = 0.4 * sin(2 * pi * 440 * t) + 0.3 * sin(2 * pi * 880 * t) + ...
       0.2 * sin(2 * pi * 1320 * t);
>> wavwrite(y, 44100, 16, 'test2.wav');
```

演習 10 任意の基本周波数を持つ音に，10 次までの各倍音を任意の振幅で合成するプログラムを書き，次のことを試せ．

1. 基音に対して高次の倍音の振幅が大きいと音はどのように聞こえるか．逆に小さいとどうか．
2. 偶数次倍音の振幅を 0 にすると，どのような音に聞こえるか．

演習 11 上の演習で作成した，10 次までの各倍音を任意の振幅で合成するプログラムを，以下のような GUI プログラムに改造せよ．

- スライダで各倍音の振幅を設定．
- [作成] ボタンで音響信号を作成．
- [再生] ボタンで再生．

演習 12 上の演習でつくったプログラムを使って，フルート，オーボエ，クラリネットの各楽器に近い音をつくるには，各音の振幅をどのように設定すればよいか，試行錯誤せよ．

周波数変調による音の合成（FM 音源）

FM 音源の簡単な原理は，正弦波を作り出す発信器に渡す周波数を変調させることである．この正弦波を作り出す発信器を**キャリア発信器**といい，キャリア発信器の周波数を変調させる発信器を**モジュレータ発信器**という．式で表すと次のようになる：

$$y = A\sin(2\pi f_C t + I\sin(2\pi f_M t)).$$

ここで，A はキャリアの振幅，f_C, f_Mはそれぞれキャリア，モジュレータの周波数である．I は変調指数と呼ばれる．このように，sin 関数を合成（加算ではなく）することで，sin 関数の少ない計算で豊富な倍音を発生させることができる（上の加算合成では 10 個の倍音を発生させるには 10 回 sin 関数を計算しなければならない）．なお，市販されている FM 音源のシンセサイザーでは，キャリア，モジュレータをいくつも用意し，自由に組み合わせることで，より多様な音の合成を実現している．

例題 8 FM 音源の原理を使って音を合成してみよう．

```
>> t = 0 : 1/44100 : 5;
>> y = 0.9 * sin(2 * pi * 220 * t + sin(2 * pi * 880 * t));
>> wavwrite(y, 44100, 16, 'test3.wav');
```

演習 13 キャリア周波数，モジュレータ周波数，変調指数をいろいろ変え，音がどのように変化するか確かめよ．また，フルート，オーボエ，クラリネットの各楽器に近い音をつくるには，パラメータをどのように設定すればよいか，試行錯誤せよ．

白色雑音

白色雑音 (ホワイトノイズ，white noise) とは，すべての周波数にわたって一様のパワー密度を持つ雑音である．MATLAB では，乱数発生関数（`randn`）で簡単に作成することができる．

```
>> fs = 44100;
>> y = randn(1 * fs, 1);
>> wavplay(y, fs);
```

演習 14 白色雑音は周期性がないため，再生速度を変化させても感じられる音の高さ（ピッチ）に変化が見られない．このことを確認せよ．

音量（振幅）[1] を変化させる

振幅を変化させるには，振幅の時間変化を表すベクトル（包絡あるいはエンベロープという）を用意し，元の音響信号と要素毎の掛け算をすればよい．

例題 9 最初の 0.02 秒間で最大音量まで音量が大きくなり，その後徐々に音量が小さくなる音響信号を作成しよう．

```
>> t = 0 : 1/44100 : 2;
>> A1 = linspace(0, 0.99, 0.02 / 44100);
>> A2 = linspace(0.99, 0, length(t) - 0.01 / 44100);
>> A = [A1  A2];
>> y = A .* sin(2 * pi * 440 * t);
>> wavwrite(y, 44100, 16, 'test4.wav');
```

注 `f * t` ではなく `f .* t` であることに注意．
発音開始から最大音量に達するまでの時間は**立ち上がり時間**などと呼ばれる．

演習 15 振幅の立ち上がり時間やその後の減衰率などをさまざまに変えた音響信号を作り，音がどう変化するか確めよ．

周波数を変化させる

```
>> t = 0 : 1/44100 : 1;
>> f = linspace(440, 880, length(t));
>> y = 0.9 * sin(2 * pi * f .* t);
```

注 `f * t` ではなく `f .* t` であることに注意．

演習 16 以上の技を駆使してさまざまな音をつくってみよう．

演習 17 現実の音には，振幅や周波数自体が周期的に変化するものが少なくない．このような周期的な振幅・周波数の変化は，振幅変調（amplitude modulation, AM），周波数変調（frequency modulation, FM）と呼ばれる（それぞれ音楽用語のトレモロ，ビブラートに対応する）．このような変化を持つ音を作るにはどのようにすればよいか考えよ．

[1] 音量（ラウドネスともいう）は人が感じる音の大きさ（心理量）であり，振幅やパワーは信号から定まる物理量である．これらは多くの場合密接に関連するが，本来厳密に区別されるべき概念である．

3.2 MATLABで作成した音響信号を使った簡単な聴覚心理学実験

複合音の分離を引き起こす周波数のずれについて

2つの正弦波が整数倍の関係にある周波数を持つとき，それらが融合して1つに聞こえることは前述の通りである．では，周波数の関係がどの程度整数倍からずれると2つの音に分離して聞こえるのだろうか．これについて聴取実験を行おう．

演習 18 演習11で作成したGUIプログラムを参考にして，スライダにより簡単に周波数のずれの度合いを設定して再生するGUIプログラムを作成せよ．この際，基本周波数や音の長さも自由に設定できるようにするのが望ましい．

演習 19 上のプログラムを用いて，どの程度周波数がずれると2つの音に分離して聞こえるのか実験せよ．また，さまざまな基本周波数や音の長さで試し，結果を比較せよ．

演習 20 他の学生に同様の実験をしてもらい，自分の場合と結果を比較せよ．

複合音の分離を引き起こす立ち上がりのずれについて

ここでも前項と同様に整数倍の周波数を持つ2つの正弦波について扱う．整数倍の周波数を持つ2つの正弦波が2つの音に分離して聞こえる要因としては，周波数のずれだけでなく，各正弦波の振幅や周波数が異なって（別々に，同期せずに）変化することも挙げられる．ここでは最も簡単な立ち上がりについて検討しよう．

演習 21 整数倍の周波数を持つ2つの正弦波を加算合成するGUIプログラムを作成せよ．このプログラムでは，2つの正弦波のそれぞれに対して個々に発音開始時刻や立ち上がり時間を（スライダで）設定できるようにすること．

演習 22 上のプログラムを用いて，どの程度発音開始時刻や立ち上がり時間がずれていると2つの音に分離して聞こえるのか実験せよ．

演習 23 上の実験を他の学生にもしてもらい，自分の場合と結果を比較せよ．

演習 24 周波数のずれと立ち上がりのずれが同時に存在する場合，上の実験結果はどのように変化するか確めよ．また，周波数のずれと立ち上がりのずれのどちらがより2音の融合／分離に大きな影響を与えるか確めよ．

演習 25 振幅や周波数の時間変化は立ち上がりだけでなく，振幅変調（AM）や周波数変調（FM）などもある．これらについても，上と同様に2つの正弦波のそれぞれに対して個々に設定できるGUIプログラムを作成し，2音の融合／分離にどの程度影響を与えるか確めよ．

ミッシングファンダメンタル現象について

演習 26 人間の音の知覚において，実際に基音が存在しなくても基音に相当するピッチを知覚することが知られている．たとえば，440Hz，660Hz，880Hz，…という周波数成分を持つ音は，220Hzの成分がなくても220Hzの音としてきこえることがある．これを**ミッシングファンダメンタル現象**という．演習11で作成したプログラムを使っていろいろな音を作り出し，この現象が実際に起こるか確かめよ．

音のパターンの知覚について

上の実験では，同時になった音が分離して聞こえるかどうかを扱ったが，本節では，順番になった複数の音の列が，どのようにパターンとして体制化されるかどうかについて検討する．

まず，実験の準備として，以下のようなテキストファイルを元に音のパターンを作成するプログラムについて考えよう．

```
100  50 4800 100
200  50 6000 100
300  50 4800 100
400  50 6000 100
```

このファイルは，各行が 1 つの音を表し，それぞれの数値は，左から発音時刻，音長，周波数，振幅を表す．発音時刻と音長は 10ms 単位（すなわち 100 で 1 秒），振幅は 1/100 単位とする．また，周波数はセント（cent）と呼ばれる対数スケールの単位で表す．セントは本来，何らかの基準周波数からの周波数比を表すものであるが，基準周波数を定めることで周波数を表す単位としても用いられる．ここでは，次式を用いる：

$$f[\text{cent}] = 1200 \log_2 \frac{f[\text{Hz}]}{440 \times 2^{3/12-5}}.$$

演習 27 上記の形式のファイルを読み込み，それに基づいて音のパターンを作成するプログラムを作成せよ．ファイルの読み込み部は**例題 3** を参考にすることができる．用いる音は純音でよい．また，必要に応じて最大の長さを制限するなどの制約を設けても構わない．

演習 28 上のプログラムを用いて次のファイルの音のパターンを作成せよ．

（ファイル名：sound1.txt）

```
100 40 7900 100
150 40 4100 100
200 40 7600 100
250 40 4500 100
300 40 7200 100
350 40 4800 100
400 40 7900 100
450 40 4100 100
500 40 7600 100
550 40 4500 100
600 40 7200 100
650 40 4800 100
```

演習 29 上のプログラムに対して，音響信号作成時に音響信号の速さを任意に変えられるよう拡張せよ．

演習 30 「sound1.txt」について，通常の速度と高速（5〜10 倍程度）でどのように聞こえ方が異なるか考察せよ．

演習 31 次の 2 つの音パターンを作成し，聞こえ方がどのように異なるか考察せよ.

（ファイル名：sound2.txt）

```
100 10 7700 100
110 10 7600 100
120 10 7700 100
140 10 7700 100
150 10 7600 100
160 10 7700 100
180 10 7700 100
190 10 7600 100
200 10 7700 100
220 10 7700 100
230 10 7600 100
240 10 7700 100
```

（ファイル名：sound3.txt）

```
100 10 7700 100
110 10 6000 100
120 10 7700 100
140 10 7700 100
150 10 6000 100
160 10 7700 100
180 10 7700 100
190 10 6000 100
200 10 7700 100
220 10 7700 100
230 10 6000 100
240 10 7700 100
```

3.3 フーリエ変換とスペクトログラム

フーリエ変換

フーリエ変換 (Fourier transform) とは，本来，無限に続く周期信号の周波数を，正弦波の加算の形で表現する技術である．いま，周期信号を $f(t)$ とすると，$f(t)$ を周波数の関数 $F(\omega)$ を使って

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega$$

と表され，逆に，$F(\omega)$ は

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\exp(j\omega t)dt$$

と表される（j：虚数単位）．この式により，与えられた時間領域の信号を周波数領域に変換することをフーリエ変換という．

離散フーリエ変換

上のフーリエ変換を離散化したものを**離散フーリエ変換** (discrete Fourier transform; DFT) と呼ぶ．離散信号 $y(n)$ に対する離散フーリエ変換は，

$$Y(e^{j\omega}) = \sum_{n} y(n)\exp(-j\omega n)$$

と定義される[2]．

高速フーリエ変換

高速フーリエ変換 (fast Fourier transform; FFT) は，離散フーリエ変換を高速に行うアルゴリズムである．2 つに分割する処理を再帰的に繰り返すため，信号の長さが 2 のべき乗でなければならない．

MATLAB では，高速フーリエ変換の関数が用意されている．これは，信号の長さが 2 のべき乗で無い場合には，本来の離散フーリエ変換の定義式にしたがって計算が行われる．

例題 10 前節で合成した音に対してフーリエ変換をしてみよう．

```
>> t = 0 : 1/44100 : 5;                          % 時間軸の作成
>> y = 0.9 * sin(2 * pi * 440 * t);              % 440Hz の正弦波を作成
>> Y = fft(y);                                   % フーリエ変換
>> f = linspace(0, 44100, length(Y));            % 周波数軸を作成
>> plot(f, abs(Y));                              % フーリエ変換結果を図示[3]
```

フーリエ変換後のデータは，元の信号と同じ長さのベクトルであり，0〜f_sHz（f_s：サンプリング周波数）に対応している．しかし，**シャノンの標本化定理**[4] より，再現される周波数成分は 0〜f_s/2Hz のみである．そこで，この区間のみを描画する．

---

[2] 時間軸が離散的な場合，時間を t ではなく n を使って表すことがある．

[3] フーリエ変換を行うと，定義式から分かるように複素数になる（**複素スペクトル**という）．しかし，通常，興味があるのは，どの周波数にどの程度の強さの成分があるかである．そこで，abs 関数で複素数の絶対値（大きさ）を図示している（**パワースペクトル**という）．

[4] ある信号に最大 fHz の周波数成分が含まれているならば，$2f$Hz 以上の周波数でサンプリングしなければ，信号を再現できない．逆に言えば，f_sHz でサンプリングした信号からは，f_s/2Hz までの周波数成分しか再現されない．

```
>> t = 0 : 1/44100 : 5;                          % 時間軸の作成
>> y = 0.9 * sin(2 * pi * 440 * t);              % 440Hz の正弦波を作成
>> Y = fft(y);                                   % フーリエ変換
>> Y = Y(1 : length(y) / 2 + 1);                 % フーリエ変換結果の後半分を破棄
>> f = linspace(0, 22050, length(Y));            % 周波数軸を作成
>> plot(f, abs(Y));                              % フーリエ変換結果を図示
```

短時間フーリエ変換

上で述べたフーリエ変換は，原理的に定常信号にしか適用できない．しかし，音を含むほとんどの信号は非定常的であるため，非定常信号に適用できる技術が不可欠である．そこで，非定常信号の一部を取り出し，取り出した信号は定常である（すなわち，取り出した部分が無限に続く）と仮定し，フーリエ変換を適用する．これを**短時間フーリエ変換** (short-time Fourier transform, short-term Fourier transform; STFT) という．

```
>> [y, fs, bits] = wavread('sample1.wav');      % sample1.wav を読み込む
>> y1 = y(1 * fs + (0 : 4095));                 % 開始から 1 秒後の 4096 点を取り出す
>> Y1 = fft(y1);                                % フーリエ変換
>> Y1 = Y1(1 : length(y1) / 2 + 1);             % フーリエ変換結果の後半分を破棄
>> f = linspace(0, fs/2, length(Y));            % 周波数軸を作成
>> plot(f, abs(Y));                             % フーリエ変換結果を図示
```

窓関数

上で行った，非定常信号から一部を取り出す（たとえば開始から N 点取り出す）処理は，元の信号に次の関数を掛け算することに相当する：

$$w(n) = \begin{cases} 1 & (\text{if } \ 0 \leq n \leq N-1) \\ 0 & (\text{if } \ n \geq N) \end{cases}$$

この関数を**方形窓** (rectangular window) という．短時間フーリエ変換は，この方形窓で取り出した部分が無限に続くと仮定している．しかし，窓長 N（窓幅ともいう）が信号の周期の正数倍でないと，この無限に続く信号に不連続点が生じ，本来存在しないピークが観測される．そこで，この問題を避けるため，次の関数が提案されている．

- **ハニング窓** (Hanning window)：$w(n) = \begin{cases} 0.5 - 0.5\cos\left(\dfrac{2\pi n}{N-1}\right) & (\text{if } \ 0 \leq n \leq N-1) \\ 0 & (\text{if } \ n \geq N) \end{cases}$
- **ハミング窓** (Hamming window)：$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\ pin}{N-1}\right) & (\text{if } \ 0 \leq n \leq N-1) \\ 0 & (\text{if } \ n \geq N) \end{cases}$

これらの信号の一部を取り出すための関数を総称して**窓関数** (window function) と呼ぶ．

MATLAB では，ハニング窓は hanning(N)，ハミング窓は hamming(N) で簡単に作成できる．

```
>> [y, fs, bits] = wavread('sample1.wav');      % sample1.wav を読み込む
>> y1 = y(1 * fs + (0 : 4095));                 % 開始から 1 秒後の 4096 点を取り出す
>> w = hanning(y1);                             % ハニング窓作成
>> y1 = w .* y1;                                % 作成したハニング窓をかける
>> Y1 = fft(y1);                                % フーリエ変換
>> Y1 = Y1(1 : length(y1) / 2 + 1);             % フーリエ変換結果の後半分を破棄
>> f = linspace(0, fs/2, length(Y));            % 周波数軸を作成
>> plot(f, abs(Y));                             % フーリエ変換結果を図示
```

スペクトログラム

スペクトログラム (spectrogram) とは，横軸に時刻，縦軸に周波数をとり，各時刻，各周波数における信号の強さ（パワー）を色で表したものである．これは，信号の取り出す部分をずらしながら短時間フーリエ変換を繰り返すことで得られる．MATLAB では，specgram 関数を使うことで簡単に得られる．

specgram(y, N, f_s, w, N_{overlap}) y：信号が格納されたベクトル
N：窓長
f_s：サンプリング周波数
w：窓関数
N_{overlap}：信号の取り出す部分をずらすのにどの程度オーバラップさせるか

```
>> [y, fs, bits] = wavread('sample1.wav');      % sample1.wav を読み込む
>> specgram(y, 4096, fs, hanning(4096), 4096 - 0.01 * fs);
                                                % スペクトログラム作成・表示
```

演習 32 スペクトログラムは，横軸が時刻，縦軸が周波数を表すところが，音楽で使われる楽譜と共通している．簡単な旋律の演奏を収録した wav ファイルからスペクトログラムを作成し，スペクトログラムと楽譜を見比べよ．

演習 33 specgram 関数は，fft 関数の繰り返しを for 文を使わずに実現している．specgram 関数のソースファイルを読み，どのように実現しているか理解せよ．

また，

[Y, F] = specgram(y, N, f_s, w, N_{overlap})

とすると，各時刻のスペクトルを行列に格納することができる．

```
>> [y, fs, bits] = wavread('sample1.wav');      % sample1.wav を読み込む
>> [Y, F] = specgram(y, 4096, fs, hanning(4096), 4096 - 0.01 * fs);
                                                % スペクトログラム作成
>> plot(F, Y(:, 100));                  % 開始から 1 秒後の時点のパワースペクトルを表示
```

3.4 基本周波数推定

与えられた音響信号から，その基本周波数を推定することは，音響信号処理の最も基本的かつ重要な処理である．ここでは，基本周波数推定の手法はいくつか紹介する．

最も単純な方法：パワースペクトルのピークの位置から求める

短時間フーリエ変換によりパワースペクトルを求め，そのピークの位置から基本周波数を推定する方法である．パワースペクトルのピークの位置は，パワースペクトルの導関数の零交差点であり，2 次導関数が負である点を調べればよい．零交差点は，その隣りの点の値との積が負になる点である．

```
function F0 = estimateF0(filename, th)

[y, fs, bits] = wavread(filename);                  % sample1.wav を読み込む
[Y, F] = specgram(y, 4096, fs, hanning(4096), 4096 - 0.01 * fs);
                                                    % スペクトログラム作成
Y = abs(Y);
Y(max(max(Y)) ./ Y > th) = 0;                       % パワーの小さいピークを削除
for n = 1 : size(Y, 2)
  Y1 = Y(:, n);                                     % n 番目の時刻のスペクトル
  dY1 = diff(Y1);                                   % 導関数
  d2Y1 = diff(dY1);                                 % 2 次導関数
  s = sign(dY1(1 : end-1)) .* sign(dY1(2 : end));   % 隣あう点の値の符合を掛け算
  i = find(s == -1);                                % 上記の積が負だったら零交差点
  i = i(d2Y1(i) < 0);                               % 2 次導関数が負のものを pick up
  F0(n) = min(F(i));                                % 最も低い周波数が基本周波数
end
```

隣り合う倍音の周波数の差に基づく方法

たとえば，基本周波数が 440Hz の音響信号は，基本周波数成分の周波数が 440Hz，2 次倍音の周波数が 880Hz，3 次倍音の周波数が 1320Hz…というように，隣り合う倍音の周波数の差は基本周波数と等しくなる．そこで，これを求めることで基本周波数推定を行う．

演習 34 この手法に基づく基本周波数推定プログラムを MATLAB で実装せよ．

自己相関関数に基づく基本周波数推定

自己相関関数とは，信号 $y(t)$ とそれをτ秒ずらしたもの $y(t+\tau)$ の相関を測る関数である．通常，周期的な信号は，ちょうど周期の分だけずらすと元の信号と一致する．そこで，τをさまざまな値に変えながら自己相関関数を求め，これが最大になるτが基本周期（逆数が基本周波数）であるとみなすことができる．これを用いた基本周波数推定は，広く用いられている．

演習 35 自己相関関数に基づく基本周波数推定プログラムを MATLAB で実装せよ．

ケプストラム分析に基づく基本周波数推定

ケプストラム分析は，パワースペクトルをスペクトル包絡とスペクトル微細構造に分離する手法である．いま，音声信号 $y(t)$ が，ある基本周波数の信号を発生される音源 $g(t)$ と，スペクトル包絡を変化させるフィルタ $v(t)$ とからなるとすると，音声信号のパワースペクトル $|Y(\omega)|$ は，次式により表される：

$$|Y(\omega)| = |G(\omega)| \cdot |V(\omega)|.$$

両辺に対して対数をとると，

$$\log|Y(\omega)| = \log|G(\omega)| + \log|V(\omega)|.$$

ただし，実際には離散信号を扱うので，

$$\log|Y_k| = \log|G_k| + \log|V_k|$$

である．この式に対して離散フーリエ逆変換を適用すると，

$$C_n = \frac{1}{N}\sum_k \log G_k \exp\left(j\frac{2\pi kn}{N}\right) + \frac{1}{N}\sum_k \log V_k \exp\left(j\frac{2\pi kn}{N}\right)$$

となる．これを**ケプストラム係数** (cepstrum coefficient) という．また，周波数をフーリエ逆変換したところから，ケプストラムの軸を**ケフレンシー** (quefrency) という．スペクトル包絡は低ケフレンシー部に，微細構造は高ケフレンシー部に集中し，微細構造の周期が，元の信号の基本周期に相当するので，高ケフレンシー部のケプストラムの周期を求めることで，音声信号の基本周波数を推定することができる．

演習 36 ケプストラム分析による基本周波数推定プログラムを MATLAB で実装せよ．

演習 37 上で紹介したもの以外の基本周波数推定手法を調べ，MATLAB で実装せよ．

演習 38 3〜4 人程度でグループを作って**演習 34〜37** を分担して行い，基本周波数推定の比較実験を行え．

4 課題

次の 2 つの課題のいずれかを選ぶこと．

課題 1 3.2 節を参考に，自分で独自に聴覚心理学の問題を考え，簡単な実験を行え．

課題 2 自分で独自に調べた，あるいは独自に考案した基本周波数推定手法を実装し，実験せよ．実験は，計算機上で作成した音響信号の他に，楽器音などさまざまな音響信号に対して行い，また，3.4 節で紹介した基本周波数推定手法と比較すること．

いずれも，ただ単に実験を行うだけでなく，結果に対して考察を加えること．

演習の最終日に 5 分程度のプレゼンテーションを行ってもらった上で，LaTeX または Word などのワープロソフトで作成した 2〜4 ページ程度のレポート（内容はプレゼンと同じで構わない）を提出してもらう予定．

大まかな採点基準

90 点以上	実験に十分な考察が加えられている他，独自の工夫がみられる．
80 点	実験しただけでなく，十分な考察がなされている．
70 点	実験しただけでなく，多少の考察がなされている．
60 点	最低限の実験はしている．
60 点未満	最低限の実験すらできていない．

5　参考文献

MATLAB 全般の使い方の学習には，次の文献が役に立つ：

[1] “Get Started with MATLAB”, MATLAB 付属マニュアル.
[2] 上坂 吉則：“MATLAB プログラム入門”，牧野書店，2000.
[3] 小林 一行：“MATLAB 活用ブック”，秀和システム，2001.
[4] 小国 力：“MATLAB と利用の実際—現在の応用数学と CG—”，サイエンス社，1995.

ディジタル信号処理の基礎を学ぶには，次の文献が適している：

[5] 中村 尚五：“ビギナーズデジタルフーリエ変換”，東京電機大学出版局，1989.
[6] 南 茂夫：“科学計測のためのデータ処理入門”，CQ 出版社，2002.

フーリエ変換などのスペクトル解析について解説した有名な本としては，次の文献がある：

[7] 日野 幹雄：“スペクトル解析”，朝倉書店，1977.

ディジタル信号処理を MATLAB による演習付きで取り上げた文献としては，次のものがある：

[8] 樋口 龍雄，川又 政征：“MATLAB 対応 ディジタル信号処理”，昭晃堂，2000.
[9] ジェームズ H マクレラン：“MATLAB による DSP 入門”，ピアソン・エデュケーション，2000.

基本周波数推定の解説記事としては，次のものがある：

[10] 鈴木 久喜：“ピッチ抽出の今昔”，日本音響学会誌，Vol.56, No.2, pp.121–128, 2000.