MATLAB Quick-Start

(Version 4　2019 年 12 月 15 日)

東京大学　数理科学研究科
数理・情報教育研究センター
(MATLAB ユーザーグループ)

藤原毅夫

目 次

第I部

MATLABについて

第1章　始めに

2019年4月（実質3月）からMATLABの利用に関する東京大学とMathWorksの間の包括契約がスタートいたします．これは，東京大学の正規メンバーであれば誰でも，特に利用負担等なしで幾つでも利用できるというものです．特別の制限なしに，各自のPC，タブレット，スマートフォンのクラウドサービスなど形を問わず利用できます．
本資料は，MATLABを学部教育に利用し役立てていくための共通のものであり，以下を基本の原則として行っています．

- 学部学生が，特に講義等で手ほどきを受けなくても基本的な利用ができること．
- 教員は，学部学生がこの資料に書いてある程度の知識と技能を持っていることを前提として，MATLAB使用を前提として講義ができること．
- このマニュアルでは，基本的には数学の説明はしない．

1.1　MATLABとは

1.1.1　MATLABとは何に使うプログラムか

C言語やPython，R（アール）などのソフトウェア言語を学ぶことは多くの人にとって必要なことです．しかしすぐに講義で使うのは困難です．MATLABは，初学者にとって負荷の少ないものであると位置付けています．
MATLABは，数値計算，数式処理，統計，画像処理，信号処理あるいはシミュレーションのライブラリが良く揃ったソフトウェアです．またコンパイラを行わずに，使えるという点でも，かなり多様な環境で利用できる点が優れていると思います．

1.1.2 MATLAB 利用のための条件

東京大学の正規メンバーは，本学が提供する MATLAB を利用できます．
正規メンバーとは，正規の学生，正規の教員，正規の職員を意味します．

1.1.3 MATLAB を使うための準備

東京大学では特に学生に対して，自身の PC を購入すること等を要求していません．それらは学生自身が自分の将来像とともに考えるべきことだからです．
MATLAB を利用するためには，以下の準備をしてください．

Step1 **UTokyo Account (*.u-tokyo.ac.jp) を取得してください．**
総ての正規学生・教職員は，UTokyo Account が取得できます．東京大学構成員のための情報サービスの多くがこのアカウントで提供されます．このアカウントが MathWorks Account のユーザー ID となります．
https://www.ecc.u-tokyo.ac.jp/doc/announce/newuser_student.html

Step2 東京大学ホームページの下にある [情報システム本部]→[サービス案内]→[UTokyo MALAB Campus-Wide Licence] (下記 URL)
https://www.u-tokyo.ac.jp/adm/dics/ja/matlabcwl.html から，東京大学用 MATLAB ポータルサイトに跳んでください．

Step3 **東京大学用 MATLAB ポータルサイトで、 UTokyo Account を使用して認証後，MathWorks Account を取得*し、MATLAB をインストールしてください．**
＊既に、UTokyo Account を使用して MathWorks Account を持っている場合は新たにこれを作成する必要はありません．(必ず UTokyo Account で MathWorks Account を取得するようにしてください．)

Step4 **どこに居ても何時でも，必要な時に**，MATLAB を利用してください．

注意 本学の計算機資源の利用は、教育・研究に関する目的に限定されています（情報倫理・コンピュータ利用ガイドライン）．個人の遊びなどの利便を想定してはいません．特に教室内での UTokyo WiFi の使用は，通信トラフィックを害するので注意してください．

1.1.4　MATLAB 利用の環境

MATLAB は以下のような利用が可能です.

(1) 自分の PC 等（Windows，Mac，Linux）にインストールして，スタンド・アローンでの利用.

(2) タブレット端末やコンピュータのウェブブラウザでの利用　（MATLAB online）.

(3) スマートフォン（iPhone, Android 端末）からの利用（MATLAB mobile）.

これについても [UTokyo MALAB Campus-Wide Licence] の URL（以下）を参照してください.
https://www.u-tokyo.ac.jp/adm/dics/ja/matlabcwl.html

1.1.5　MATLAB の構成

MATLAB は「**MATLAB** と **Toolbox** 群」から構成されています．シミュレーションをする場合に，システムをモデル化し複数のプログラムコードを結び付けて使用するために，**Simulink** を用いなくてはいけないことがあります．Simulink を用いると，実行したいシミュレーションをコントロールするための GUI(Graphical User Interface）を使うことができます.
東京大学は，当面すべての Toolbox が利用できる契約を結んでいますが，不要な Toolbox をダウンロードすることはお勧めできません.
すべての Toolbox をインストールすると約 23 GB のディスクスペースを必要とします.
まずは

MATLAB
Curve Fitting Toolbox
Econometric Toolbox
Optimization Toolbox
Statistics and Machine Learning Toolbox
Symbolic Math Toolbox

位をダウンロード，インストールして試してみるのが良いのではないでしょうか．必要に応じていつでも利用する Toolbox を足していくことが可能です.

1.1.5.1 MATLAB

MATLAB は，Toolbox 群が用いているプログラム言語の名称であり，またこれら数値計算プログラムの名称でもあります.

1.1.5.2 Toolbox

Toolbox の名前と用途については，以下を見てください.
・東京大学向け MATLAB ポータルサイト
・https://jp.mathworks.com/products.html

1.2 利用の準備

1.2.1 PC にインストールするには

インストーラをダウンロードして，それをクリックすると，どの Toolbox をダウンロードするか聞いてきます．必要なプログラムの所にチェックを入れてアイコンをクリックすればインストールが始まります.
以上で準備は完了です．インストール完了後は以後ネットワーク接続無しで利用可能です．PC 上に MATLAB 起動のためのアイコンが現れます.

1.2.2 Web サービス -Chrome book, iPad, iPhone など-

教室で大人数が一度に，ダウンロードや Web サービスの利用を行うと，極端にスピードが遅くなることが予想されます．そのような利用が予想されるときは，教員の方は事前に情報システム本部にご相談ください.

1.2.3 MATLAB 学習のためには

MATLAB の使い方は以下を参照するのも良いでしょう.
MATLAB 入門：
https://jp.mathworks.com/help/matlab/getting-started-with-matlab.html

自己学習形式コース：https://matlabacademy.mathworks.com/jp

第II部

対話型利用：電卓のように

第2章　MATLABの基礎

MATLABのインストールが終わり起動すると，いろいろ見慣れないものが出てきます．Webを経由する場合もほぼ同じだと思います．まず電卓並みの作業から始めてみましょう．

2.1　MATLABの起動と利用のスタート

2.1.1　MATLABの起動

インストールが完了すると，PCの画面上にMATLABのアイコンが現れます．これをクリックしてMATLABを起動すると，複数のウィンドウ，ブラウザーが開きます．各ウィンドウ，ブラウザーの形式は，[ホーム]→[設定]→[MATLAB]の中で変更できます．

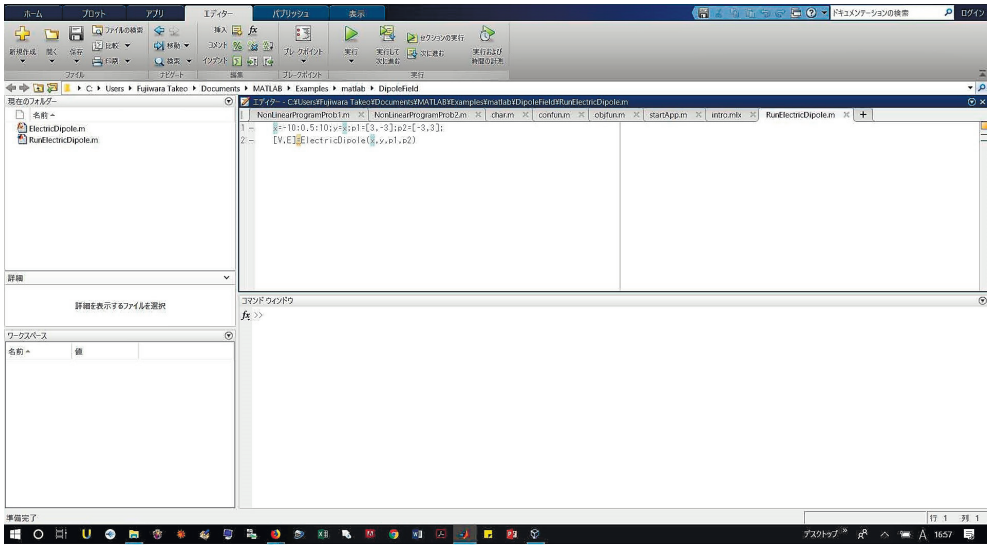


図 2.1: MATLABを起動すると，複数のMATLABウィンドウが開く．

- 「コマンド」ウィンドウ：ここでは対話型に，コマンドを入力して結果を得ることができる．
- 「エディター」: 自分が作成したファイルを編集することができる．新しい計算プログラムを作成，保存するためにもこのウィンドウを使う．
- 「ワークスペース」ブラウザー：現在使用している変数の一覧などが表示される．
- 「現在のフォルダー」ブラウザー：現在，どのフォルダーを使用しているかが表示され，またフォルダーをここで変更することもできる．
- 「コマンド履歴」ウィンドウ：コンソールから入力したコマンドが，入力の順にしたがって表示される．
- 「Figure」 ウィンドウ：図を描くと開く．

2.1.2　MATLAB の使用で迷ったとき

2.1.2.1　誰かに聞く

MATLAB を使用しているとき，どうやったらよいか，どのような関数が用意されているかなどに迷う場合には，色々なやり方で調べることができます．
(1) 製品付属のヘルプ文書を，(オンライン接続しない状態でも）ドキュメンテーション・キーワードで調べることができます．
(2) オンラインヘルプ
(3) オンラインマニュアル
(4) MATLAB ユーザーグループへの質問
(5) ネット検索
(6) 東京大学ユーザーグループへのメールによる問い合わせ
いずれが最も有効であるか，使いやすいかは各自で判断してください．
(2)∼(4) は，MathWorks Account が必要です．MATLAB マニュアルは以下にあります．
https://jp.mathworks.com/help/

(6) は私たち東京大学 MATLAB ユーザーグループが用意した質問サイトです．情報システム本部のサイト
https://www.u-tokyo.ac.jp/adm/dics/ja/matlabcwl.html
の中の WebForm を利用してください．東大ユーザーグループの中の誰かがお答えします.

2.1.2.2 マニュアルについて

MathWorks 社が提供している日本語マニュアルは，英語マニュアルの翻訳です．そのため分かり難いあるいは不正確なことがあるようです．日本語マニュアルが分かり難いと感じたときには，英語マニュアルにも当たるようにしてください.
マニュアルを全部読もうと考えるのは止めましょう．むしろ**一般の web 検索で,「MATLAB　****」と入れる**と効率的にマニュアルの箇所を探すことができます．****は MATLAB のキーワードか数学の術語です.
計算をするときには，マニュアルに書いてあることをウのみにせず，検算をやって確かめてください.

2.2 変数と簡単な計算

以下のステートメントをコマンドウィンドウに表示されているコマンドプロンプト >> の後ろに入力し実行（Return キー) すれば，結果が表示されます。

(a) 変数　*a, b* と　*a+b*　(b) 行列と簡単な行列演算

```
>> a=1
a =
     1

>> b=2.5
b =
    2.5000
>> a+b
ans =
    3.5000
```

```
>> A=[16 3+i 2 13;5 10-i 11 8;9 6 7+2i 12;4 15 14-4i 1]
A =
  16.0000 + 0.0000i   3.0000 + 1.0000i   2.0000 + 0.0000i  13.0000 + 0.0000i
   5.0000 + 0.0000i  10.0000 - 1.0000i  11.0000 + 0.0000i   8.0000 + 0.0000i
   9.0000 + 0.0000i   6.0000 + 0.0000i   7.0000 + 2.0000i  12.0000 + 0.0000i
   4.0000 + 0.0000i  15.0000 + 0.0000i  14.0000 - 4.0000i   1.0000 + 0.0000i
>> inv(A)
ans =
   1.4762 + 1.8175i   3.6349 + 5.6587i  -3.9048 - 5.5952i  -1.4127 - 1.7540i
  -3.2381 + 0.8095i  -9.7143 + 1.8095i   9.7143 - 2.0000i   3.2381 - 1.0000i
   3.2381 - 0.3095i   9.7143 - 0.3095i  -9.7143 + 0.5000i  -3.2381 + 0.5000i
  -1.4286 - 2.1270i  -3.5873 - 6.5873i   3.9048 + 6.5238i   1.4127 + 2.0635i
```

```
>> A'
ans =
  16.0000 + 0.0000i   5.0000 + 0.0000i   9.0000 + 0.0000i   4.0000 + 0.0000i
   3.0000 - 1.0000i  10.0000 + 1.0000i   6.0000 + 0.0000i  15.0000 + 0.0000i
   2.0000 + 0.0000i  11.0000 + 0.0000i   7.0000 - 2.0000i  14.0000 + 4.0000i
  13.0000 + 0.0000i   8.0000 + 0.0000i  12.0000 + 0.0000i   1.0000 + 0.0000i
>> A.'
ans =
  16.0000 + 0.0000i   5.0000 + 0.0000i   9.0000 + 0.0000i   4.0000 + 0.0000i
   3.0000 + 1.0000i  10.0000 - 1.0000i   6.0000 + 0.0000i  15.0000 + 0.0000i
   2.0000 + 0.0000i  11.0000 + 0.0000i   7.0000 + 2.0000i  14.0000 - 4.0000i
  13.0000 + 0.0000i   8.0000 + 0.0000i  12.0000 + 0.0000i   1.0000 + 0.0000i
```

図 2.2: (a) 変数 a, b と $a+b$．(b) 行列の定義と簡単な演算，転置行列.

2.2.1　変数と加減乗除，べき乗，初等数学関数

（図 2.2a)

- >> がコマンドライン，その後ろが結果．変数 a, b の値を設定.
- $a + b$ と打ち込めば，$a + b$ の値が返される．
- その値を例えば c としたければ，$c = a + b$ と打ち込めばよい.
- 乗除算はそれぞれ，$a * b$，a/b.
- x^n は x^n と書く．
- 様々な初等関数は既に定義されている：$\sin(a)$ など.
- 複素数 $x + \mathrm{i}y$ は $x + yi$ あるいは $0.1 + 0.6i$ 等と書く．i の代わりに j と書いてもよい.

演算子の一覧表は
https://jp.mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html

2.3　行列と行列演算

2.2.1　ベクトルおよび行列

（図 2.2b)

- 行列の定義：各要素はコンマ (,) またはスペース（空白）で区切る．
- セミコロン (;) で区切ると，次の行に移ることを示す.
- 行列を転置するにはドットダッシュ (.') を付ける．これは**数学の記号（上付きの t または T）**とは異なる．
- 行列のエルミート共役をとるにはダッシュ (') を付ける．実行列ではドットダッシュ (.') と同じ．これは**数学の記号（上付きの*）**とは異なる．
- 行ベクトルは 1 行 n 列の行列として定義.

- 列ベクトルは n 行 1 列の行列として定義するか，あるいは行ベクトルを転置する．
- A の逆行列は $\mathrm{inv}(A)$.

2.3.2 行列演算

MATLAB は行列が重要な役割を果たします．そのため通常の線形代数における演算規則と異なる MATLAB 特有な計算もあります．その他の言語，例えば Python，でも利用されます．

```
>> A=[1 2 3;4 5 6;7 8 9]
A =
   1   2   3
   4   5   6
   7   8   9
>> A+10
ans =
   11   12   13
   14   15   16
   17   18   19
>> sin(A)
ans =
   0.8415   0.9093   0.1411
  -0.7568  -0.9589  -0.2794
   0.6570   0.9894   0.4121
```

```
>> B=[sin(1) sin(2) sin(3);
      sin(4) sin(5) sin(6);
      sin(7) sin(8) sin(9)]
B =
   0.8415   0.9093   0.1411
  -0.7568  -0.9589  -0.2794
   0.6570   0.9894   0.4121
>> B==sin(A)
ans =
  3×3 の logical 配列
   1   1   1
   1   1   1
   1   1   1
```

```
>> A*B
ans =
   1.2988   1.9595   0.8186
   3.5238   4.7787   1.6401
   5.7488   7.5979   2.4616
>> A.*B
ans =
   0.8415   1.8186   0.4234
  -3.0272  -4.7946  -1.6765
   4.5989   7.9149   3.7091
```

図 2.3: (左) 行列 A の定義，$A+10$, $\sin(A)$. (中央）$B=\sin(A_{ij})$ および B が $\sin(A)$ となっていることの確認．（右）$A*B$, $A.*B$

行列 A を定義します（図 2.3）.

- 行列 A の i,j 要素は $A(i,j)$. $A_{ij}=A(i,j)$
- $A+10$ は，A の各要素に 10 が足されます．$(A+10)_{ij}=A_{ij}+10$. **これは数学での規則とは異なります**.
- $\sin(A)$ の各要素は
$$(\sin(A))_{ij}=\sin(A_{ij})$$
となります．**通常，数学での定義は 2.3.3 です**.

- A, B がそれぞれ同じ大きさの行列であるとき，$A * B$ は通常の行列の掛け算

$$(A * B)_{ij} = \sum_k A_{ik} B_{kj}$$

です．
しかし，MATLAB 特有の演算 $A. * B$ は要素ごとの掛け算，

$$(A. * B)_{ij} = A_{ij} B_{ij}$$

です．これを特に**アダマール積** (Hadamard product) または**シュール積**といいます．

- .^　および　./　も同様な要素ごとの演算です．これを**アダマール冪** (Hadamard power)，**アダマール除算** (Hadamard division) と呼びます．

- 行列 A, B に対して $a = [A, B]$ あるいは $a = [A\ B]$（A と B の間にはスペース）は，行列 A と B を連結させ，A と B を横に並べます．

- $b = [A; B]$ は，行列 A と B を連結させ，A と B を縦に並べます．

(図 2.3)

MATLAB で A.^ (-1) と A^ (-1) は違うものです．また 1/A と入れると次のように返ってきます．

エラー: /
行列の次元は一致しなければなりません。

これらは上に示したことから直ぐに理解できるはずです．試みに A として例えば 2×2 の行列を入れていろいろ試してみて下さい．

アダマール積その他の演算は注意しなくてはなりませんが，慣れると便利です．また途中で使った「$A == B$」は，A と B は等しいか，という**関係演算子**といわれるものです．等しい要素の所は 1，異なる要素の所は 0 を返します．後で出てきますが,「$f(x) == 0$」は「恒等的に $(x) = 0$ という関係が成り立つ」という意味になります．

2.3.3 行列関数について

行列 A が関係する計算については，MATLAB では要素ごとの計算をします．しかし数学（線形代数）では次のように定義します．

$$A^n = \overbrace{A * A * \cdots * A}^{n\text{ 個の }A} \tag{2.1a}$$

$$\mathrm{e}^{\mathrm{A}} = E + A + \frac{1}{2!}A^2 + \cdots + \frac{1}{n!}A^n + \cdots \tag{2.1b}$$

$$\sin A = A + \frac{1}{3!}A^3 + \cdots + \frac{1}{(2n+1)!}A^{2n+1} + \cdots \tag{2.1c}$$

$$\cos A = E + \frac{1}{2!}A^2 + \cdots + \frac{1}{(2n)!}A^{2n} + \cdots \tag{2.1d}$$

これらを行列関数と呼び，MATLAB では別に定義をします．これについてはもう少し後で説明します．

第3章　線形代数：初級編

3.1　連立1次方程式

3.1.1　逆行列の計算

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{cases}$$

という連立1次方程式は，行列とベクトルを用いて次のように書けます.

$$A\boldsymbol{x} = \boldsymbol{b}$$

ここで A は $n \times n$ 行列, $\boldsymbol{x}$, $\boldsymbol{b}$ は n 次元縦ベクトルです．これに左から逆

```
>> A = fix(10*rand(3,3))
A =
    7    6    0
    7    1    2
    3    7    0
>> rank(A)
ans =
    3
>> B=eig(A)
B =
   11.7274
    1.0960
   -4.8235
```

```
>> A00=det(A)
A00 =
  -62.0000

>> C=inv(A)
C =
    0.2258   -0.0000   -0.1935
   -0.0968   -0.0000    0.2258
   -0.7419    0.5000    0.5645
```

```
>>  D=A*C;
>>  E=[1 0 0;0 1 0;0 0 1];
>> D==E
ans =

  3×3 の logical 配列

   1   0   1
   1   1   1
   1   0   1
```

図 3.1: 逆行列の計算．(左) 行列 A の定義，階数，固有値. (中央）行列 A の行列式および $C = A^{-1}$. （右）$AC =$ 単位行列 になることの確認.

行列 A^{-1} をかけて次式を得ます.

$$\boldsymbol{x} = A^{-1}\boldsymbol{b}.$$

ただし行列が大きくなった場合の実際の計算では，A の逆行列を求め，それを $\boldsymbol{b}$ に掛けるという方法はお勧めできません．その理由は，逆行列を直接計算するという方法は計算時間の面からもまた精度の面からいっても，なかなか問題があるからです．この問題は後で触れることにしましょう.

図 3.1に行列計算のいくつかを示します.

- rand は $(0, 1)$ 区間の一様乱数.
- $X = \mathrm{rand}(n_1, \cdots, n_N)$ は，$(n_1 \times \cdots \times n_N)$ 個の乱数が格納された N 次元の $n_1 \times \cdots \times n_N$ 配列を返す.
- fix は 0 方向に少数以下を丸める関数（$\mathrm{fix}(1.9) = 1$, $\mathrm{fix}(-0.9) = 0$).
- $\mathrm{rank}(A)$ は行列の階数（ランク）を返す.
- コマンドの後に ;（セミコロン）を入力すると，実行結果は印字されない.

最後に $D == E$ としたとき，3×3 の総ての成分が 1 であることを期待しました．しかしいくつかの成分は 0（等しくない）という答えを返しました．何故でしょうか.（もし分からなければ $D - E$ と入れてみてください．）二つの実数の比較には注意が必要です.

3.1.2 連立 1 次方程式の MATLAB における適切な解法

$$A\boldsymbol{x} = \boldsymbol{b}$$

の解法は（$\boldsymbol{x} = A^{-1}\boldsymbol{b}$ ではなく）

$$\mathrm{x} = \mathrm{A}\backslash\mathrm{b}$$

です．ここではどのようなアルゴリズムを用いるか説明しません（第 11章を参照）が，行列の扱いは，精度と計算時間が最も典型的に計算手順に依存します．ですから，逆行列を計算することはせずに，上のように書くことを強くお勧めします．これについては，第 11章で，再び触れることにします.

3.2　固有値および固有ベクトル

$n \times n$ 行列 A が与えられたとき次式が得られたとしましょう．これは n この変数（$x_1, x_2, \cdots, x_n$）を持った n 元の連立 1 次方程式です．

$$(A - \lambda E)\boldsymbol{x} = 0\ . \tag{3.1}$$

λ は数，E は $n \times n$ 単位行列，$\boldsymbol{x}$ は縦行列です：$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$．これから λ （**固有値**）とそれに対応した $\boldsymbol{x}$（**固有ベクトル**）を求めようというのが問題です．
このとき自明でない解 $\boldsymbol{x} \neq 0$ が存在するためには，$(A - \lambda E)$ が逆行列を持たないことが必要充分です．従って必要十分条件は

$$\det(A - \lambda E) = 0 \tag{3.2}$$

です．λ を固有値，$\boldsymbol{x}$ を固有ベクトルといいます（図 3.2）．

```
>> A = fix(10*rand(3,3))
A =
    9    7    0
    2    3    0
    7    5    5
>> rank(A)
ans =
    3
>> eig(A)
ans =
   5.0000
   1.2042
  10.7958
```

```
>> [V D]=eig(A)
V =
         0    0.6479   -0.5672
         0   -0.7215   -0.1455
    1.0000   -0.2443   -0.8106
D =
    5.0000         0         0
         0    1.2042         0
         0         0   10.7958
```

図 3.2: (左) 行列 A を一様乱数を用いて定義，その階数と固有値．（右）行列 A の固有値を収めた対角行列 D と，固有ベクトル（縦ベクトル）を並べて作った行列 V．

- $\mathrm{rank}(A)$ は行列 A の階数（rank）を返す．
- $e = \mathrm{eig}(A)$ は，正方行列 A の固有値を要素とする列ベクトルを返す．
- $[V, D] = \mathrm{eig}(A)$ は，固有値からなる対角行列 D と，対応する右固有ベクトルを列にもつ行列 V (つまり $A * V = V * D$) を返す．

第4章　シンボリックな計算

数学で必要なことは，数値計算だけではありません．通常の計算機上の演算は数値に対して行われます．しかしシンボリック演算では数式に対して数式の記号のままの演算が行われます．
MATLAB は**シンボリック演算（数式処理）**の機能も備えています（Symbolic Math Toolbox をインストールすることが必要です）．シンボリック演算をするためには，変数や行列，関数がシンボリックなもの（記号のままで処理されるべきもの）であることを宣言する必要があります．

4.1　シンボリック変数とシンボリック演算

4.1.1　シンボリック変数の定義とシンボリックな処理

シンボリック演算の例を図 4.1に示しましょう．

```
>> s=sym('s')
>> expand((s+1)^3)
ans =
s^3 + 3*s^2 + 3*s + 1
```

```
>> syms y z
>> expand((y+1)^3)
ans =
y^3 + 3*y^2 + 3*y + 1
>> simplify(z^3+3*z^2+3*z+1)
ans =
(z + 1)^3
```

図 4.1: シンボリック変数の定義と計算.

- x をシンボリック変数として定義するためには syms x あるいは $x = \mathrm{sym}('x')$ である．
- expand は式の展開.
- simplify は因数分解など（代数的な単純化）を行う.

行列をシンボリック行列として定義するときは次のようにします.

$$\mathrm{A} = \mathrm{sym}('\mathrm{A}', [2\ 4])$$

これで 2×4 行列 A が定義され，A の i, j 成分は Ai_j という名前になります．sym は行インデックスと列インデックスをアンダースコア underscore（_）で区切りますが，これを変えることもできます.

4.1.2　シンボリックな求解

シンボリックな演算によって解を求めることも可能です．ここでは代数方程式の解の一般的な式を求めることと，三角関数の式の解を求める方法を示してみましょう（図 4.2).

```
>> syms a b c x
>> eqn=a*x^2+b*x+c==0;
>> sol=solve(eqn)
sol =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
>> sol2=solve(eqn,x)
sol2 =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

```
>> syms x
>> eqn=sin(x)==1
eqn =
sin(x) == 1
>> sol3=solve(eqn,x)
Sol3 =
pi/2
```

図 4.2: シンボリックな求解.

- eqn= $\cdots$ == $\cdots$ は恒等式を定義し，それを 'eqn' と名付ける.
- solve(eqn) は恒等式 'eqn' を解く．複数の変数がある場合，変数は 'symvar' によって検索され、アルファベット順に並べて返されるとマニュアルにはある．[1] 複数のシンボリック変数がある場合には，何について解くかを，symvar で書かせて確認するか，あるいは明示的に指定するか，いずれかを勧める.
- solve(eqn,x) は恒等式 eqn を変数 x について解くというコマンドである.

[1] しかし実際にそのルールどおりになっているのかはよく理解できません.

4.2 シンボリック関数演算

シンボリックな表現を用いれば，微積分その他の計算が，数式として行えます.

4.2.1 微分

シンボリックな変数あるいは関数を用いれば，解析的な微分を行うことができます．そのときは演算子

$$\mathrm{diff}(f),\ \mathrm{diff}(f,x)$$

を用います．f は関数です．n 次導関数は

$$\mathrm{diff}(f,n)$$

です.
さらに関数 $f(x)$ について

$$\mathrm{eq2}{=}\mathrm{diff}(f,2)$$

として，2 階微分を求めた後で，$\mathrm{eq2}(x_0)$ を求めれば，$x = x_0$ における 2 回微分係数の値が与えられます.（図 4.3）

```
>> syms x n
>> f(x,n)=x^n
f(x, n) =
x^n
>> df1(x,n)=diff(f,x)
df1(x, n) =
n*x^(n - 1)
>> df1(2,n)
ans =
2^(n - 1)*n
>> df2(x,n)=diff(f,2)
df2(x, n) =
n*x^(n - 2)*(n - 1)
```

```
>> syms x n
>> g(x,n)=n*x^(n-1)
g(x, n) =
n*x^(n - 1)

>> intg(x,n)=int(g,x)
intg(x, n) =
piecewise(n == 0, 0, n ~= 0, x^n)

>> int(g,[1,2])
ans(n) =
piecewise(n == 0, 0, n ~= 0, 2^n - 1)
```

図 4.3: シンボリックな微積分.

4.2.2　偏微分

偏微分に関しては，特に特別なことはありません．

4.2.2.1　偏微分

例えば2変数 x, y の関数 $f(x, y)$ を

```
syms x y
f=x^ 2+x*y+3*y^ 3
```

と定義します．x による（偏）微分は

```
diff(f,x)
```

y による（偏）微分は

```
diff(f,y)
```

です．高階（偏）微分は

```
diff(f,x1,x2, •••• ,xn)
```

と書き，これは f を x1, x2, •••• , xn の順番で微分します．

【問題】

$$f(x, y) = \frac{xy(x^2 - y^2)}{x^2 + y^2}$$

の1階偏微分係数，2階偏微分係数などを計算し，手計算の結果と比べてみてください．

4.2.2.2　多変数関数の勾配

多変数関数 $f(x_1, x_2, \cdots, x_n)$ に関する偏微分（勾配）に関しては，gradient というコマンドがあります．

```
syms x1 x2 ··· xn
[fx1, fx2, ··· , fxn] = gradient(f)
```

f(x) は n 次元配列です．

4.2.2.3 ヤコビアン（ヤコビ行列式）

一般の変数変換を考えてみます.

$$x = x(u, v)$$
$$y = y(u, v)$$

変換により x, y に関する積分を u, v についての積分に変換することを考えましょう．これは一般に

$$\int f(x, y)dxdy = \int f(x(u, v), y(x, v))|J|dudv$$

と書けます．$|J|$ はヤコビ行列式 J の絶対値であり，二つの空間 (x, y) と (u, v) の面素の比です：

$$J = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{pmatrix} = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u}.$$

例えば次のように入力します：

```
syms u v
x = u.^ 2 + v.^ 2
y = 2 * x. * y
jacobian([x, y], [u, v])
```

4.2.3 テイラー級数展開

微分学における重要な結果の 1 つは，テイラー級数展開です．MATLAB でテイラー級数展開を行うのは簡単です:

```
>> syms x
>> f=log(1+x)
>> taylor(f,'Order',8)
ans =
x^ 7/7 - x^ 6/6 + x^ 5/5 - x^ 4/4 + x^ 3/3 - x^ 2/2 + x
```

また，テイラー展開を任意の点の周りで行うことも可能です：

```
>> syms x
>> taylor(log(1+x),x,'ExpansionPoint',0)
ans =
x^ 5/5 - x^ 4/4 + x^ 3/3 - x^ 2/2 + x
>> taylor(log(1+x),x,'ExpansionPoint',1)
ans =
x/2 + log(2) - (x - 1)^ 2/8 + (x - 1)^ 3/24 - (x - 1)^ 4/64
+ (x - 1)^ 5/160 - 1/2
```

【問題】対数関数 log(1+x) を x=0 の他，幾つかの点例えば x=1/2 の周りでテイラー展開してみてください．さらにそれらを図に描くことで，収束の様子（あるいは元の関数値との比較）を調べてみましょう．

上のように**式または関数**を図に描くときは，

```
fplot(f,[x1 x2])
```

を使います．(plot(x,y) は点 (x, y) を線分で結ぶコマンドです.）[x1 x2] はグラフを描く変数の値の範囲を示します．これを省略するとデフォルトの範囲 [-5 5] で描画します．

4.2.4　積分

数式処理による積分（不定積分）は

```
int(expr,var)
```

です．シンボリックなスカラー変数 var について 関数 expr の不定積分を計算します．
さらにこれの定積分の値を求めたいときは，例えば積分区間を $[1.0, 2.0]$ とするならば

```
int(expr, [1,2])
```

とします．(図 4.3)
~= は不等号 $\neq$ です．

2 重積分および 3 重積分用のコマンド

```
integral2, integral3
```

等も用意されています．一般には，積分の方がシンボリック演算のプログラムを用意するのは難しいようです．

第5章　グラフ

5.1　MATLABの描画機能

計算結果をグラフに描くことができれば，私たちの理解が進みます．ですから計算の結果をグラフにしてみることを勧めます．
色々な場合を図にするのは，手作業で実行するのは難しいものです．コンピュータプログラムでこれが実行できるようになって，作業が著しく簡単になりました．しかし，同時にグラフの意味するところを時間をかけて吟味することが疎かになったというのも事実です．昔は手間をかけてグラフ描きながら，いろいろなことを考えていたということなのでしょう．空いた時間を考えることに使うよう，心してかからなくてはなりません．

5.1.1　グラフの種類

プロットルーティンには x 軸，y 軸を共に**線形**とするものの他，**片対数**プロットである semilogx, semilogy，**両対数**プロットである loglog があります．
また線の**太さと種類，色**，点に**マーカ**ー付与，などができます．さらに軸に**軸ラベル**を描くことや，**図のタイトル**なども書くことができます．

プロットは目的により様々なものがあります．

- 等高線図,
- 局面
- 3 次元空間のベクトル図（矢印入り）

などなどです．ここではそれらを例示しませんが，インターネットで ‘MATLAB プロットのタイプ’ と検索してみてください．下のようなサイトが見つかるでしょう．

https://jp.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html
この中から必要なタイプのものを選ぶと良いでしょう．

5.1.2　2 次元プロット

最も一般的な 2 次元プロットについて述べてみましょう．これは独立変数 x に対する関数 $f(x)$ の振る舞いを図にします（図 5.1）．

```
>> x=[0:0.1:2*pi];
>> y=[sin(x);cos(x)];
>> plot(x,y)
```

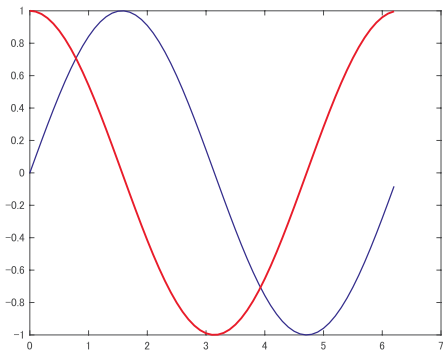


図 5.1: 2 次元のグラフプロット．コマンドと $\sin(x)$, $\cos(x)$ の図．

- ここで変数 x はベクトルとして定義．[0:0.1:2*pi] は，0 から 2π まで 0.1 刻みで与えるということ．x がベクトルであるから $\sin(x)$, $\cos(x)$ はベクトルとして定義される．このように定義されることは，行列のところで説明した．**ここが MATLAB 特有の関数定義の妙味の 1 つである．**
- コマンドの最後にセミコロン（；）を入力すると，計算結果は出力されない．
- 最後の描画コマンド $\mathrm{plot}(x, y)$ は 2 次元の点 (x_i, y_i) を直線でつなぐ，2 次元プロットのためのコマンド．これにより，横軸に x，縦軸に y をとった図が，Figure ウィンドウに出力される．
- clf コマンドで Figure ウィンドウはクリアされる．

5.1.3 陰関数プロット

ここで便利な機能を示しておきましょう.

```
>> f=@(x,y) x.^2+x.*y+2*y.^2-1
>> fimplicit(f,[-1.5 1.5 -2 2])
```

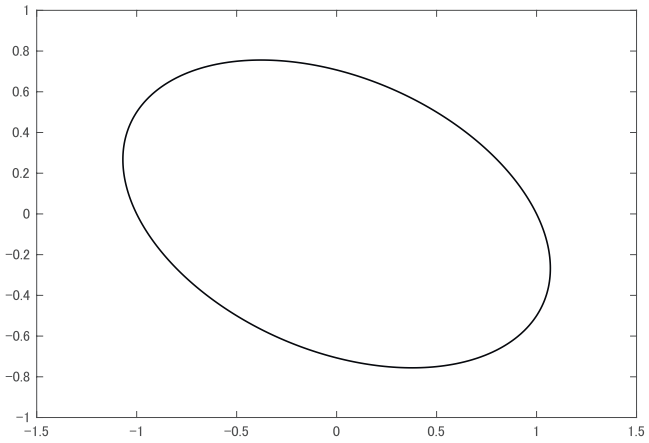


図 5.2: $f(x,y) = x^2 + xy + 2y^2 - 1 = 0$ の図.

図 5.2では，方程式 $f(x,y) = 0$ により定義された x の関数 $y(x)$ をプロットしています．このような機能を用いて図が得られれば，関数の性質を理解する大きな助けになるでしょう.

f = @(x,y) x.^ 2 + x.*y+2*y.^ 2 - 1

で（x,y）を変数とした関数 f （無名関数，anonymous function）が定義されます.

fimplicit(f,[-1.5 1.5 -1.0 1.0])

により，方程式 $f(x,y) = 0$ が決める曲線 (x,y) を，領域 $-1.5 \leq x \leq 1.5$, $-1.0 \leq y \leq 1.0$ で描きます.

5.1.4 3 次元プロット

3 次元空間で，上昇とともに半径が拡大する渦を描いてみましょう（図 5.3）.

plot3(x,y,z) は 3 次元座標データ (x_i, y_i, z_i) を直線でつなぎます．プロットされたデータは，**マウスの操作によりズーム，移動，回転など**ができます．3 次元グラフを様々な方向から見るなどしてより良い理解を得ることができます.

```
>> z = 0:pi/50:10*pi;
>> sz = z.*sin(z);
>> cz = z.*cos(z);
>> plot3(sz,cz,z)
```

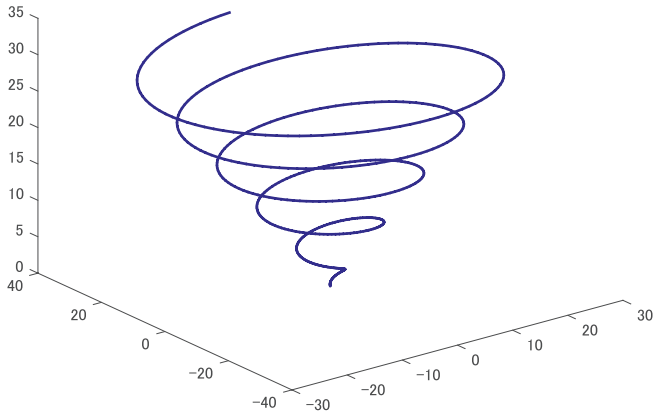


図 5.3: 3 次元プロット．コマンドと $x*\sin(z)$, $z*\cos(z)$ での図.

5.2　グラフを描く意味

関数の性質を知るために，その関数を可視化してみることは大いに意味があります.
数学的命題を考える際，証明という方法が重要ですし, 必要です．しかし，証明という段階が無くても，図に描けば十分であるというわけではありません．証明の前に，可視化という手順で性質を把握するのは，実際的で，重要な指針を得るという点からも必要です.

5.2.1　関数の極限，収束性の振る舞いを見る

対数関数

$$y = \log x$$

は

$$x = \exp y$$

の逆関数として定義されます.

$$\lim_{x\to 0+} \log x = -\infty \ , \quad \lim_{x\to\infty} \log x = +\infty$$

はよく知られた性質です．それでは $\alpha,\ \beta > 0$ としたとき

$$\lim_{x\to 0+} x^{\alpha} \log x \ , \quad \lim_{x\to\infty} x^{-\beta} \log x$$

はどうなるでしょう．極限値を持つのでしょうか.

最初の $\lim_{x\to 0+} x^\alpha \log x$ については $t = x^\alpha$ と置くと

$$x^\alpha \log x = \frac{1}{\alpha} t \log t$$

です（$x \sim 0+$ では $t \sim 0+$）．同様に $t = x^{-\beta}$ と置けば

$$x^{-\beta} \log x = -\frac{1}{\beta} t \log t$$

です（$x \sim +\infty$ では $t \sim 0+$）．上から，$\lim_{x\to 0+} x^\alpha \log x$ でも，あるいは $\lim_{x\to\infty} x^{-\beta} \log x$ の場合でも，$t \to 0+$ での $t \log t$ の振る舞いが問題であることが分かります（図 5.4）．

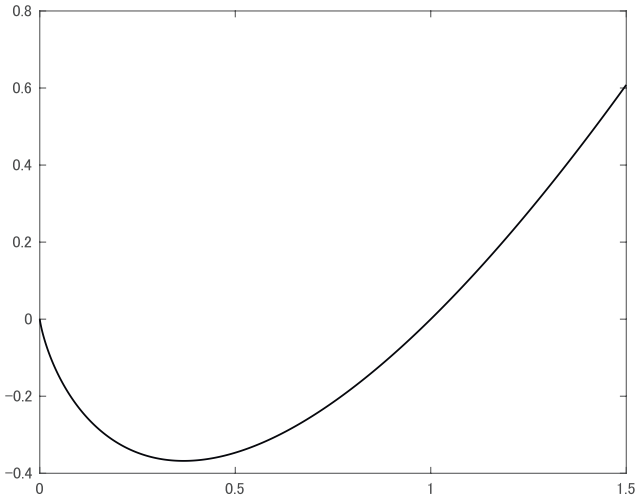


図 5.4: $t \log t$.

$u = \log t$ と置けば $t = \mathrm{e}^u$ ですから

$$t \log t = u\mathrm{e}^u$$

です．$x \to 0+$, $(t \to 0+)$ および $x \to \infty$, $(t \to 0+)$ のいずれでも $u \to -\infty$ であり，上式の e^u が（u に比べて）遥かに速く 0 となるので

$$\lim_{u\to -\infty} u\mathrm{e}^u = 0$$

です．こうして $\alpha,\ \beta > 0$ としたとき

$$\lim_{x\to 0+} x^\alpha \log x = 0\ , \quad \lim_{x\to\infty} x^{-\beta} \log x = 0$$

という極限値を持ちます．つまり $|\log x|$ は，$x \to 0+$ で x のどんな負冪よりゆっくりと無限大になり，$x \to +\infty$ で x のどんな正冪よりゆっくりと無限大になるのです．
一般には上のようにして式で示すのが一般的ですが，その前にここでやったように図に描いてみると，良く分かるのではないでしょうか．

5.2.2　関数の大局的な振る舞いを見る

きちんと証明するのは難しくても，関数の性質を大掴みに理解しようという際にも図は大いに役立ちます．ワイエルシュトラス関数（Weierstrass）

$$f(x) = \sum_{n=0}^{\infty} a^n \cos(b^n x) \ , \quad (0 < a < 1, b \text{ は奇数}, ab > 1 + \frac{3\pi}{2})$$

を考えてみましょう．これは「至る所で連続であるが至る所で微分不可能な関数」として知られています．
この級数が「至る所で連続である」ことは，$|a^n \cos(b^n x)| \leq |a|^n$, $|a| < 1$ であることからこの無限級数が収束することが言え，したがって一様連続であることはすぐに分かります．
「至る所で微分不可能な関数」であることを証明するにはもう少し手間がかかりますが，図を描いてみれば納得できると思います（図 5.5）．

```
a=0.5;
b=13.0;
t=[-pi/2*1:pi/10000.0:pi/2*1];
  x1=cos(t);
  x2=x1+cos(13*t);
  x3=x2+cos(13^2*t)
  plot(t,x1);grid on
  hold on
  plot(t,x2);grid on
  hold on
  plot(t,x3);grid on
  hold on
x=zeros(1,numel(t));
  for  n = 0:1:100
     x=x+ a^n*cos(b^n*t);
  end
  plot(t,x)
```

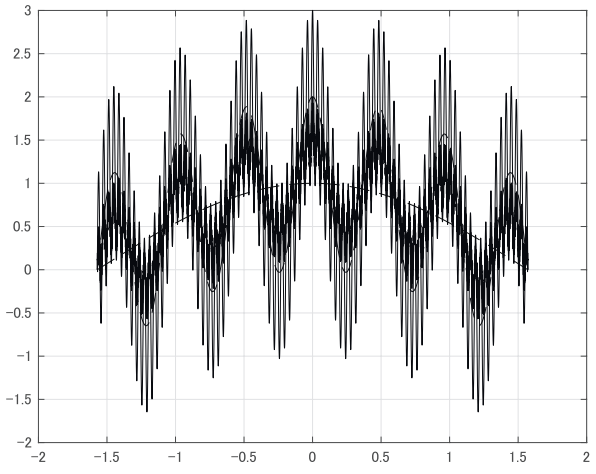


図 5.5: ワイエルシュトラス関数 ($a = 0.5,\ b = 13$).

第III部

非対話型利用：プログラムファイル

第6章　スクリプトの利用

MATLAB の多くの操作は対話型で行うことができます．しかし計算が複雑で多くの手順を要するとき，あるいは同じような計算をデータを変えながら何度も行うとき，それでは手間がかかり過ぎて不便です．そのようなときのためには，プログラム用のファイルあるいはデータのためのファイルを利用するのが便利です．

6.1　スクリプトファイル

スクリプト（script）は，単純な形のプログラム ファイルです．一連のコマンドを繰り返し行う計算を実行するときに便利です．そのプログラムを書くときに使用するのがスクリプト言語です．
通常は，計算プログラム（ソースコード）はコンパイラにより機械語に翻訳しなくてはなりません．これをコンパイル（compile）といいます．スクリプトではこのようなコンパイルの手続きを経ないですぐに計算が実行できます．MATLAB もこのような言語ですので，コンパイルの手続きを要しません．
新しいスクリプトは、次の方法で作成できます．

1. [ホーム] タブにある [新規スクリプト] ボタンをクリックします．すでにスクリプトがある場合には，そのスクリプトファイルをダブルクリックします．

2. エディター（ウィンドウ）にスクリプトファイルの内容が表示されます．ここにプログラムを書き込みます．

3. スクリプトファイルが出来上がったら，[保存] タブをクリックしてこれを保存します．

4. [ホーム] タブの［実行］ボタンをクリックすると，スクリプトは実行され，コマンドウィンドウに実行結果が出ます.

5. スクリプトファイルは名前を付けて保存すると，….m という拡張子が付きます．名前を付けないで保存すると，Untitled.m という名前になります.

第IV部

数学基礎 - 中級編

第7章　最適化

計算が進むと，内容もプログラムも複雑になります．この章あたりから，前章で説明したスクリプトファイルを利用した方が便利でしょう．

7.1　ラグランジュ未定乗数法

7.1.1　ラグランジュ未定乗数法の定式化

束縛条件 $g(x,y)=0$ のもとで、$f(x,y)$ が最大値となる点 (a,b) を求める問題を考えてみましょう．そのためには，新たな変数 λ（**ラグランジュ乗数**と呼びます）を導入して，

$$F(x,y,\lambda)=f(x,y)-\lambda g(x,y)$$

を考えます．点 (a,b) で $\partial g/\partial x\neq 0,\quad \partial g/\partial y\neq 0$ ならば，

$$\frac{\partial F}{\partial x}=\frac{\partial F}{\partial y}=\frac{\partial F}{\partial \lambda}=0 \tag{7.1}$$

により $\lambda=\alpha$ および最大値を与える点 (a,b) が決まります．

7.1.2　典型的問題と解法

【問題】

$$f(x,y) \;=\; (x+y)^2 \tag{7.2a}$$
$$g(x,y) \;=\; x^2+y^2-1 \tag{7.2b}$$

として，束縛条件 $g(x,y)=0$ のもとで $f(x,y)$ の最適解を求めよ．
【解】

$$F(x,y,\lambda)=f(x,y)-\lambda g(x,y) \tag{7.3}$$

と置いて，ラグランジュ未定乗数法の条件から

$$(1-\lambda)x + y = 0 \tag{7.4a}$$
$$x + (1-\lambda)y = 0 \tag{7.4b}$$
$$x^2 + y^2 = 1 \tag{7.4c}$$

が得られます．これを解いて，$\lambda = 2$ 又は 0 を得ます．この λ を用いれば以下のような結果が得られます．

$$\lambda = 2 \text{ のとき} \qquad x = y = \pm\frac{1}{\sqrt{2}}, \qquad f = 2 \tag{7.5a}$$
$$\lambda = 0 \text{ のとき} \qquad x = -y = \pm\frac{1}{\sqrt{2}}, \qquad f = 0 \tag{7.5b}$$

7.1.3　MATLAB の利用

これを MATLAB で解いてみましょう．(図 7.1)

```
syms x y lambda  f(x,y)  g(x,y)  real
% maxmize
f(x,y) = (x+y)^2;
% subject to
g(x,y) = x^2 + y ^2 - 1;
L(x,y,lambda) = f - lambda*g
eqn1 = diff(L,x) == 0;
eqn2 = diff(L,y) == 0;
eqn3 = diff(L,lambda) == 0;
ss = solve([eqn1,eqn2,eqn3],[x,y,lambda]);
ss.x
ss.y
ss.lambda
f(ss.x,ss.y)
T =
table(double(ss.x),double(ss.y),double(ss.lambda),double(f(ss.x,ss.y)));
T.Properties.VariableNames =
{'x','y','lambda','f'}
```

```
>> LagrangeMultiplier
L(x, y, lambda) =
(x + y)^2 - lambda*(x^2 + y^2 - 1)
ans =
 2^(1/2)/2
-2^(1/2)/2
-2^(1/2)/2
 2^(1/2)/2
ans =
-2^(1/2)/2
 2^(1/2)/2
-2^(1/2)/2
 2^(1/2)/2
ans =
0
0
2
2
ans =
0
0
2
2
```

```
T =

  4×4 table

       x          y       lambda    f
    _______    _______    ______    _

    0.70711    -0.70711    0     0
    -0.70711    0.70711    0     0
    -0.70711    -0.70711   2     2
    0.70711     0.70711    2     2
```

図 7.1: Lagrange 未定乗数法．(左) プログラムファイル，(中）(右）出力結果．

- %に続く行はコメント行です．

- ss=$\cdots$ では，solve コマンドを用いて，数式　eqn1〜eqn3 を変数 x , y, lambda について解き，その解を縦ベクトル ss.x, ss.y, ss.lambda に格納．ここで作られる ss は構造体と呼ばれる．ここでは行列形式になっている．

- プログラムの中でデータ ss.x ss.y ss.lambda がどう入っているか，f に何が入っているか確認している．その答えが中央の 4 つの ans=$\cdots$ に対応する．

- double(ss.x) 等でシンボリック変数を倍精度数値（double）に変換し，T=table($\cdots$) で table T を定義する．セミコロンがあるから結果は出力されない．

- T.Properties.VariableNames = $\{'x','y','\text{lambda}','f'\}$ では，table プロパティ VariableNames （変数名）を変更して，table 内の各変数の変数名を指定する．その出力が右．

7.2　線形計画法

一般的な関数（目的関数）の大域的な最小値または最大値（とパラメータ空間の場所）を探す問題は，現在は数学としても**（非）線形計画問題**として大きな分野を形成しています．

7.2.1　線形計画法とは

線形計画問題とは，目的関数と制約条件がすべて，値を定めるべき変数について線形の最適化問題のことをいいます．変数が 2 つ $x_1(\geq 0), x_2(\geq 0)$ の場合に一般的に書けば，不等式条件

$$a_{11}x_1 + a_{12}x_2 \leq b_1, \tag{7.6a}$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2 \tag{7.6b}$$

の制約（2 次元空間内で (x_1, x_2) の領域を決める）の下で

$$c_1x_1 + c_2x_2 \tag{7.7}$$

の最大小値およびそのときの x_1, x_2 を求める問題です．

目的関数が線形なので，局所的な解は大域的な意味でも最適解になります．さらに目的関数も線形ですから，最適解は (x_1, x_2) 平面内の凸多角体の境界上に存在するはずです．

7.2.2　線形計画法の問題と考え方

[問題]：
$4x + 2y \leq 30,\ x + 3y \leq 14$ の制約条件の下で
目的関数 $f(x, y) = -5x - 4y$ を最小化せよ（図 7.2）.
[考え方]：
領域 $4x+2y \leq 30,\ x+3y \leq 14$ のをグラフ上に描き，$f(x, y) = -5x-4y =$ 一定 の直線をいろいろ描いてみると，すぐに解答が見つかるでしょう．図 7.2に解答を示してみましょう．

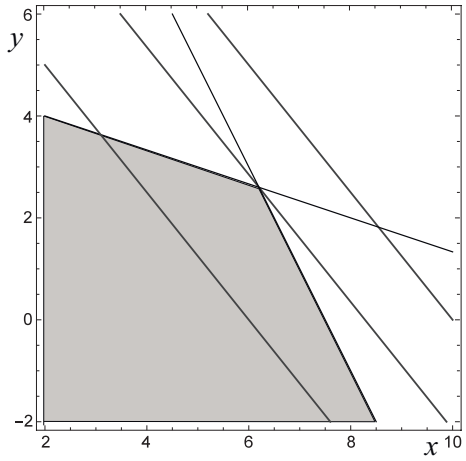


図 7.2: 線形計画問題の例．制約条件 $4x + 2y \leq 30,\ x + 3y \leq 14$ の下で，目的関数 $f(x, y) = -5x - 4y$ を最小化する．グレー部分が制約条件で指定された領域．並行した直線は $f(x, y) = -30, -207/5, -50$ の等高線．$x = 31/5,\ y = 13/5$ で最小値 $a = -207/5$ をとる．図から見て取れるように，最適値をとる場所は，領域の端，2 つの直線の交点となる．

7.2.3　MATLAB の利用

MATLAB には線形計画法のルーチン $\mathrm{linprog}(f, A, b)$ が用意されています．ここで A は行列，f，b はベクトルです（図 7.3）.

- $x = \mathrm{linprog}(f, A, b)$ は $\{A\boldsymbol{x}\}_i \leq b_i \quad (i = 1, 2)$ の条件の下で，$\boldsymbol{f} \cdot \boldsymbol{x}$ の最小値を与える $\boldsymbol{x}$ 求める．
- この問題では

$$A = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 30 \\ 14 \end{pmatrix}, \quad f = \begin{pmatrix} -5 & -4 \end{pmatrix} \tag{7.8}$$

と選べばよい．

- Sol=f*x では，上の問題の解である $\boldsymbol{x}$ を用いてこの値を計算する．

```
%Linear Programing Problem
%Ax ¥le b  ( Constraint Conditions)
%Oprimization of f*x
A=[4 2;1 3]; b=[30;14];f=[-5 -4];
x=linprog(f,A,b)
% x is the point of the solution
Sol=f*x % Answer
```

```
>> LinearProgramProb1
Optimal solution found.
x =
   6.2000
   2.6000
Sol =
  -41.4000
```

図 7.3: 線形計画問題の例．

7.3 非線形計画法

制約条件，目的関数（のすべて，あるいは一部）が一般に非線形である場合，たとえば次のようなものを非線形計画問題といいます．

7.3.1 非線形計画法の問題と解答

[**問題**]：$x \geq 0,\ \mathrm{y} \geq 0$ において，
制約条件 $1 \leq x^2 + y^2 \leq 2$ の下で，
目的関数 $f(x, y) = x^2 + y$ を最大化せよ．

この問題を理解するために，領域 $x \geq 0,\ \mathrm{y} \geq 0$ で制約条件 $1 \leq x^2 + y^2 \leq 2$（円環領域）を描き，その中で $f(x, y) = x^2 + y = c$（放物線）を，定数 c を変えながら描いてみましょう（図 7.4）．

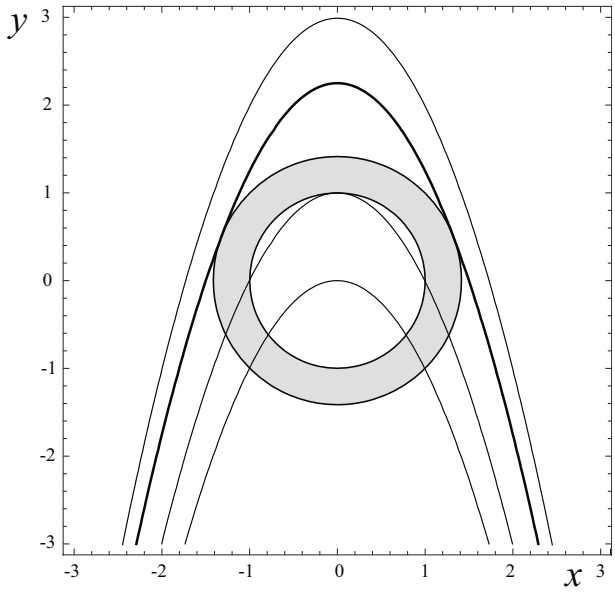


図 7.4: 非線形計画問題の例．グレー部分が制約条件で指定された領域で，目的関数の等高線 $f(x,y)=a$ は放物線である（$a=3,\ 9/4,\ 1,\ 0$）．答は図から読み取れるように $x=\sqrt{7}/2,\ \ y=1/2$（太い放物線 $(a=9/4)$ と外円との接点）のときに $f(\sqrt{7}/2,1/2)=9/4$ である．

7.3.2　MATLAB の利用

https://jp.mathworks.com/help/optim/ug/nonlinear-equality-and-inequality-constraints.html　を見てください．

手順 1.　（右上） ファイル objfun.m を定義します．目的関数の定義です．問題では最大化ですから,「最小化」に対応するため符号を反対にして定義しています．x と y が縦行列 x の第 1，第 2 成分になります．

手順 2.　（右上の下半分）ファイル confun.m を定義します．非線形制約条件の定義です．$x(1)\text{\^{}}2+x(2)\text{\^{}}2-2$，$-x(1)\text{\^{}}2-x(2)\text{\^{}}2+1$，$-x(1)$，$-x(2)$ のどれもが負という条件です．

手順 3.　（左）制約付き最適化ルーチンを呼び出すメイン・パートです．$x0$ は x(1), x(2) の試行初期値です．

```
%Main Program   NonLinearProgramProb2
x0 = [0.1,0.1];    % Make a starting guess at the solution
options = optimoptions(@fmincon,'Algorithm','sqp');
%options = optimoptions(SolverName,Name,Value) は、
%名前付きパラメーターSolverNameを指定値Valueにより変更し、optionsを返す.
[x,fval] = ...
fmincon(@objfun,x0,[],[],[],[],[],[],@confun,options);
%行末の 3 つ以上のピリオドは、現在のコマンドを次の行
% = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
%に続けるために使用.
x
fval
Sol=fval;
```

```
%Program    objfun.m
% Step 1
function f = objfun(x)
f = -x(1)^2 -x(2) ;

%Program
%Step 2
function [c, ceq] = confun(x)
% Nonlinear inequality constraints
c = [x(1)^2+x(2)^2-2 ; -x(1)^2-x(2)^2+1 ; -x(1); -x(2)];
ceq = [];
```

```
>> NonLinearProgramProb2
<stopping criteria details>
x =
    1.3229    0.5000
fval =
   -2.2500
```

図 7.5: 非線形計画問題の例. main プログラム（左），objfun.m と confun.m の定義（右上），結果（右下）.

出力 （右下）$x = 1.3229\,(=\sqrt{7}/2)$, $y = 0.5000\,(=1/2)$ のとき，最低値 $-2.2500\,(=-9/4)$ を与えます.

第8章　統計

ここでは，MATLAB の Statistics and Machine Learning Toolbox を使用して，統計データの扱いを説明します．

8.1　データの入力と表示

8.1.1　データの形

まず最初に我々が思いつくのは，データが Excel 上に用意されている場合でしょう．
エクセルデータ Jap-Math-Sci.xslx に図 8.1のようなデータが用意されているとします．使用するのはこの内の，第 3 列 number（個体番号），第 4 列 sex（男女の別），第 5 列 JapA（国語得点），第 7 列 MathA（数学得点）です．[1]
ここでは説明しませんが，カンマ付きデータ（csv ファイル）の場合には readmatrix(filename) を用います．

		number	sex	JapA	JapB	MathA	MathB	SciA	SciB	SciC
2015	3	1	2	27	7	33	9	16	6	10
2015	1	2	2	26	4	20	5	7	3	4
2015	1	3	2	27	7	29	12	19	6	13
2015	2	4	1	22	7	14	2	17	5	12
2015	1	5	2	27	8	31	8	16	6	10

図 8.1:　５０行の入力データの最初の数行．

[1] このデータは文科省が公表した全国学力・学習状況調査（全国学力テスト）のパブリックユースデータ（擬似データ）の一部にさらに手を加えた架空データである．

8.1.2 分布図および散布図

データの分布図あるいは散布図を上手に使えば，データの中身の概要を把握することができます.

8.1.2.1 分布図

histogram コマンドを用いれば，データの分布をヒストグラムで表すことができます.

```
ds=xlsread('Jap-Math-Sci.xlsx');
histogram(ds(:,5));%Score of Jap.
hold on
histogram(ds(:,7));%Score of Math.
hold off
xlabel('Score')
ylabel('Number of Students')
legend('Mathematics','Japanese')
```

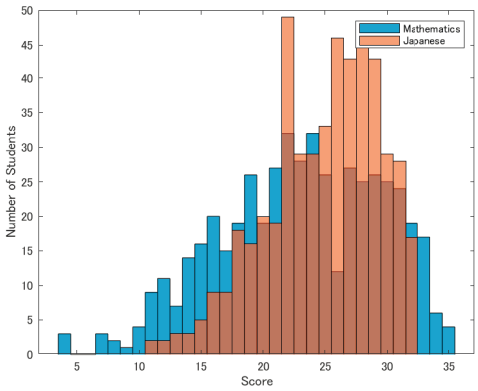


図 8.2: データ分布図.

- ds=xlsread(‘filename’)　では，Excel ファイル filename の数値データのみ，読み込みます．数値でないものも読みたいときには [ds, headertext] = xlsread(‘filename’) とします.
- ここではヒストグラムを描くコマンド histogram を 2 回呼んで，データ MathA と JapA の分布を重ねることにより，分布の違いなどが分かります.
- hold on を挟まないと次の histogram 命令で前の図が消えてしまいます.

8.1.2.2　散布図

サンプルデータから，第 1 パラメタを x 軸とし，第 2 パラメタを y 軸とし散布図を描くコマンド

scatter(x,y)

が用意されています．また，サンプルの点の属性（group）を，色で区別して表すために

gscatter(x,y,group)

があります（図 8.3）．

```
ds=xlsread('Jap-Math-Sci.xlsx');
gscatter(ds(:,3),ds(:,5),ds(:,4),...
  'br','.',20,'off','Number','Score')
hold on
gscatter(ds(:,3),ds(:,7),ds(:,4),'br','x',10,'off')
hold off
legend('Location','northeastoutside')
legend('Japanese 1','Japanese 2',...
  'Mathematics 1','Mathematics 2')
```

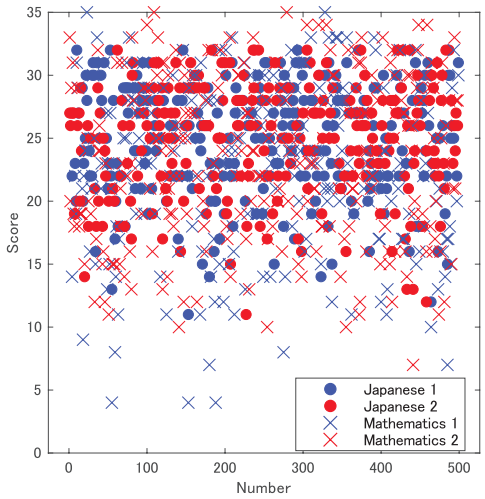


図 8.3: データ散布図.

ここでは男女の別を違う色で示したいので，gscatter を使いました．

- gscatter(x,y,group) は group 内のデータに従ってグループ分けされた（x,y）データの散布図を描きます．ここでは ds.sex の中の第 4 列のデータ 1，2（青、赤）に対応します．
- legend によってサンプルデータの説明が与えられます．
- 「...」は改行です．命令が横に長すぎるとき，これで改行・継続を示します．

```
ds=dataset('xlsfile', 'Jap-Math-Sci.xlsx');
mMathA=mean(ds.MathA)
sigMathA=std(ds.MathA)
mJapA=mean(ds.JapA)
sigJapA=std(ds.JapA)
cor=corrcoef(ds.MathA,ds.JapA)
```

```
>> Stat
mMathA =
  23.0100
sigMathA =
   6.4812
mJapA =
  24.8377
sigJapA =
   4.4945
cor =
   1.0000   0.6607
   0.6607   1.0000
```

図 8.4: 平均 mean ，標準偏差 std および相関係数行列 corrcoef. 左：スクリプト，右：結果

8.2 平均，分散，相関

与えられたデータの平均値，標準偏差（分散の平方根）および相関係数の計算は簡単です（図 8.4）．ここで xlsread の代わりに使った dataset コマンドは，‘将来サポートされなくなる’ とアナウンスが出ていますので注意してください．

8.3 回帰直線

相関係数を見ると，例えばここで扱っている例では，MathA と JapA の間には 0.66 の相関があることを示しています．この相関は小さいのでしょうか，あるいは大きいのでしょうか．一般には，この数字はかなり強い相関があると認識されるでしょう．そのようなときには，MathA と JapA の値をそれぞれ x 軸，y 軸にとってデータの分布を見るとより良く分かり，役立ちます（図 8.5）．

- scatter で，横軸に MathA, 縦軸に JapA の数値をとり，個々のデータの散布図を作る．
- lsline は回帰直線（最小二乗法による直線）を引くコマンド．

```
ds=dataset('xlsfile', 'Jap-Math-Sci.xlsx');
scatter(ds.MathA,ds.JapA,'r','o','filled')
h=lsline;
xlabel('Mathematics')
ylabel('Japanese')
set(h,'linewidth',2,'color','bl');
polyfit(ds.MathA,ds.JapA,1)
```

```
ans =
    0.4581   14.2959
```

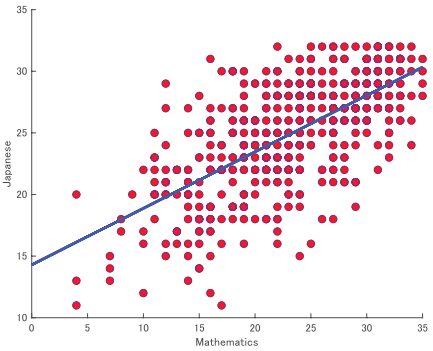


図 8.5: データの散布図と回帰直線.

- polyfit で，データを多項式で最小二乗フィットした式を求める．ここでは $n = 1$ として直線の当てはめをし，$\mathrm{JapA} = 0.4581 * \mathrm{MathA} + 14.2959$ を得た．
- 散布図からも，数学の成績の良い生徒は国語の成績も良い傾向が見られる．このことを定量的に表すのが上の直線の方程式である．さらには，数学の成績が 0 であっても，国語の成績は 0 ではないだろうことが見てとれる．

第9章　微分方程式

9.1　常微分方程式の解法

9.1.1　常微分方程式の初期条件と解

微分方程式

$$y'' + 4y' + 4y = 0$$

を初期条件

$$y(0) = 0.5, \quad y' = 4.0$$

の下で解いて，その振る舞いを図に示しましょう（図 9.1）.

```
syms y(x);Dy=diff(y)
eqn=diff(y,x,2)+4*diff(y,x,1)+4*y
cond1=y(0)==0.5;cond2=Dy(0)==4.0
conds=[cond1 cond2]
ySol(x)=dsolve(eqn,conds)
x=0:0.05:5
plot(x,ySol(x))
xlabel('x')
ylabel('y(x)')
title('Plot of D2(y)+4D(y)+4y,...
y(0)=0.5, D(y(0))=4.0')
```

```
ySol(x) =
exp(-2*x)/2 + 5*x*exp(-2*x)
```

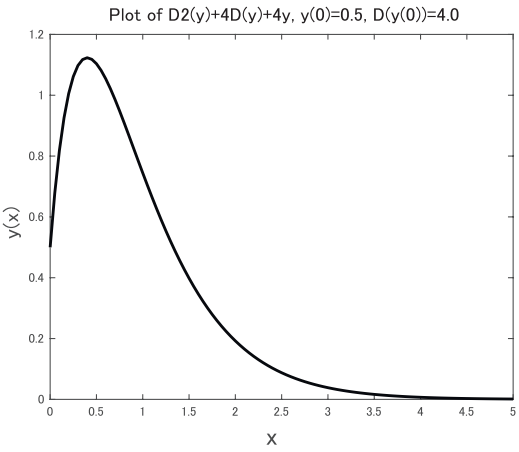


図 9.1: 常微分方程式.

- syms で y, Dy を定義（シンボリック変数）.
- eqn= $\cdots$ で微分方程式を定義.

- 恒等式 cond1, cond2 を定義し，その二つを合わせて conds= $\cdots$ で条件式として定義.
- dsolve(eqn,conds) は,「条件 conds の下で微分方程式 eqn= 0 を解け」，というコマンド．解として　$y\mathrm{Sol}(x) = \exp(-2 * x)/2 + 5 * x * \exp(-2 * x)$ を返す（$y(x) = \frac{1}{2}\mathrm{e}^{-2\mathrm{x}} + 5\mathrm{xe}^{-2\mathrm{x}}$ が初期条件を満足する解である）.
- x=0:0.05:5　は 0 から 5 まで 0.05 の刻み間隔で数ベクトル成分を定義.

9.1.2　常微分方程式の数値解法

同じ問題

$$y'' + 4y' + 4y = 0$$

を，4 次の Runge − Kutta 法を基礎にした ODE45 ソルバーを用いて，数値的に解きましょう．ODE は 1 階微分方程式の数値ソルバーです（図 9.2）．ODE では微分方程式を *m ファイルで定義し，それをメインのスクリプトで読み込みます.

```
[t,y] = ode45(@deq1,[0 5],[0.5, 4]);
plot(t,y(:,1),'-o',t,y(:,2),'-o')
xlabel('Time t');
ylabel('Solution y');
legend('y_1','y_2')
xlabel('x')
ylabel('y(x)')
title('Plot of D2(y)+4D(y)+4y,y(0)=0.5,
D(y(0))=4.0')
```

```
function dydt = deq1(t,y)
%function of DiffEqSymbEx1p.m
dydt = [y(2); -4*y(2)-4*y(1)];
```

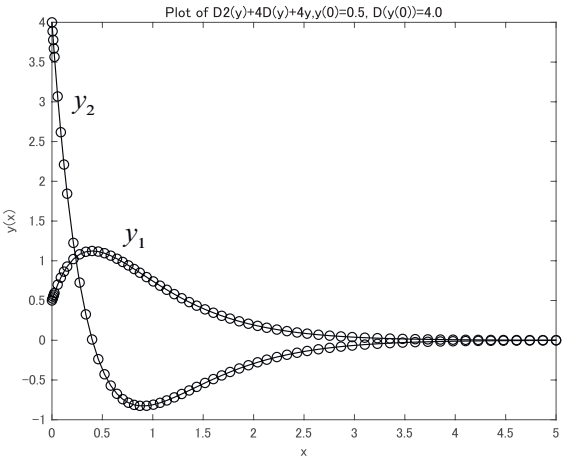


図 9.2: 常微分方程式の数値解法（ode45 の利用）．左上：メインスクリプト．左下：deq1.m ファイル.

まず，高階微分方程式を連立 1 階微分方程式に書き換えます：

$$\begin{aligned} y_1' &= y_2 \ , \\ y_2' &= -4y_2 - 4y_1 \ . \end{aligned}$$

- ここでは微分方程式を定義するプログラムは deq1.m と名付けられている（図 9.2左下）.
- ベクトル成分 $y(:,1)$ が y，$y(:,2)$ が y' となる.
- 初期値は ODE45 のパラメータ $[0.5, 4]$ で与える.

9.2 非正規型の微分方程式

9.2.1 非正規型の微分方程式の特異解

微分方程式 $y' = f(x,y)$ において，$f(x,y)$ が x, y の ‘滑らかな 1 価関数’ でないとき，これを非正規型と呼びます．非正規型の場合，解の一意性条件が壊れて，一般解に含まれない解（特異解）が現れることがあります.

```
syms y(x);Dy=diff(y)
eqn=diff(y,x,1)-3*y^(2/3)
ySol(x)=dsolve(eqn)
```

```
ySol(x) =
 0
 (C4 + 3*x)^3/27
```

```
x=-3:0.05:3;
  for c=-1.5:0.2:1.5
    yc=(x-c).^3
    plot(x,yc)
    hold on
  end
y0=0*x;
plot(x,y0);
hold on
ylim([-20 20])
xlabel('x')
ylabel('y(x)')
title('Plot of D2(y)+4D(y)+4y, y(0)=0.5, D(y(0))=4.0')
```

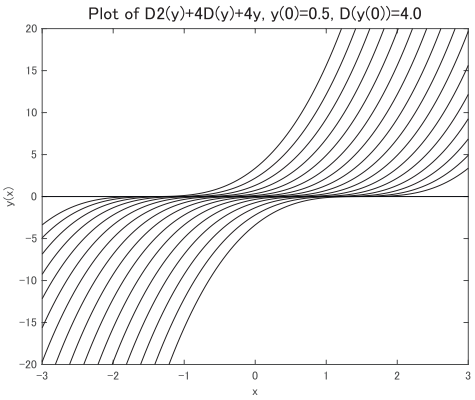


図 9.3: 非正規微分方程式.

微分方程式

$$y' = 3y^{2/3}$$

を解いてみましょう（図 9.3）．$y^{2/3}$ は y について多価です．

- 図右上が 2 種類の解 ySol(x) を返したところ．
- 左下は図を描く部分．一般解 $y(x) = (x - c)^3$ を $c = -1.5$ から 1.5 までの範囲で 0.2 刻みで変化させた．
- 図でも分かるように，一般解の**包絡線**がもう 1 つの解 $y = 0$．これを特異解という．

9.2.2　クレーローの方程式

非正規型微分方程式として有名なもの，クレーローの常微分方程式

$$y = y'x + \frac{1}{4}(y')^2$$

を解いてみましょう．図 9.4で，右上が返ってきた解．解は 2 種類のもの

```
syms y(x);Dy=diff(y)
eqn=y(x)-Dy*x-Dy^2/4
ySol(x)=dsolve(eqn)
```

```
ySol(x) =
       -x^2
 C4^2/4 + x*C4
```

```
x=-1.5:0.05:1.5;
for c=-3:0.2:3
 yc=c.*x+c.^2/4;
 plot(x,yc);
 hold on
end
ylim manual
y0=-x.^2;
plot(x,y0);
hold on
ylim([-2 2])
xlabel('x')
ylabel('y')
```

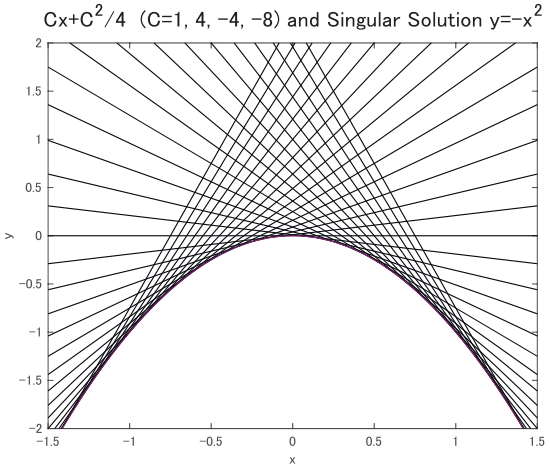


図 9.4: クレーローの方程式．

が与えられています．左下は図を描く部分．一般解 $y(x) = cx + (1/4)c^2$ を $c = -3$ から 3 までの範囲で 0.2 刻みで変化させた直線群が示されています．この直線群の包絡線が $y = -x^2$ であり，特異解です．

第10章　フーリエ級数展開

10.1　フーリエ級数

複雑な変化をする量は，単純な成分に分けて考えるのが鉄則です．信号解析その他で広く用いられるフーリエ級数展開を考えることにしましょう．

10.1.1　フーリエの方法

$[-a, a]$ を周期とする関数 $f(x)$ を 3 角関数の級数

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos\frac{n\pi x}{a} + b_n \sin\frac{n\pi x}{a}))$$

と表すとき，これを 「フーリエ級数展開」といい，展開係数 a_n, b_n は次の様に計算できます．

$$a_n = \frac{1}{a}\int_{-a}^{a} f(x)\cos\frac{n\pi x}{a}dx \quad , \quad n = 0, 1, 2, \ldots$$
$$b_n = \frac{1}{a}\int_{-a}^{a} f(x)\sin\frac{n\pi x}{a}dx \quad , \quad n = 1, 2, 3 \ldots .$$

10.1.2　簡単な例題

次の関数を考えましょう．

$$f(x) = \begin{cases} -1 & : \ -\pi < x \leq 0 \\ 1 & : \ 0 < x < \pi \end{cases}$$

この関数は、x の奇関数ですから sin 成分だけが現れます：

$$a_n = \frac{1}{\pi}\Big[\int_0^{\pi}\cos nxdx - \int_{-\pi}^{0}\cos nxdx\Big] = 0, \quad n = 0, 1, 2 \ldots$$
$$b_n = \frac{2}{\pi}\int_0^{\pi}\sin nxdx = \frac{2}{\pi}\frac{1-(-1)^n}{n}, \quad n = 1, 2, \ldots \quad .$$

したがって上の関数のフーリエ級数展開は

$$f(x) \sim S(x) = \frac{4}{\pi} \sum_{n=0}^{\infty} \frac{\sin(2n+1)x}{2n+1}$$

となります.
ここで、右辺で $x = 0$ とすると 0 となりますが，一方で $f(0) = -1$ ですから等号は成り立ちません：

$$S(0) = \frac{1}{2}\{f(0_+) + f(0_-)\} \neq f(0) \ .$$

このことがあるので $f(x) = S(x) = \cdots$ とは書かず，$f(x) \sim S(x) = \cdots$ と書きました.
一般にフーリエ級数 $S(x)$ は，$x = c$ が $f(x)$ の連続点であるなら正しく $f(c)$ を与え，また c が不連続なら左右から c に近づいた値の平均値

$$S(c) = \frac{1}{2}\{f(c+0) + f(c-0)\}$$

を与えることが知られています．これを**ディリクレ (Dirichlet) の定理**といいます.

10.1.3　MATLAB の適用

```
t=[-pi:0.01:pi];
  x=zeros(1,numel(t));
  for  n = 1:2:3
     x=x+ sin(n*t)/n*4/pi;
  end
  plot(t,x);grid on
  hold on
  for  n = 5:2:10
     x=x+ sin(n*t)/n*4/pi;
  end
  plot(t,x);grid on
  hold on
  for  n = 11:2:60
     x=x+ sin(n*t)/n*4/pi;
  end
  plot(t,x);grid on
   hold on
```

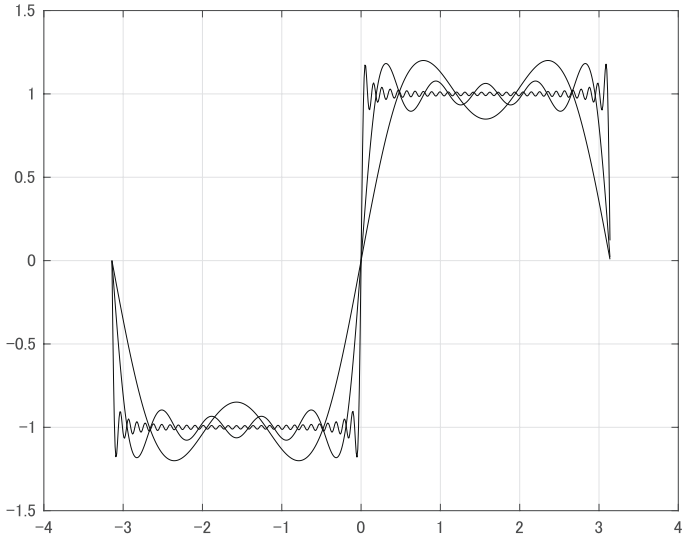


図 10.1: 部分和 $S_N(x) = \frac{4}{\pi}\sum_{n=1}^{N} \frac{\sin(2n+1)x}{2n+1}$ を N を変えて示す.

図 10.1に部分和

$$S_N(x) = \frac{4}{\pi} \sum_{n=1}^{N} \frac{\sin(2n+1)x}{2n+1}$$

示します.
$x = 0$ および $\pm\pi$ 近傍での振る舞いに注意してください. これについては詳しい説明に踏み込みませんが.

第Ⅴ部

数学上級編

第11章　線形代数：上級編

第3章（初級編）では，連立方程式を解くとき逆行列の計算を経由するのは実際的でなく，避けなくてはいけないと述べました．
計算の精度や効率を考えるのは，純粋の数学とは離れた議論のように感じるかも知れませんが，決してそうではありません．特に計算を主題とするならなおさらです．計算は有限の資源（時間や手間，計算機の規模）に依存しており，計算アルゴリズムの適切な選択によって，不可能な計算が実行可能な計算に変わることはよくあります．

11.1　連立方程式の解法

11.1.1　解法の復習

逆行列の計算あるいは連立方程式の計算，固有値・固有ベクトルの計算は，勿論多くの共通性があります．ここでは連立方程式

$$A\boldsymbol{x} = \boldsymbol{b}$$

を解くことを考えてみましょう．A は $n \times n$ 行列，$\boldsymbol{x}, \boldsymbol{b}$ は n 個の成分からなる列ベクトルです．

MATLAB ではこの演算は，A を $n \times n$ 行列, b を n 個の成分からなる列ベクトルと定義した後,

$$x = A\backslash b$$

と書きます.（MATLAB を含む）実用的なプログラムでは，逆行列を計算して b にかけるというアルゴリズムはとりません．MATLAB では、行列 A の特徴により，以下で述べる掃き出し法，LU 分解あるいは他の方法を自動的に選択します.
http://matlab.izmiran.ru/help/techdoc/ref/mldivide.html

（MATLAB の線形代数関数および行列演算は，Fortran サブルーチン・ライブラリとして評価の高い LAPACK を基に構築されています.）

11.1.2　行列の分解

11.1.2.1　ガウス - ジョルダンの掃き出し法

通常，教科書などで教えるのはガウス - ジョルダンの「掃き出し法」と呼ばれる次の手順です：

- [**Step1**]　$n \times n$ 行列 A および列ベクトル $\boldsymbol{b}$ を

  $$[A\ \boldsymbol{b}]$$

  と並べて n 行 $n+1$ 列（$n \times (n+1)$) の行列を作る.

- [**Step2**] 「基本行演算（行の定数倍，加減のみからなる基本変形演算）」を繰り返し，$[A\ \boldsymbol{b}]$ を

  $$[E\ \boldsymbol{x}]$$

  となるようにする．E は $n \times n$ の単位行列である．$\boldsymbol{b}$ があったところに答えである列ベクトル $\boldsymbol{x}$ が現れる．ただし基本行演算により成分の順番が入れ替わっていることがある.

- [**計算量**] 上の計算に必要な演算回数は乗除算が n^3 のオーダーとなる．掃き出し法では，連立方程式の大きさ n が例えば 10 倍になれば，演算回数は 1000 倍になり，すぐに現実的な時間内で結果を得ることができなくなる.

図 11.1に掃き出し法で逆行列を計算した結果を示しておきましょう．行列 A と単位行列を並べて，基本行操作を行ったものです．コマンド rref を用います．結果では，単位行列の所に逆行列が現れます．その確認のための計算が右に示してあります.

```
B=[2 1 0 1 0 0;1 2 1 0 1 0;0 1 2 0 0 1]
B =
   2   1   0   1   0   0
   1   2   1   0   1   0
   0   1   2   0   0   1
>> R=rref(B)
R =
  1.0000    0         0    0.7500   -0.5000    0.2500
       0    1.0000    0   -0.5000    1.0000   -0.5000
       0         0    1.0000    0.2500   -0.5000    0.7500
```

```
>>  A=[2 1 0;1 2 1;0 1 2]
A =
   2   1   0
   1   2   1
   0   1   2
>> inv(A)
ans =
   0.7500   -0.5000    0.2500
  -0.5000    1.0000   -0.5000
   0.2500   -0.5000    0.7500
```

図 11.1: ガウス - ジョルダンの掃き出し法

11.1.2.2 LU 分解

A を，下三角行列 L と上三角行列 U の積

$$A = LU = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

に書き換える LU 分解という方法があります．この方法は実際によく使われます．
演算の数は，A が密行列（要素の大部分が 0 ではない行列）である場合には，ガウス-ジョルダンの掃き出し法と比べて大きく異なることはありません．A が疎（要素の大部分が 0 である行列）である大行列の場合にも，LU 分解したときの L および U はやはり疎であるという性質を保ちます．そのため A が疎である場合には計算量に大きな差 (演算回数が n^2 程度の量になることもある）が生じます．
LU 分解の例を図 11.2示します．LU 分解を行うために，コマンド lu が用意されています．[L,U,P] = lu(A) は、L*U = P*A となる上三角行列 U，単位対角をもつ下三角行列 L，および置換行列 P を返します．

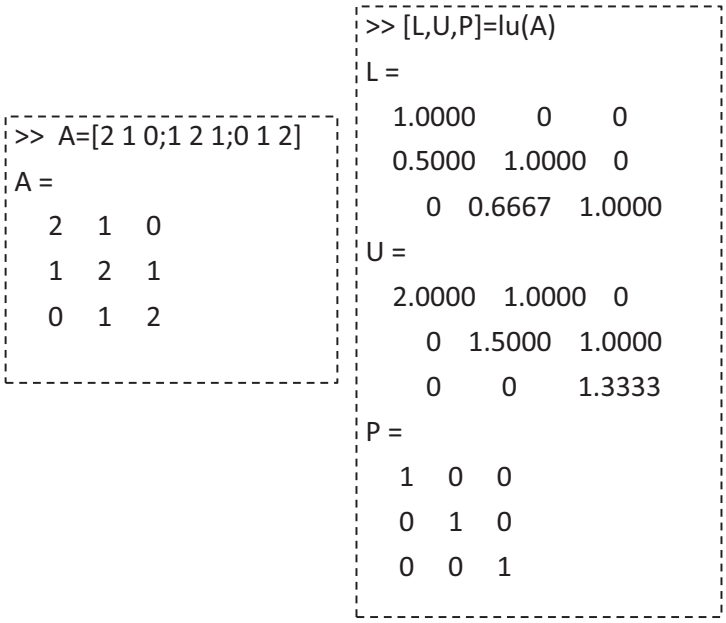


図 11.2: LU 分解

11.1.2.3　コレスキー（Cholesky）分解

行列 A が正定値エルミートであるとき，行列 A を上三角行列 U と U のエルミート共役 U^* との積に分解することができます．これをコレスキー分解と呼び，やはりよく用いられます．A のエルミート性を利用した LU 分解の特別な場合です．これのためにも U=chol(A) または U=chol(A,'upper') というコマンドが用意されています．

$$\mathrm{A} = \mathrm{U}' * \mathrm{U}$$

(U' は U の転置．　数学の記号なら t または T を右または左の上付きとします.）また下三角行列で求めるときは L=chol(A,'lower') とします．：

$$\mathrm{A} = \mathrm{L} * \mathrm{L}'$$

11.1.2.4　シュール（Schur）分解

任意の n 次正方行列 A に対しては次の性質が成り立ちます．

$$A = UTU^* \quad .$$

```
>> A=[1 2 3;4 5 1;2 3 4]
A =
   1   2   3
   4   5   1
   2   3   4
>> rank(A)
ans =
   3
>> eig(A)
ans =
   8.5233
  -0.3255
   1.8022
```

```
>> A=[1 2 3;4 5 1;2 3 4]
A =
   1   2   3
   4   5   1
   2   3   4
>> [U,T]=schur(A)
U =
   0.4233   0.8063  -0.4131
   0.6575  -0.5871  -0.4722
   0.6233   0.0717   0.7787
T =
   8.5233   0.2591  -1.1689
        0  -0.3255   2.7507
        0        0   1.8022
```

図 11.3: シュール分解とその性質.

U はユニタリ行列, T は上三角行列で, その対角成分は A の固有値 $(\lambda_1, \lambda_2, \cdots,)$ が並びます. U^* は U のエルミート共役です. シュール分解のコマンドは schur です（図 11.3）.

11.1.2.5 正規行列のシュール（Schur）分解

A が正規行列（$A^*A = AA^*$) である場合には, シュール分解は以下のようになります（図 11.4）.

$$A = UDU^* \quad .$$

U は直交行列, D は対角に A の固有値が並んだ対角行列です. U の第 j 列は, A の固有値 λ_j に対応した固有ベクトルです.

11.2 特異値分解

実際の連立方程式の場合には, 常に解が得られるとは限りません. 行列 A が正方行列でない場合, あるいは正方行列であっても正則でない場合

```
>> A=[1 3;3 1]
A =
    1    3
    3    1
>> [U,D]=schur(A)
U =
   -0.7071    0.7071
    0.7071    0.7071
```

```
D =
   -2    0
    0    4
>> U*D*U'
ans =
    1.0000    3.0000
    3.0000    1.0000
```

```
>> eig(A)
ans =
   -2
    4
>> U'*U
ans =
    1.0000    0.0000
    0.0000    1.0000
```

図 11.4: 正規行列のシュール分解

などです．より厄介な問題は，A のいくつかの固有値が他と比べて非常に小さくなっていて，ほとんど正則でない状態にある場合です．このようなときは，固有値問題あるいはシュール分解と深い関係にある以下の性質が大変有用です．

11.2.1　特異値分解の概要

11.2.1.1　特異値分解定理

A を階数 r の $m \times n$ 複素行列とします．このとき

$$A = U\Sigma V^*$$

という A の分解が存在します．ここで U は $m \times m$ のユニタリ行列，V^* は $n \times n$ のユニタリ行列 V のエルミート共役です．Σ は $m \times n$ 行列で，下のような形になります：

$$\Sigma = \begin{pmatrix} \mathrm{diag}(\sigma_1, \cdots \sigma_r) & 0_{r,n-r} \\ 0_{m-r,r} & 0_{m-r,n-r} \end{pmatrix}$$

これを複素行列 A の特異値分解と呼び，Σ の (i,i) 要素 σ_i を A の**特異値**と呼びます．ここで，$\mathrm{diag}(\sigma_1, \cdots \sigma_r)$ は $\sigma_1, \cdots \sigma_r$ を対角成分とする $r \times r$ の対角行列であり，$O_{l,k}$ は $l \times k$ のゼロ行列です．

```
>> A=[1 2 3 4;5 6 7 8]
A =
   1   2   3   4
   5   6   7   8
>> [U S V]=svd(A)
U =
  -0.3762   0.9266
  -0.9266  -0.3762
S =
  14.2274   0        0        0
      0     1.2573   0        0
V =
  -0.3521  -0.7590  -0.4001  -0.3741
  -0.4436  -0.3212   0.2546   0.7970
  -0.5352   0.1165   0.6910  -0.4717
  -0.6268   0.5542  -0.5455   0.0488
>> U*S*V'
ans =
   1.0000   2.0000   3.0000   4.0000
   5.0000   6.0000   7.0000   8.0000
```

図 11.5: 特異値分解の例.

11.2.1.2 特異値分解の例

[U,S,V] = svd(A) は，$A = USV^*$ となるように，行列 A の特異値分解を行います（図 11.5）.

11.2.2 疑似逆行列

これまでの多くの場合では，行列 A の逆行列が存在していると仮定し，それをいかに効率よく求められるか，ということを考えました．また逆行列が存在しない場合にも行列の分割という手段で対応しました．ここでは，逆行列が存在しない場合にも，逆行列に代わる概念である疑似逆行列という考え方を紹介しましょう（ムーア-ペンローズの擬似逆行列）.
A の疑似逆行列 A^+ は一意に決まり，以下の性質を満たします.
(1) $AA^+A = A$
(2) $A^+AA^+ = A^+$
(3) $(AA^+)^* = AA^+$，すなわち AA^+ はエルミート行列.
(4) $(A^+A)^* = A^+A$，すなわち A^+A はエルミート行列.
この性質から A が正則な場合には，$A^+ = A^{-1}$ であることが分かります.
A^+ を A の疑似逆行列と呼びます.

11.2.2.1　疑似逆行列の具体的な形

- $n > m = r$ の場合

$$A = U\Sigma V^*$$

としてもう少し踏み込んでみましょう.

$$\Sigma = (\mathrm{diag}(\sigma 1, \cdots, \sigma_m) \;\; 0_{m,n-m})$$

です．また

$$\Sigma^+ = \begin{pmatrix} \mathrm{diag}(1/\sigma_1, \cdots, 1/\sigma_m^{-1}) \\ 0_{n-m,m} \end{pmatrix}$$

とすると A^+ は

$$A^+ = V\Sigma^+ U^*$$

です．実際これが疑似逆行列の持つべき性質を満たすことは簡単に確かめることができます.

- $m > n = r$ の場合にも同じように書かれます．ただしこの場合には

$$\Sigma = \begin{pmatrix} \mathrm{diag}(\sigma_1, \cdots \sigma_n) \\ 0_{n-m,m} \end{pmatrix}$$

とします.

ユニタリ行列 U および V を，正規直交（列）ベクトル $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m$

$$\boldsymbol{u}_i^* \cdot \boldsymbol{u}_j = \delta_{ij} \ , \ i, j = 1 \sim m$$

および正規直交（列）ベクトル $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n$

$$\boldsymbol{v}_i^* \cdot \boldsymbol{v}_j = \delta_{ij} \ , \ i, j = 1 \sim n$$

を用いて書き換えてみましょう：

$$U = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m)$$

および

$$V = (\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n) \ , \ V^* = \begin{pmatrix} \boldsymbol{v}_1^* \\ \vdots \\ \boldsymbol{v}_n^* \end{pmatrix}$$

```
C=[1 2;3 4]
C =
    1    2
    3    4
>> rank(C)
ans =
    2
>> D=pinv(C)
D=
   -2.0000    1.0000
    1.5000   -0.5000
>> inv(C)
ans =
   -2.0000    1.0000
    1.5000   -0.5000
```

```
>> A=[2 2;1 1]
A =
    2    2
    1    1
>> rank(A)
ans =
    1
>> B=pinv(A)
B =
    0.2000    0.1000
    0.2000    0.1000
>> A*B*A
ans =
    2.0000    2.0000
    1.0000    1.0000
>> B*A*B
ans =
    0.2000    0.1000
    0.2000    0.1000
```

```
>> (A*B)'
ans =
    0.8000    0.4000
    0.4000    0.2000
>> A*B
ans =
    0.8000    0.4000
    0.4000    0.2000

>> (B*A)'
ans =
    0.5000    0.5000
    0.5000    0.5000
>> B*A
ans =
    0.5000    0.5000
    0.5000    0.5000
```

図 11.6: 疑似逆行列の例.

と書くと（$\boldsymbol{v}_i^*$ は行ベクトル）

$$
\begin{aligned}
A &= U\Sigma V^* = \sum_{i=1} \sigma_i \boldsymbol{u}_i \ \boldsymbol{v}_i^* \\
A^+ &= V\Sigma^+ U^* = \sum_{i=1} \frac{1}{\sigma_i} \boldsymbol{v}_i \ \boldsymbol{u}_i^*
\end{aligned}
$$

となります $\boldsymbol{u}_i \ \boldsymbol{v}_i^*$ は $m \times n$ 行列です．これらの式から

$$
\begin{aligned}
A\boldsymbol{v}_i &= \sigma_i \boldsymbol{u}_i \\
A^+ \boldsymbol{u}_i &= \frac{1}{\sigma_i} \boldsymbol{v}_i
\end{aligned}
$$

も得られます．このように固有値問題が成り立たない場合にも，こので示したような一般化固有値問題が成り立ち，大変に有用です．
疑似逆行列の計算を図 11.6に示しましょう．B = pinv(A) は行列 A の Moore-Penrose 疑似逆行列を返します。

11.3　特異値分解の応用

ここでは簡単のために，A は $m \times n$ の実行列であり，その階数は r であるとします．このとき，特異値分解

$$A = U\Sigma V^T \quad .$$

が存在します．ただし U および V はそれぞれ，$m \times m$ および $n \times n$ の直交行列で，Σ は $m \times n$ 行列です．特異値 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ は Σ の対角に並びます．

11.3.1　最小二乗法

上の A に対して実ベクトル $\boldsymbol{b}$ が与えられたとします．このとき

$$||A\boldsymbol{x} - \boldsymbol{b}||^2 \equiv (A\boldsymbol{x} - \boldsymbol{b})^T (A\boldsymbol{x} - \boldsymbol{b})$$

を最小にするように $\boldsymbol{x}$ を決める問題を考えます．これを最小 2 乗問題といい，よく知られた最小 2 乗法と同等の問題です．またこの解を**最小 2 乗解**といいます．
今の場合，最小 2 乗問題の解 $\boldsymbol{x}'$ は

$$A^T A\boldsymbol{x}' = A^T\boldsymbol{b}$$

の解です．
連立方程式系 $A\boldsymbol{x} = \boldsymbol{b}$ が与えられたとき，この方程式系の解がない場合には，A の疑似逆行列 A^+ を用いて，$\boldsymbol{x}' = A^+\boldsymbol{b}$ が最適解になります．これは上の問題の最小 2 乗解でもあります．

11.3.1.1　簡単な例

$$A\boldsymbol{x} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \boldsymbol{b} \quad .$$

この方程式は解を持ちません．

これを $A^TA\boldsymbol{x} = A^T\boldsymbol{b}$ と変形すれば

$$A^TA\boldsymbol{x}' = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} = A^T\boldsymbol{b} \quad .$$

これを解いて

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

を得ます．これが最小 2 乗解です．
この計算を疑似逆行列を用いて解いた結果を，図 11.7に示しましょう．

```
>> A=[1 1;1 1]
A =
   1   1
   1   1
>> b=[3;1]
b =
   3
   1
```

```
>> rank(A)
ans =
   1
>> P = pinv(A)*b
P =
   1.0000
   1.0000
```

図 11.7: 連立方程式の最小 2 乗解

11.3.2 主成分分析 (Principal Component Analysis)

多くのデータが与えられ，それらが互いに相関のある多数の変数からなっているとき，互いに相関の無い少数の変数（**主成分**）で全体のデータを記述することが望まれます．そのような成分を探す方法の 1 つが**主成分分析**です．

11.3.2.1　データ行列の特異値分解

11.3.2.1.1　データの標準化

N 個のサンプルについて其々P 個の変数（データ）を持つデータ群が与えられていて，サンプル $i(i = 1 \sim N)$ についての生のデータを $\{t_{i\alpha}\}(\alpha = 1 \sim P)$ とします．このデータが，N 行 P 列の**データ行列** T にまとめられているとします：

$$T = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1P} \\ t_{21} & t_{22} & \cdots & t_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ t_{N1} & t_{N2} & \cdots & t_{NP} \end{pmatrix} .$$

一般にこれらの P 個の変数はそれぞれ単位も異なり，値やそのバラツキの大きさも異なります．そのため一般にデータの**標準化**という手法がとられます．すなわちデータの中心を 0 と定め，値をそれぞれの標準偏差で規格化します：

$$\bar{t}_s = \frac{1}{N}\sum_{m=1}^{N} t_{ms} \ , \quad \sigma_{t_s}^2 = \frac{1}{N-1}\sum_{m=1}^{N}(t_{ms} - \bar{t}_s)^2 \ ,$$
$$x_{ns} = \frac{t_{ns} - \bar{t}_s}{\sigma_{t_s}} \ , \quad \sum_{m=1}^{N} x_{ms} = 0 \ , \quad \sum_{m=1}^{N} x_{ms}^2 = N - 1 \ .$$

データの標準化により，**（標準化された）データ行列**は

$$X = \{x_{ij}\} \ , \quad i = 1 \sim N, \quad j = 1 \sim P$$

です．データの各要素 x_{mi} については

$$\text{平均値} \ : \ \frac{1}{N}\sum_{m=1}^{N} x_{ms} = 0$$
$$\text{標準偏差} \ : \ \frac{1}{N-1}\sum_{m=1}^{N} x_{ms}^2 = 1$$

となります．

11.3.2.1.2 データ行列と共分散行列

$$Q = \frac{1}{N-1} X^T X = \begin{pmatrix} \sigma_{x_1x_1} & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_P} \\ \sigma_{x_2x_1} & \sigma_{x_2x_2} & \cdots & \sigma_{x_2x_P} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{x_Px_1} & \sigma_{x_Px_2} & \cdots & \sigma_{x_Px_P} \end{pmatrix}$$

と定義される $P \times P$ 行列を**共分散行列**と呼びます．ただし

$$\sigma_{x_{s_1}x_{s_2}} = \frac{1}{N-1} \sum_{m=1}^{N} x_{ms_1} x_{ms_2} \ , \quad \sigma_{x_{s_1}x_{s_1}} = \sigma^2_{x_{s_1}} = 1 \ .$$

11.3.2.1.3 共分散行列の特異値分解

正規直交行列

$$V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_P) = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1P} \\ v_{21} & v_{22} & \cdots & v_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ v_{P1} & v_{P2} & \cdots & v_{PP} \end{pmatrix}$$

を用いて

$$Z = XV \quad , \quad (Z)_{m\alpha} = (XV)_{m\alpha} = \sum_{\beta=1}^{P} x_{m,\beta} v_{\beta,\alpha}$$

という変換を行いましょう．これは各データを特徴付ける $1 \sim P$ の変数の変数の線形結合で新しい変数を定義したことになります．
ここで V を

$$V^T Q V = V^T \ \frac{X^T X}{N-1} \ V = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_P^2 \end{pmatrix} = \Sigma$$

を満たすように決めるとすれば

$$QV = V\Sigma \quad , \quad Q\boldsymbol{v}_\alpha = \sigma_\alpha^2 \boldsymbol{v}_\alpha \ , \quad \sum_{\alpha=1}^{P} \sigma_\alpha^2 = P$$

となり，Q の固有値問題であることが分かります．つまり主成分分析ではまず共分散行列の固有空間への分解が行われます．
これまでのことを少し整理しましょう．

1. Q の固有値 σ_α^2 $(\alpha = 1 \sim P)$ を対角に並べた行列 $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_P)$ は共分散行列 Q を対角化（固有空間に分解）したものです．

2. 行列 $X^T X$ の固有値問題；

$$X^T X \boldsymbol{v}_\alpha = \lambda_\alpha^2 \boldsymbol{v}_\alpha \ ,$$

固有値 $\{\lambda_\alpha^2\}$ が大きな順に

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_P > 0$$

と並べられているとします．$(\lambda_\alpha, \boldsymbol{v}_\alpha)$ を**第** α **主成分**といいます．

3. 上の諸量をまとめて，

$$\begin{aligned} \Lambda &= \mathrm{diag}(\lambda_1, \cdots, \lambda_P) \\ \boldsymbol{u}_\alpha &= \frac{1}{\lambda_\alpha} X \boldsymbol{v}_\alpha \\ U &= (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_P) \end{aligned}$$

を定義します（$\lambda_\alpha^2 = (N-1)\sigma_\alpha^2$）．

$$\begin{aligned} X \boldsymbol{v}_\alpha &= \lambda_\alpha \boldsymbol{u}_\alpha \\ X^T \boldsymbol{u}_\alpha &= \lambda_\alpha \boldsymbol{v}_\alpha \end{aligned}$$

などは定義から直ぐに証明できます．

4. $\boldsymbol{v}_\alpha$ は正規直交ベクトルですから，$\boldsymbol{u}_\alpha$ も正規直交ベクトルです（$U^T U = E$：$P \times P$ 単位ベクトル）．

5. 上から以下の式が成り立つことが分かります：

$$\begin{aligned} XV &= (X\boldsymbol{v}_1, \cdots, X\boldsymbol{v}_P) = (\lambda_1 \boldsymbol{u}_1, \cdots, \lambda_P \boldsymbol{u}_P) \\ &= (\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots)\Lambda = U\Lambda \end{aligned}$$

これは

$$X = U\Lambda V^T$$

と書き直すことができ，X の特異値分解であることが分かります．

6. また行列 X^TX、XX^T のスペクトル分解；

$$X^TX = \sum_{\alpha=1}^{P} \lambda_\alpha^2 \boldsymbol{v}\boldsymbol{v}^T ,$$
$$XX^T = \sum_{\alpha=1}^{P} \lambda_\alpha^2 \boldsymbol{u}\boldsymbol{u}^T$$

が成り立ちます.

11.3.2.2 主成分分析の目的

与えられた N 個の各データが P 個の変数で特徴付けられています．X^TX の特異値を順次大きい順番で考えると，大きい固有値が行列 X^TX の特徴を決めることが分かります（スペクトル分解）．小さい固有値を無視すれば，少しの変数でデータの本質を捉えられると期待できます.

11.3.2.2.1 寄与率

以下に定義される**寄与率** は，得られた主成分がデータをどの程度説明しているかを示す尺度です.
第 α 固有値 (特異値) λ_α は各空間がどの程度 X^TX の性格を決めているかが分かりますから，寄与率として通常用いられるのはこれを規格化した

$$\frac{\lambda_\alpha}{\sum_{\beta=1}^{P} \lambda_\beta}$$

です．また，寄与率を第 1 主成分から順に累積していったもの（例えば α 主成分まで足したもの）を累積寄与率といいます：

$$\frac{\sum_{\beta=1}^{\alpha} \lambda_i}{\sum_{\gamma=1}^{P} \lambda_\gamma} .$$

11.3.2.2.2 因子負荷量

分散共分散行列の固有ベクトルから，例えば第 α 主成分の主軸方向の長さを計算できます．この長さを**主成分スコア**といいます．また固有ベクトルが因子負荷量として使われます.

11.3.2.3　主成分分析の例

MATLAB が提供しているデータ hald （セメントの発熱と混合成分）を例に主成分分析の演習をしましょう．13 種類のセメント組成に対して，ingredients 部分（4 つの列）には 4 種類のセメント原料の組成率が与えられています．この部分をデータ行列 T とします．

$$T = \begin{array}{rrrr} 7 & 26 & 6 & 60 \\ 1 & 29 & 15 & 52 \\ 11 & 56 & 8 & 20 \\ 11 & 31 & 8 & 47 \\ 7 & 52 & 6 & 33 \\ 11 & 55 & 9 & 22 \\ 3 & 71 & 17 & 6 \\ 1 & 31 & 22 & 44 \\ 2 & 54 & 18 & 22 \\ 21 & 47 & 4 & 26 \\ 1 & 40 & 23 & 34 \\ 11 & 66 & 9 & 12 \\ 10 & 68 & 8 & 12 \end{array}$$

- 各列について要素の平均値 ts および標準偏差 sigt を計算します．平均値は mean, 分散は var が与えます．

$$ts = \mathrm{mean}(T) \quad = \quad 7.4615 \quad 48.1538 \quad 11.7692 \quad 30.0000$$
$$sigts = \mathrm{sqrt}(\mathrm{var}(T)) \quad = \quad 5.8824 \quad 15.5609 \quad 6.4051 \quad 16.7382$$

- 用意されているコマンド zscore を使い，平均値，標準偏差，標準化されたデータ行列を計算します．

$$[X, ts, sigts] = \mathrm{zscore}(T)$$

  $ts, sigts, X$ はそれぞれ，もとのデータ行列の各列ベクトル成分に関する平均値 $\bar{t}_s$，標準偏差 σ_{t_s}，および標準化されたデータ行列 X です．mean, var を用いた計算と比較して確認しましょう．

- X の共分散行列 Q は（ここでは $N = 13$ ですから）MATLAB の言葉で書けば

$$Q = X' * X/12$$

  です．共分散行列の固有値 σ_α^2 を対角に並べた行列 $SIGMA2$，固有ベクトル（主成分 $\alpha = 1 \sim P$) を各列成分（もとの ingredients=$1 \sim P$ を行成分に）並べた行列 V を計算します：

$$[V, SIGMA2] = \mathrm{eig}(Q)\ .$$

$$V = \begin{matrix} 0.2411 & 0.6755 & 0.5090 & -0.4760 \\ 0.6418 & -0.3144 & -0.4139 & -0.5639 \\ 0.2685 & 0.6377 & -0.6050 & 0.3941 \\ 0.6767 & -0.1954 & 0.4512 & 0.5479 \end{matrix}$$

$$SIGMA2 = \begin{matrix} 0.0016 & 0 & 0 & 0 \\ 0 & 0.1866 & 0 & 0 \\ 0 & 0 & 1.5761 & 0 \\ 0 & 0 & 0 & 2.2357 \end{matrix}$$

$SIGMA2$ は Q の固有値行列ですから，主成分分散 σ_α^2 を対角に並べたものです．列は σ_α の昇順です.

さらに固有ベクトルをサンプルの成分を行成分（$j = 1 \sim N$) で書き表した行列 $Z = XV$ を計算してみてください.

- これまでの計算を特異値分解で計算します：

$$[U1, Lambda1, V1] = \mathrm{svd}(X) \ .$$

$$Lambda1 = \begin{matrix} 5.1796 & 0 & 0 & 0 \\ 0 & 4.3489 & 0 & 0 \\ 0 & 0 & 1.4964 & 0 \\ 0 & 0 & 0 & 0.1396 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

$$V1 = \begin{matrix} 0.4760 & -0.5090 & 0.6755 & 0.2411 \\ 0.5639 & 0.4139 & -0.3144 & 0.6418 \\ -0.3941 & 0.6050 & 0.6377 & 0.2685 \\ -0.5479 & -0.4512 & -0.1954 & 0.6767 \end{matrix}$$

ここの V は前項の V と同じもの（列の並べ方は降順），$Lambda1$ は X の特異値 λ_α を並べた行列です．$\lambda_\alpha^2 = (N-1)\sigma_\alpha^2$ より，各成分に対して $Lambda1 = \sqrt{12 \times SIGMA2}$ であることが確かめられます.

- これまでの計算（主成分分析）を一度に行うコマンドが pca です：

$$[COEFF, SCORE, LATENT] = \mathrm{pca}(X)\ .$$

これらの返り値の列は主成分分散 σ_α^2 の大きい方からの順（降順）に並びます．$COEFF$ は $N \times p$ (標準化された）データ行列 X の主成分係数 (負荷量）$V1$ を，$SCORE$ は主成分スコア $Z = X * V$ を，$LATENT$ は主成分分散 σ_α^2 （$SIGMA2$）を返します：

$$LATENT = \begin{matrix} 2.2357 \\ 1.5761 \\ 0.1866 \\ 0.0016 \end{matrix}$$

第12章　非線形微分方程式

12.1　相空間と安定性

捕食系の振る舞いをよく捉えたモデルとして，ボルテラ系（あるいはロトカ・ボルテラ方程式）があります:

$$\begin{aligned} \frac{dx}{dt} &= x - xy \\ \frac{dy}{dt} &= xy - y \end{aligned}$$

x, y はそれぞれ被捕食者、捕食者の個体数です.
これから t を消去すると

$$\frac{dx}{dy} = \frac{dx/dt}{dy/dt} = \frac{x - xy}{xy - y} = \frac{-1 + \frac{1}{y}}{1 - \frac{1}{x}}$$

となり

$$(x - \log x) + (y - \log y) = C$$

が得られます．解は，x-y 空間で

$$xy\ \mathrm{e}^{-x-y} = f = \text{一定}$$

の曲線上を動くことが分かります.
このような解析は重要であり，ここで x-y 空間を相空間と呼びます．相空間での振る舞いを知ることが，解の性質，特に解の安定性（充分時間がたったとき，解が有限の領域に留まらないとき，その解を ‘不安定’ といいます.）を知るために大変重要です．相空間内において，上で得られた解の運動を調べたものが図 12.1です．$f =$ 一定の等高線図の書き方が分かるでしょう.

```
[x,y]=meshgrid(0:0.01:3);f=x.*y.*exp(-x-y);
[C,h]=contour(x,y,f,5)
clabel(C,h,'manual')
hold on
%
[x,y]=meshgrid(-3:0.1:3);f=x.*y.*exp(-x-y);
[C,h]=contour(x,y,f,25)
clabel(C,h,'manual')
[C,h]=contour(x,y,f,500)
clabel(C,h,'manual')
hold on
```

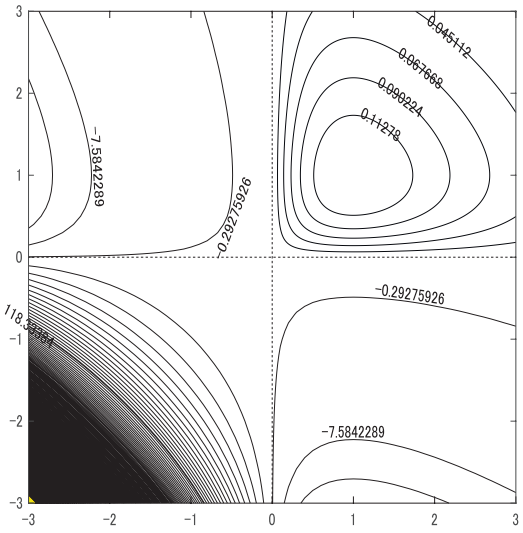


図 12.1: 相空間と解の安定性

- 図 12.1では，$x, y > 0$, $x, y < 0$，$xy < 0$ において，等高線の間隔（スケール）をそれぞれ変えてある．
- [x, y] = meshgrid(· · ·) では，2 次元空間 xx-y のメッシュを定義 ([始点：間隔：終点]).
- contour で等高線を表示する．最後のパラメータで等高線数を選択.
- clabel で等高線の値をラベルに与える．パラメータ 'manual' はラベル付けする等高線を手で選択することを指示.

12.2　非線形微分方程式の例

ボルテラ系のような非線形微分方程式を，MATLAB を用いて数値的に議論してみましょう.

12.2.1　ボルテラ系

前の図 12.1を数値的に取り扱ってみましょう（図 12.2）.

```
ode=@(t,y)dydt0(t,y); % Define the Ord. Diff.Eq.
[t,y]=ode45(ode,[0,20],[1.0,1.8]); % Solver ode45
subplot(2,1,1);plot(t,y(:,1));
hold on
plot(t,y(:,2));
xlabel('t');ylabel('solution x and y');
txt2=sprintf('%.6f',1.0);
txt3=sprintf('%.6f',1.8)
title(['Lotka-Volterra equation: Starting Pt y0=(',txt2,' , ',txt3,')'])
legend('x(t)','y(t)')
hold off
subplot(2,1,2);plot(y(:,1),y(:,2));
xlim([0 2.0]);ylim([0 2.0])
xlabel('x(t)');ylabel('y(t)');
title(['Lotka-Volterra equation : Starting Pt y0=(',txt2,' , ',txt3,')'])
hold on
```

```
function dydt0 = dydt0(t,y)
%Lotka?Volterra equation
% dy1/dt=y1-y1*y2
% dy2/dt=y1*y2-y2
dydt0 = [y(1)-y(1)*y(2);y(1)*y(2)-y(2)]
```

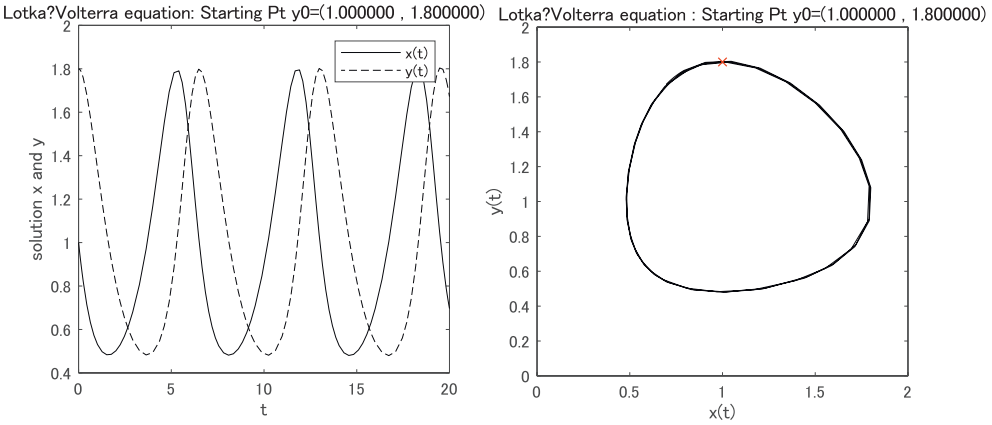


図 12.2: ボルテラ系（あるいはロトカ・ボルテラ方程式）

- 連立微分方程式（上段右に function dydt0 を定義）として関数を定義したのが 1 行目.
- 2 行目は微分方程式を解くスクリプトとして ODE45 を用いる．時間 t は (0,20) の範囲で変化させ，初期値は $x(0) = 1.0, y(0) = 1.8$ とする.
- １つの図に 2 つの図を収めることを，subplot コマンドで定義する．subplot$(2, 1, n)$ は，図を全体として 2 行 1 列に並べ，そのうちの n 番目の図として配する，という意味.

- txt2=sprintf(···), txt3=sprintf(···) で，図の説明に書き込む変数（初期値）を定義. sprintf はデータから書式を指定した文字列データに変換する.

- 第 1 の図は $x(t), y(t)$ を時間 t の関数として，第 2 の図は $(x(t), y(t))$ の軌道を表す．ここでは $x, y > 0$ の領域で，周期軌道（閉軌道）を表す．注意をして見れば，数値的誤差のために，軌道が完全には前の軌道をなぞっていないことが分かる.

- x（被捕食者=餌）の数が減少（増加）すると y（捕食者）の数が減少（増加）し出す．しばらくすると y の数が減った（増えた）ため，x（餌）の数が増加（減少）し出す．このようなサイクルが繰り返される.

第VI部

応用編

第13章　信号処理

13.1　フーリエ変換

フーリエ変換は連続変数による積分変換です．実際の多くの場合には，離散的変数値を用いた有限項の和に置きかえる必要があります．したがって，実際には離散変数のフーリエ変換，あるいは離散変数のフーリエ級数展開が必要です．

13.1.1　連続変数を離散変数に（離散フーリエ変換）

関数 $f(x)$ は区分的に滑らか（いくつかの孤立した点以外では滑らか）で連続であり，$[0, 2a]$ を基本の区間とし，周期 $2a$ の周期関数としましょう．この様な周期関数の複素フーリエ級数展開は

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} c_n e^{i\frac{n\pi x}{a}} \quad , \\ c_n &= \frac{1}{2a}\int_{-a}^{a} f(x) e^{-i\frac{n\pi x}{a}} dx \end{aligned}$$

と表されます．
区間 $[0, 2a]$ を

$$\begin{aligned} 0 &= x_0 < x_1 < x_2 < \ldots < x_{N-1} < x_N = 2a, \\ x_k &= \frac{2ak}{N} \quad (k = 0, 1, \ldots, N-1) \ , \quad \delta k = \frac{2a}{N} \end{aligned}$$

のように N 等分して，分点上の和によって c_n の積分を置き換えます：

$$c_n = \frac{1}{N}\sum_{k=0}^{N-1} f(\frac{2ak}{N}) e^{-i\frac{2nk\pi}{N}} \quad .$$

あるいはこれを

$$\begin{aligned} \omega &= \exp\frac{2\pi i}{N}, \\ f_k &= f(x_k), k = 0, 1, 2 \ldots, N-1 \\ c_n &= \frac{1}{N}\sum_{k=0}^{N-1} f_k(\bar{\omega})^{nk}, n = 0, 1, 2, \ldots, N-1 \end{aligned}$$

と書きかえます．$\{f_0, f_1, \ldots f_{N-1}\}$ から $\{c_0, c_1, \ldots, c_{N-1}\}$ への変換を **離散フーリエ変換** と呼びます．この逆変換 (**離散フーリエ逆変換**) は

$$f_k = \sum_{n=0}^{N-1} c_n \omega^{nk} \qquad (k = 0, 1, 2, \ldots, N-1)$$

となります．ω および c_n の定義により

$$\sum_{n=0}^{N-1} c_n \omega^{nk} = \frac{1}{N}\sum_{k'=0}^{N-1} f_{k'} \sum_{n=0}^{N-1} \omega^{n(k-k')} = \frac{1}{N}\sum_{\substack{k'=0 \\ (k'\neq k)}}^{N-1} f_{k'} \frac{1-\omega^{N(k-k')}}{1-\omega^{(k-k')}} + f_k$$

です．ω は 1 の N 乗根 ($\omega^N = 1$) ですから右辺第 1 項は 0 となります．これで $f_k = \cdots$ の式を直接たしかめることができました．
離散フーリエ (逆) 変換は直接この式に従って計算すると，n と k はともに 0 から $N-1$ まで動きますから，N^2 回に比例する乗・加算が必要です (正しくは乗算 N^2 回、加算 $N(N-2)$ 回)．このため N の増加にともなう計算の手間の増加は急激で，N が 300 程度になると実用上無視できない問題となります．効率よく計算を行うという点から，高速フーリエ変換 (FFT=Fast Fourier Transform) という重要かつ有効な計算アルゴリズムが発明されました．
MATLAB では離散フーリエ変換は，高速フーリエ変換を用いて行います．に具体例を示しましょう．

13.1.2 高速フーリエ変換：MATLAB を用いて

$Y = \mathrm{fft}(X)$ は高速フーリエ変換 (FFT) アルゴリズムを使用して，X の離散フーリエ変換 (DFT) を計算します．
ここでは，信号をフーリエ成分に分解してみましょう（図 13.1）．

```
Fs = 1000;          % Sampling frequency
T = 1/Fs;           % Sampling period
L = 1500;           % Length of signal
t = (0:L-1)*T;      % Time
Signal = 0.5*sin(2*pi*50*t) + 1.0*sin(2*pi*120*t);
Noise  = 2.0*randn(size(t)); X = Signal+ Noise;
subplot(2,1,1);plot(1000*t(1:50),Signal(1:50)) ; hold on
subplot(2,1,1);plot(1000*t(1:50),X(1:50))
title('Signal + Random Noise')
xlabel('t');ylabel('X(t)')
%
Y = fft(X);
P2 = abs(Y/L);P1 = P2(1:L/2+1);
f = Fs*(0:(L/2))/L;
subplot(2,1,2);plot(f,P1)
title('One-Sided Power Spectrum')
xlabel('f');ylabel('|P1(f)|')
```

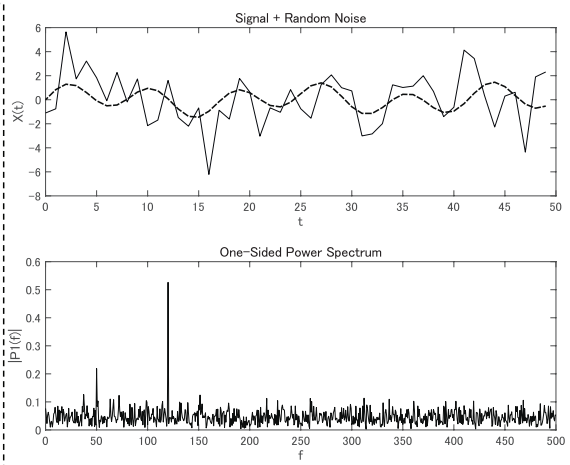


図 13.1: 信号（Signal，点線）に白色ノイズ（Noise）を重ねた実時間信号スペクトル（X，実線）（右上）とその FFT による周波数スペクトル（右下）.

- 2 つのサイン関数を重ねた信号（Signal）と白色ノイズ（Noise）を生成．この 2 つを重ねて実時間信号スペクトル（X）を作成し，これをサンプルとする（図右上）.
- Y=fft(X) により実時間信号 X から，FFT により周波数スペクトル Y を計算する（図右下）.
- 周波数スペクトル Y をプロットした右下図には，一様なノイズの中の $f = 50,\ 120$ の所に鋭いピークが見え，その強度も入力強度の相対的強さ（1：2）を反映している.

13.2　ウェーブレット変換

13.2.1　ウェーブレット変換の理論

フーリエ変換は，3 角関数を重ね合わせることにより与えられた信号を再現しようというものでした．3 角関数の性質は理解しやすく，またそれが持つ直交性，完全性も理解しやすいため，フーリエ変換は大変分かり易

いものです．しかし，それは無限に広がった周期性を持つということに起因した性質であるため，局所的に特徴を備えた信号の取り扱いは最も不得手とするところでもあります.（デルタ関数と平面波関数という両極端の波の性質は，量子力学では（実空間と運動量空間，あるいは観測時間と周波数の間の）不確定性原理として広く知られています.）

13.2.1.1 ウェーブレット変換と逆変換

このような欠点を補う目的のために，ウェーブレット（wavelet）と呼ばれる時間 t の狭い幅の中の信号波の重ね合わせで表現することが考えられています:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\Big(\frac{t-b}{a}\Big) \quad .$$

a，b を色々変えた波（波の中心をずらせたり，波の広がり方を変える）を重ね合わせることにより特徴をとらえた波を構成します.
連続なウェーブレットを用いた次の積分変換 (関数 $f(t)$ を 2 乗可積分関数として)；

$$[W_\psi f](a,b) = \int_{-\infty}^{\infty} dt \;\; \overline{\psi_{a,b}(t)}\; f(t)$$

を連続ウェーブレット変換と呼びます．$[W_\psi f](a,b)$ は a, b の有界連続関数となります.
選んだウェーブレット $\psi(t)$ がある条件（許容条件（admissibility condition）という）を満たすなら，次の逆変換が成り立ちます：

$$f(t) = C_\psi^{-1} \int_{-\infty}^{\infty} da \int_{-\infty}^{\infty} db \;\; a^{-2}\, [W_\psi f](a,b)\, \psi_{a,b}(t) \quad .$$

係数 C_ψ は $\psi(t)$ のフーリエ変換の適当な積分であり，また許容条件はこれが有限の値であることです．ウェーブレット変換は高速フーリエ変換を用いて計算できます．詳細は他の教科書（例えば「応用のためのウェーブレット」（日本応用数理学会監修，山田・萬代・芦野著，共立出版 2016 年)）を参照してください.

13.2.1.2　代表的なウェーブレット

代表的なウェーブレット（マザーウェーブレット）として，次のようなものが用いられます：

(1)　Real Shannon wavelet (sinc wavelet)

$$\psi(t) = 2\ \mathrm{sinc}\ (2t) - \mathrm{sinc}\ (t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t}$$

(2)　Molret wavelet　$\psi(t) = \pi^{-1/4} \exp(-\frac{t^2}{2}) \cos\left(\pi t \sqrt{\frac{2}{\log 2}}\right)$

(3)　Mexican hat wavelet　$\psi(t) = \frac{2}{\sqrt{3}\pi^{1/4}}(1-t^2)\ \exp\left(-\frac{t^2}{2}\right)$

13.2.1.3　直交基底による（離散）ウェーブレット展開

ウェーブレット $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ が 2 乗可積分関数系の正規直交基底となるなら，任意の関数 $f(t)$ がこれにより展開できます：

$$f(t) = \sum_j \sum_k c_{j,k}\psi_{j,k}(t)\ .$$

展開係数 $c_{j,k}$ を詳細係数と呼びます.

13.2.2　MATLAB によるウェーブレット応用：ノイズ除去とデータ圧縮

MATLAB でウェーブレットを用いるには，Wavelet Toolbox が必要です. 短時間の信号の解析のみならず，色々な局面でウェーブレット変換が使われています.

13.2.2.1　ノイズ除去

ノイズは，信号をウェーブレットに分解した後の高周波数成分に含まれ，意味のある信号情報はほとんどありません.（離散）ウェーブレット展開の展開係数（詳細係数）数 $c_{j,k}$ を，どのくらいの大きさ（閾値）以下ならどのように変更するか（ハードまたはソフト）が，一般に選択できます. 閾値は高周波数領域では高い値に、低周波数領域では低い値に設定するのが普通のようです.
図 13.2に簡単な例を示しましょう.

```
load noisdopp
denoise = wdenoise(noisdopp);
subplot(2,1,1);plot(noisdopp);
title('Original Signal')
hold on
subplot(2,1,2);plot(denoise)
title('Denoise Signal')
```

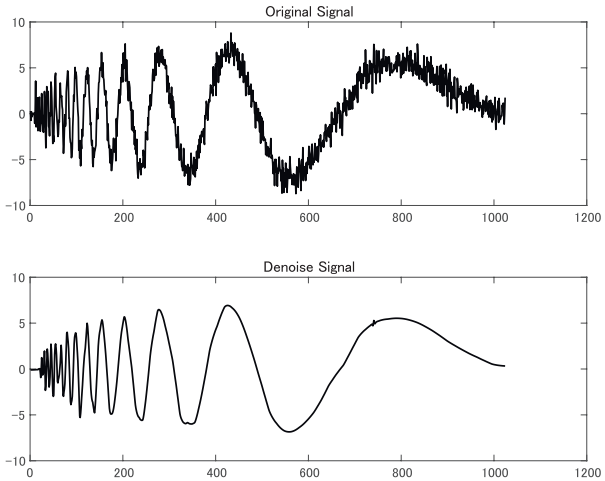


図 13.2: ウェーブレットののった信号（右上）とウェーブレット変換を用いたノイズ除去後の信号（右下）.

- Wavelet Toolbox に用意されている 1 次元信号強度の入った 1×1024 データ noisdopp (noisy Doppler signal) を Load し使用.
- ウェーブレットのノイズ除去プログラム wdenoise(x) を使用．wdenoise(x, level) とすれば閾値指定も可能．level に関してはマニュアル参照のこと.

13.2.2.2 データ圧縮

写真やイラストの画像ファイルとして用いられている GIF. JPEG, JPEG 2000 などは画像の圧縮形式を意味し，それぞれの方式で作られた画像ファイルには拡張子 .gif，.jpg あるいは .jp2 が付きます．これらのファイル形式ではデータ圧縮の際に，色数を落とす方法（GIF) あるいは色調の変化のデータを落とす方法（JPEG，JPEG2000）方式を採用しています．JPEG と JPEG 2000 では，その色を周波数成分に分解しその内の高周波成分を落とすことにより，データ圧縮を行います．JPEG および JPEG2000 では，入力画像に対する周波数変換に，それぞれ，離散フーリエ変換あるいは離散ウェーブレット変換を行います．その他にも両方式に大きな相違点もあります.

ここでは離散ウェーブレット変換を用いたデータ圧縮を，MATLAB を用いて行ってみましょう．データ圧縮は $1/2^N$ 単位で高周波成分を平滑化していきます．高周波成分を落とすため，データ圧縮を大きく進めると像の明瞭さが失われます．以下手順を追っていきます．

- **Step1** Toolboxで提供されている画像 'woman' を利用．自分の JPEG2000 ファイルを利用するなら imread('⋯.jp2') とする．

$$\begin{aligned}&\text{load woman;}\\&\text{image(X)}\\&\text{title}(\prime\text{Original Image0}\prime)\\&\text{colormap(map)}\\&\text{size(X)}\end{aligned}$$

  - load した段階で（名前は指示してないが），ファイル map（255 × 3，倍精度）と X （256 × 256，倍精度）ができている（原図は図 13.3）．
  - image(X) は、配列 X 内のデータをイメージとして表示する．
  - colormap(map) は画像ファイル X のカラーマップを， map で指定されたカラーマップに設定．
  - size(X) でデータ X の情報（256 × 256）を確認．

- **Step2**　データ分解のプロセス．

$$\begin{aligned}&\text{wv} =\prime \text{db1}\prime;\\&[\text{cfs}, \text{inds}] = \text{wavedec2}(\text{X}, 3, \text{wv});\end{aligned}$$

  wavedec2(X,3,wv) は ウェーブレット wv（上の指定では，Daubechies ファミリ の db1-ウェーブレット）を使用して行列 X の レベル 3 の 2 次元ウェーブレット分解を行う．

- **Step3**　分解データ cfs からレベル 2（64 × 64），およびレベル 3（32 × 32）のデータを抽出．レベル 2 のスクリプトのみ，図 13.4に

図 13.3: 画像データ圧縮のためのサンプル（256 × 256).

は両方の画像を示す.

```
cfs2 = appcoef2(cfs, inds, wv, 2);
figure;
imagesc(cfs2)
colormap('gray')
title('Level2ApproximationCoefficients')
size(cfs2)
```

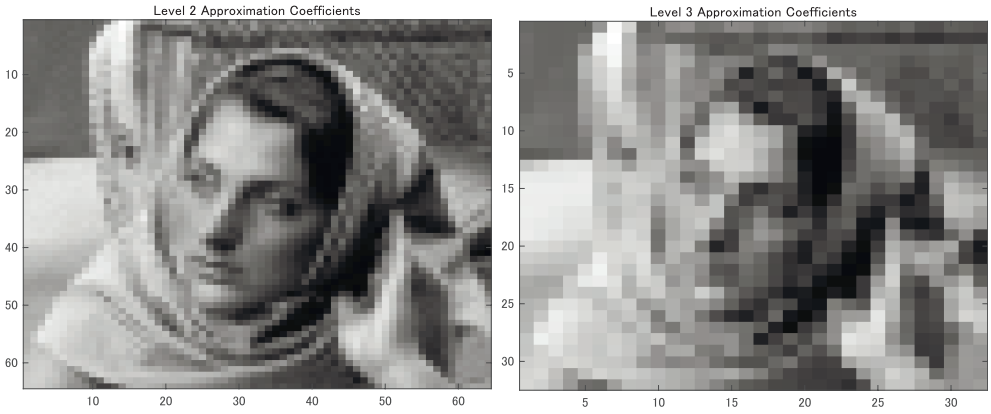


図 13.4: ウェーブレットによりデータ圧縮された画像.（左）レベル 2（64×64）と（右）レベル 3（32×32）.

第14章　行列の特異値分解を用いた低ランク近似と画像圧縮

デジタルな画像について，その処理に特異値分解を応用してファイル容量を圧縮することができます.

14.1　画像の行列表現と低ランク近似

画像データは$n \times m$ピクセルの点に3色に分けて（サイズ$n \times m \times 3$) 収められます.
簡単のために画像を白黒画像とし，そのデータをどの程度圧縮できるか考えてみましょう．画像データを2次元行列に見立て，その特徴を少ない主要な要素で再現しようというものです．そのために**行列の特異値分解を用いた低ランク近似**を採用します.

もとの画像が白黒で$n \times m$行列Iで与えられているとします．ピクセル数は変えないでおきます．データ量は

$$n \times m$$

です．実際にはこのファイルに8bitsデータを載せて，各点の明度を2進数を用い256段階（$2^8 = 256$）で表します.

この行列を特異値分解

$$I = U * S * V^T$$

するのです．特異値分解のために数値計算をします．そのためには8bitsデータを実数型に直さなくてはいけません.

必要な情報データ量は $P = \min(n, m)$ として，

$$P \times (1 + n + m)$$

です．$\min(n, m)$ は，n か m のいずれか小さい方の値を意味します．
このままでは最初のデータ量 $n \times m$ より大きくなります．しかし画像を認識するにはこのようなデータ全部が必要ではありません．P をどの程度小さくできるかがポイントです．P を n, m の 10 分の 1 にすることができれば，全体として数分の 1 までデータを圧縮しても元のデータが持つ情報量を落とさない，あるいは元の画像が持つ特徴を，特異値という形でとらえることができた，ということになります．

14.2　MATLAB を用いた画像圧縮

利用する特異値が k 個であるとして，k をどの程度まで小さくできるか，データ量

$$k \times (1 + n + m)$$

をどこまで小さくできるかがポイントとなります．実際に MATLAB を用いてやってみましょう．

Step1 指定された画像ファイル YasudaP.jpg を行列 img として読み込みます：

```
>> img = imread('YasudaP.jpg');
```

この中身がどのようなファイルか，whos コマンドを用いて確認します．

```
>>  whos  img
  Name          Size              Bytes    Class    Attributes
  img     2515 × 2525 × 3       19051125   uint8
```

画像ファイル YasudaP.jpg は　img(2515, 2525, 3) ファイルとして読み込まれました．unit8 はデータ形式が 符号なし整数 (unsigned integer) 8bits ファイルであることを示しています．

Step2-1 カラー画像は RGB 画像といい，3 枚の二次元画像から成ります．R(red)，G(Green)，B(blue) が，行列の 3 番目の引数の 1 ～ 3 に対応します．

$$>> \quad \text{redImg} = \text{img}(:,:,1);$$
$$>> \quad \text{greenImg} = \text{img}(:,:,2);$$
$$>> \quad \text{blueImg} = \text{img}(:,:,3);$$

とすれば，RGB の 3 要素を分離して其々が白黒画像に変換されます．

これらの画像ファイルを開くには

$$>> \text{imtool(redImg)}$$

とするか，

$$\text{mwrite(redImg,}' \text{redImg.jpg}');$$

として画像ファイルに出力するかの 2 つの方法があります．

Step2-2 カラー画像を一度に白黒画像に変換するためには rgb2gray という関数が用意されています．

$$>> \quad \text{grayImg} = \text{rgb2gray(img)};$$

この関数 rgb2gray では重み付加算

$$\text{gray} = 0.2989\text{R} + 0.5870\text{G} + 0.1140\text{B}$$

を用いています．

- このファイル grayImg は少し大きすぎますので，ファイルサイズを小さくしておきましょう．そのために 画像ファイルを小さくするための関数 imresize を使います．

$$>> \text{grayImg0} = \text{imresize(grayImg, 0.2)};$$

これでファイルのピクセル数が 0.2 倍になります：

```
>>  whos grayImg0
  Name        Size          Bytes    Class    Attributes
  grayImg0    503 × 505     254015   uint8
```

$254015 = 503 \times 505$. 数値は Bytes 単位です（2^8 : 8bits=1Bytes)

Step3 これから行列の特異値分解を行いますが，整数クラス uint8 のままでは数値演算に不向きです．そのため行列のデータを関数 double を用いて 64bits 倍精度浮動小数点数に変換します．

```
>> grayImggray0 = double(grayImg0);
>> whos  grayImggray0
```

Name	Size	Bytes	Class	Attributes
grayImggray0	503×505	2032120	double	

$2032120 = 503 \times 505 \times 8$：64bits=8Bytes

- これで特異値分解の準備が整いました．

```
>> [U, S, V] = svds(grayImggray0, 50);
```

ここでのパラメータ 50 は，特異値の大きな方から 50 個で分解するということを指示します．

こうして 50 個の特異値で行列を再成します．

```
>> gray50 = U * S * V';
```

U, S, V の大きさを whos コマンドを用いて確認してみます：

Name	Size	Bytes	Class	Attributes
gray50	503×505	2032120	double	
U	503×50	201200	double	
V	505×50	202000	double	
S	50×50	20000	double	

S は対角行列ですから 50×50 は不要で 50 だけで十分です．こうしてデータ量が大きく圧縮されました．

$$503 \times 505 \to 50 \times (1 + 503 + 505)$$

$k = 50$ を色々変えて，画像の再現性を確認してみてください．実際には k の値をもっと小さくしても何が映っているか充分識別できるでしょう．

Step4 gray50 は double のデータ型行列です．画像ファイルとするために uint8 型に変換します．

```
>> igray50 = uint8(gray50);
```

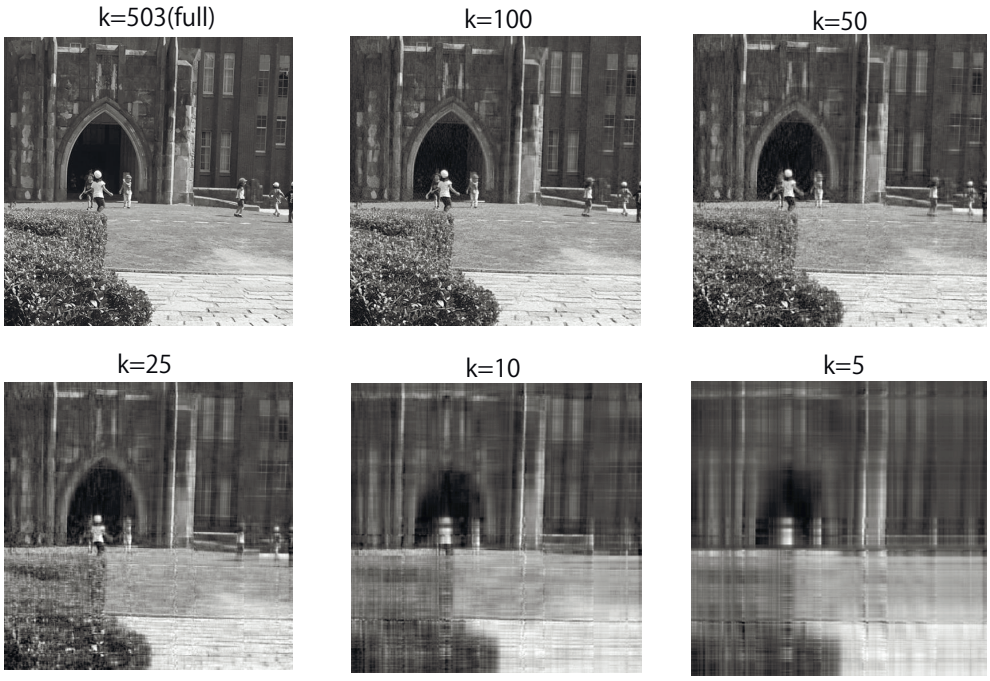


図 14.1: 特異値を用いた画像の低ランク近似（安田講堂前で遊ぶ幼児たち）．大きい方から k 個の特異値（固有値）を用いた．

Step5 最後は結果を画像ファイルとして保存します．

$$\text{imwrite}(\text{igray50},'\,\text{gray50.jpg}');$$
$$\text{imwrite}(\text{igray50},'\,\text{gray50.png}');$$

これらの結果を図 14.1に示します．特異値の 10 分の 1 程度（$k \simeq 50$）で充分に元の画像が再現されています．

第15章　シミュレーション

15.1　シミュレーションとは

実際のシステムを動かしてその動きを確かめることが難しい，危険だ，手間がかかり過ぎるため条件を種々変えることができないなどの場合，あるいは，実際の現象がゆっくりしているため限られた時間内に知見を得ることができない，逆に非常に高速な現象であったりシステムの内部の現象であるため観測不能である，という場合など，興味があり重要であるにもかかわらず観測できない現象が数多くあります．また現象をコントロールするパラメータを人為的に変えることができず，現象の知見を充分に得ることが難しいこともしばしばです．このような場合，シミュレーション（simulation）という手法によってシステムの特性や実際の現象の実際が調べられます．
実物大の模型あるいは実物を縮小した模型を作って風洞や水槽の中で試験するのもシミュレーションの一種です．ここでとり上げるのは，実際の現象を表す**数理モデル**で置き換え，コンピューターを用いた**数値実験**をおこなう方法です．具体的には，電気系，機械系のシステム，原子スケールでの物理・化学現象，地球スケールや宇宙スケールでの自然現象，社会現象に関して広く用いられています．

15.2　Simulink

MATLABでは，シミュレーションツールとして，Simulinkが用意されています．Simulinkは，様々な系の数学モデル（微分方程式等）を配置し観測するための，MATLABの下で動く**GUI（Graphical User Interface）**です．したがって容易にモデルの系を設定することができます．
Simulinkでは，**ブロック図ツール**とそれらのブロック図を種類ごとにまとめた**ブロックライブラリ**が入出力および機能要素として提供されています．

15.2.1 Simulink の使い方 1.

ここでは Simulink の立ち上げと簡単な使い方を紹介しましょう.

15.2.1.1 Simulink の立ち上げ

Simulink を立ち上げるには 2 つの方法があります.

方法 1 コマンドウィンドウに 「simulink」と入力.

方法 2 ホームタブから

1 ホームタブの「Simulink」アイコンをクリック

2 （あるいは）ホームタブの「シンク作成」アイコンをクリックして現れるスクロールバーから「Simulink モデル」を選択.

方法 1，方法 2 のいずれでも「Simulink スタートページ」が起動します.

15.2.1.2 Simulink の「モデルブラウザー」の起動

次に Simulink の操作を行う「モデルブラウザー」を起動します.

方法 「Simulink スタートページ」の「Simulink」から「空のモデル」を選択.

これで「モデルブラウザー」が新たに起動します.

15.2.1.3 Simulink の「ライブラリ ブラウザー」の起動と「ブロック」の選択

方法 「モデルブラウザー」の「シミュレーション タブ」にある「ライブラリブラウザー」アイコンをクリック.

以上で,「Simulink ライブラリ ブラウザー」が開きます.
この「ライブラリ ブラウザー」の中から，必要な部品を探します．ライブラリーは，機能ごとにグループ化され，ブロック毎に置かれています.「ライブラリ ブラウザー」の検索の窓を使うと効率の良い選択が可能ですが，名前で検索しますので慣れるまで少し時間がかかるかも知れません.

15.2.2　Simulink の使い方　2．

以上で準備が整いましたので，実際の Simulink 使用の手順に移りましょう．ここでは基本的な利用と RL 回路（または簡単な力学系）を例にとってみましょう．

15.2.2.1　波形のモニター

波形の発生とそのモニターの 2 つから成り立ちます．

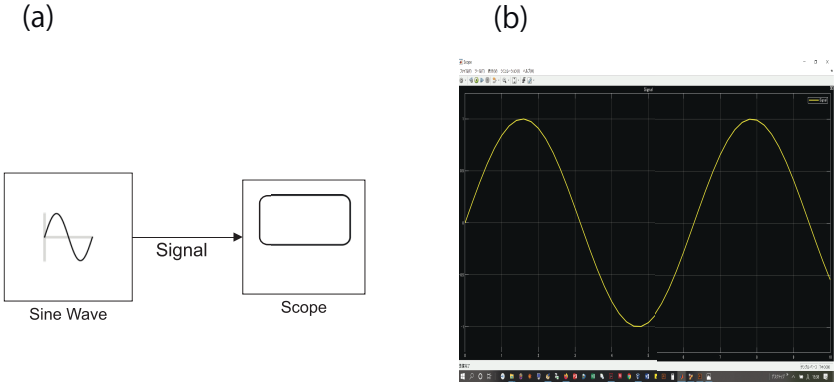


図 15.1: (a) モデルブラウザー上の，ブロックから構成されたモデル．(b) モニターに現れる信号．

Step1 Sources Blocks から「Sine Wave」を選び,「モデル ブラウザー」上にドロップする（波形の発生）．

Step2 Commonly Used Blocks から「Scope」を選び,「モデル ブラウザー」上にドロップする（信号のモニター）．

Step3 「Sine Wave」と「Scope」を**結線**する．A から B へ結線したい場合は , [Ctrl] キーを押しながら, マウスで A, B ブロックの順に左クリックする．結線上に名前を付けたいときには，描きたいところでダブルクリックして記入する．
Simulink のためのファイルを…という名前で保存すると，slx という拡張子が付き，….slx という名前が付く．

Step4 モニター信号を見るために「Scope」ブロックをダブルクリックしてグラフをオープンする．その上で，シミュレーションタブ上の実行ボタンをクリックする．

Step5 「Sine Wave」ブロックをダブルクリックすると,「ブロックパラメーター」ウィンドウが開く．ここで入力信号のパラメータを選択できる．

図 15.1にこれらを示します.

15.2.2.2 2 階常微分方程式系

2 階常微分方程式で表されるシステムは沢山あります．ここでは 1 質点振動系を考えます．問題にする微分方程式は

$$m\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + kx = f(t)$$

です．x は質点の時間 t に依存する変位，m は質点の質量，γ は (減衰) 抵抗定数, k はばね定数, $f(t)$ は時間に依存する外力です.
上の式を

$$\frac{d^2x}{dt^2} = \frac{1}{m}\left[f(t) - \gamma\frac{dx}{dt} - kx\right]$$

と書き直しましょう．このようにすると,「外力，粘性抵抗，ばねによる力を足し合わさせて $1/m$ 倍すれば，x の 2 階微分（加速度）となる」と読めます．そしてそのようなブロック線図を書けばよいということになります.

Step1 「Sum」ブロックを探しモデルブラウザーにドロップ.

Step2 「Sum」ブロックをダブルクリックし，ブロックパラメータ ウィンドウを開き，符号リストに「＋－－」と入力する．これにより Sum block の入力の数と符号を指定．＋を $f(x)$ に，残りの 2 つを $-\gamma\frac{dx}{dt}$ と $-kx$ に割り当てることができる.

Step3 出力全体に $1/m$ を掛けるために,「Gain」ブロックを探してモデルブラウザーにドロップ．Sum ブロックから Gain ブロックに結線.

Step4 （Gain ブロックをダブルクリックし，パラメーターウィンドウを開いて,）Gain ブロックの重みを $1/m$ に書き換える．m の大きさは後で指定．Gain ブロックの出力は $\frac{d^2x}{dt^2}$ であるので，そう書いておこう.

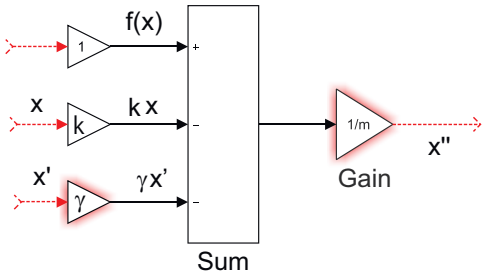


図 15.2: 振動子系モデル 1 .

その様な様子は図 15.2に示されています．次に，この力学系ダイアグラムを完成し、必要な量に対するモニターを置きます．

まず x' および x が必要です．そのために x'' を 1 回，さらにもう 1 回積分します．

Step5 積分のための「Integrator」ブロックを探しモデルブラウザーにドロップ．

Step6 必要なところを結線して全体の系を完成．

Step7 外力として適当なものを選択．ここでは，ある時刻（0）にゼロから有限の値に立ち上がり，その後一定値を保つ「Step」を外力とする．観測の必要なところに「Scope」をつなぐ（図 15.3）．

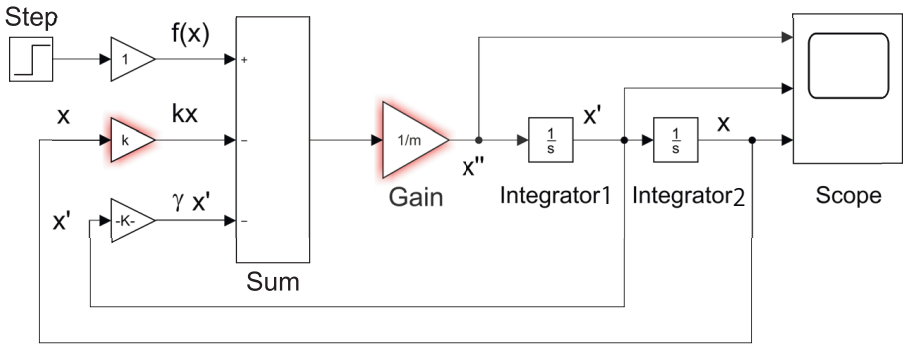


図 15.3: 振動子系モデル 2.

これで準備が完全に終わったのではありません．振動子系の定数 m, k, γ の値を決めなくてはなりません．

Step8 「Scope」をダブルクリック．

Step9 パラメータの値を設定するには 2 つの方法がある．
1. コマンドラインからの直接入力．
2. ….m ファイルで，パラメータの値を設定．そのファイルを通常の方法で走らせる．

Step10 Simulink のファイル….slx を実行する．結果は Scope に出る（図 15.4）．

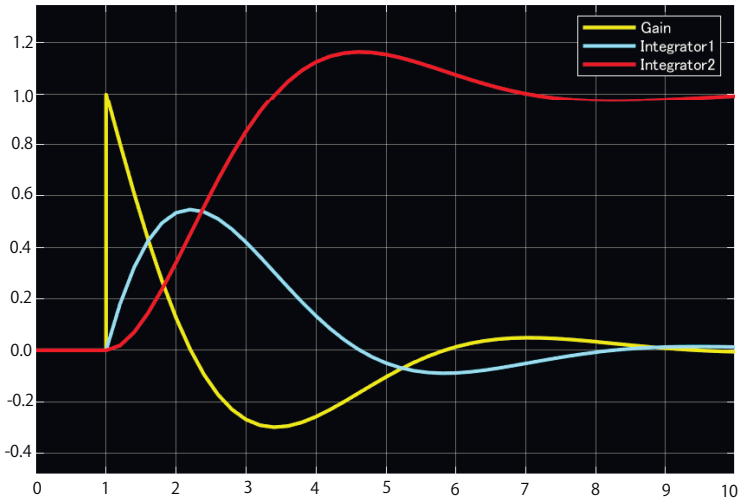


図 15.4: 振動子系モデル 3.

図 15.4で，ステップ的な外力が与えられて加速度が 0 から有限の値に立ち上がり，それに伴って測度および変位が 0 から連続的に変化すること，また粘性抵抗に従って振動が減衰する様子も観測されます．Scope において，3 つの信号が 1 つの図にプロットされていますが，それぞれの単位は違います．Scope ブロックのパラメータを変えれば 3 つのそれぞれを，異なる図に描くこともできます．

第16章　深層学習，機械学習など

16.1　人工知能，機械学習，深層学習とは

人工知能（AI=Artificial Interiggence），機械学習（Machine Learning），深層学習（Deep Learning）などの言葉が，巷では多く聞かれる時代になりました．具体的に何がどのように違うのか判然としないまま，呪文のように口にしているように思われる場面も多く見かけるようになりました．はじめに人工知能，機械学習，深層学習というキーワードをきちんと区別しておきましょう．全体の概念構成は

$$\text{人工知能} \supset \text{機械学習} \supset \text{深層学習}$$

となります．

- 人工知能（AI) という概念は 1950 年代に形成されたもので,「人間にしかできなかったような高度に知的な作業や判断をコンピュータを中心とする人工的なシステムにより行えるようにしたもの」を意味する（IT 用語辞典）．具体的応用分野として，画像認識システム，音声認識システム，自然言語処理システム，自動運転システムなどが実現されている．

- 機械学習とは、人工知能を実現するための手法の 1 つの方法で，人間の経験による学習と同様な学習機能をコンピューターに教えるアルゴリズムをいう．機械学習では，機械的（コンピューター的）方法により，データから直接情報を ”学習” する．学習するデータが増えるにしたがい，一般に，獲得する性能が向上する．

  機械学習では分析のためのロジックを全てプログラムするのではなく，コンピュータによる学習により，大量のデータから，その特徴をグループ分けする。その獲得したロジックによって，さらに与えられたデータに対して自律的に高度の予測を行う手法が構成される．

学習の過程は「入力（データの前処理，適切な表現形式の選択）」「処理」「出力（回帰などのデータ予測）」から構成される．処理の過程には**ニューラルネットワーク**（neural network）という人間の脳を構成するニューロン（neuron）のネットワーク形成を模したアルゴリズムを用いる．

- 深層学習とは，機械学習を有効に行うための方法．最初の処理の結果をさらに入力として処理するというように，処理の過程（ニューラルネットワークの構成）を多層的・多段的に行う．これにより有効な機械学習が実現された．

以下で機械学習，ニューラルネットワークに関していくつか紹介したいと思います．多分，これらに付いて解説書を読むより，実際に即して計算を経験するのが分かり易いように思います．

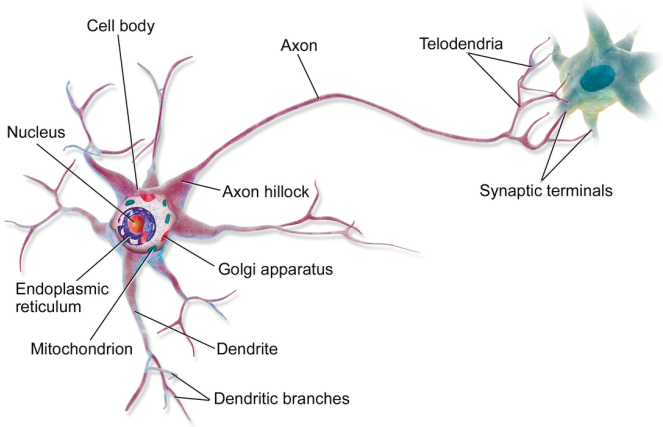


図 16.1: ニューロン．Wikipedia より（C.C. BY 3.0).

16.2 深層学習

機械学習といっても，このソフトウェアにデータを投げ込むと，知らないことも解決してくれる，というようなものはありません（多分!?）．下記項目はお互いにそれぞれがかかわりを持って機械学習を構成します．

16.2.1　機械学習

MATLAB では Statistics and Machine Learning Toolbox が用意されています．そこでは，統計および機械学習による解析とモデル化が基本となります．

16.2.1.1　機械学習のためのステップ

機械学習は以下のようなステップ（の総て，または一部）から構成され，そこでは様々な基本的数理手法が反復して使われます．
https://jp.mathworks.com/solutions/machine-learning/resources.html?s_tid=conf_addres_DA_eb の MATLAB 「機械学習入門」に多くの解説・実例があります．

- 入力データ
  - データの表現：画像データ，信号データ，文章（自然言語）
  - モデル：[観測データ] $\leftrightarrow$ [ラベル]
- 情報の抽出（データマイニング）
  - 最適化
  - 回帰分析
  - EM アルゴリズム
- 回帰（regression）と分類（classification），クラスタリング（clustering）
  - 統計手法（回帰分析）
  - 「教師あり学習」と「教師なし学習」
- ニューラルネットワーク
  - 最適化問題
  - 「教師あり学習」と「教師なし学習」
- 評価，検定
  - 訓練データと評価データ

16.2.1.2 入力データ

入力データは，音声データ，画像データ，数値データ，文章など様々な形が考えられます．例えば画像データの場合，画素単位の RGB 値や明るさあるいはその広がりなどが入力データとなります．文章の場合には，それらを単語に切り分けたものとそのつながりが入力データとなります．

16.2.1.3 「分類問題」における「教師あり学習」,「教師なし学習」

各データを表すサンプル点が特徴空間（画像なら RGB 値や明るさをそれぞれ要素とする多次元空間）に分布しているとき，このサンプル点を分類する方法は，大きく分けて二つあります．

- 「教師あり学習」の分類問題である **k 近傍法**（k-nearest neighbor algorithm, k-NN）．分類したいクラスタの特徴点からの空間内の距離によりサンプル点を分類していく．
- 「教師なし学習」の分類問題である **k 平均法**（k-means clustering）．クラスタの数を決め，最初にサンプル点に乱数によってクラスタを割り当てる．各クラスタに属するサンプル点の重心を求めクラスタの中心とする．サンプル点を近いクラスタ中心に割り当て直す．これらの作業を何回か繰り返す．

16.2.1.4 最適化

既に説明した方法により最適解を求める過程．統計処理における「推定」と「検定」，あるいは「線形計画法」と「非線形計画法」など．

16.2.1.5 潜在変数の抽出 ー EM アルゴリズム

EM アルゴリズム=expectation-maximization algorithm
サンプル点の分布 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_N\}$ に隠れた変数（潜在変数）z がある場合に，その潜在変数を最尤推定する手法の一つ．手順を（不十分ではありますが）少し具体的に示すために，いくつかの定義式および関係式を

提示します．より詳細には，例えば「機械学習」（中川祐志著，東京大学工学教程 （丸善出版））を参照してください．

$$\begin{aligned}
\text{データの集合} &: \mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_N\} \\
\text{サンプル点 } \boldsymbol{x}_n \text{に対する潜在変数の値} &: z_n \\
\text{確率分布を決めるパラメタの集合} &: \boldsymbol{\theta} \\
\text{条件付確率} &: p(\boldsymbol{x}, z|\boldsymbol{\theta}) = p(z|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{\theta}) \\
\text{潜在変数 } z \text{ 込みの対数尤度} &: \log L(\boldsymbol{x}, z|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n, z_n|\boldsymbol{\theta})
\end{aligned}$$

$\boldsymbol{\theta}$ を分布 $\mathcal{D}$ から推定しよう，その場合に対数尤度を最大にするように決めよう，というのがここでの問題です．
観測変数 $\boldsymbol{x}$ と潜在変数 z の同時分布 $p(\boldsymbol{x}, z|\boldsymbol{\theta})$ が与えられています．このときの EM アルゴリズム（対数尤度関数 $\ln p(\boldsymbol{x}|\boldsymbol{\theta})$ の $\boldsymbol{\theta}$ についての最大化）は以下のように行われます．

Step1 ．　初期化．初期値 $\boldsymbol{\theta}^{(n=1)}$ を適当に決める．

Step2 ．　(E ステップ)　z の事後分布 $p(z|\boldsymbol{x}, \boldsymbol{\theta})$ を計算する．

Step3 ．　(M ステップ)　事後分布 $p(z|\boldsymbol{x}, \boldsymbol{\theta})$ による対数尤度の期待値

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) = \sum_{z} q(z) \ln p(\boldsymbol{x}, z|\boldsymbol{\theta})$$

を計算する．
ただし $q(z) = p(z|\boldsymbol{x}, \boldsymbol{\theta}^{(n)})$ と選ぶ．ここは既知の情報から決まる z の関数なら何でもよいのだが，一般にこのように選ばれる．
$\boldsymbol{\theta}$ は $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)})$ を最大化するように決める：

$$\boldsymbol{\theta}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) \ .$$

Step4 ．　$\boldsymbol{\theta}^{(n+1)}$ に対応する対数尤度関数 $\ln p(\boldsymbol{x}, z|\boldsymbol{\theta}^{(n+1)})$ を計算し，これが収束条件を満たしていない場合には Step2 に戻る．

一般に EM アルゴリズムがどういったこと をしているのかについての直観的理解の助けとなると考えられる．

16.2.1.6 学習による結合の強化と Overfitting,「過学習」,「過適合」

学習により上式の重み w_i の重み付けを変化させます．一般に与えられたデータにモデルを適合させることは，可能です．一方で与えられたデータ（訓練データ）に適合させすぎると，モデルの予測能力が落ちることはしばしば経験することです．このような状況を overfitting，日本語で「過学習」あるいは「過適合」といいます．これを如何に避けるかは機械学習，深層学習では重要な課題です．1 つの対応策は，モデルにある程度の柔軟性を持たせることです．その様な方法を「正則化」といいます．

16.2.1.7 クラス分類，多クラス分類（ロジスティック回帰）

パーセプトロンで 2 クラスに分類することは上の式の通り．多クラス分類は一般化（多値）．

16.2.2 ニューラルネットワーク

- ニューロン：動物の神経細胞は，細胞核のある細胞体，他の細胞からの入力を受ける樹状突起，他の細胞に出力する軸索から成り立ち，これら全体をニューロン（neuron）という．軸索の先端は他の神経細胞に接続してその間にシナプス（synapse）を形成する．ニューロンは他のニューロンからの刺激（信号）を受け取り，シナプスでそれを増減させ他のニューロンに伝達する．

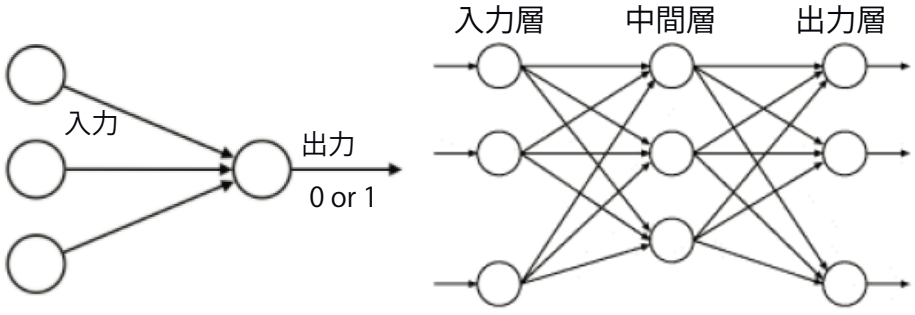


図 16.2: パーセプトロン（左）と階層型ニューラルネットワーク（右）.

- パーセプトロン ：人工ニューロン．複数の入力に対して 0 か 1 の 1 つの出力を与える回路あるいは関数（下式（単純パーセプトロン）および図 16.2左）．各入力を x_i，出力を y と書くと

$$\begin{aligned} x &= I_0 + w_1 x_1 + \cdots + w_n x_n \\ y &= h(x) \\ h(x) &= \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases} \end{aligned}$$

  また I_0 をバイアスと，$h(x)$ を一般に活性化関数という．活性化関数には，ここで示した階段関数のほかに，0 と 1 の間を滑らかにつなぐ種々の非線形関数が考えられている．

- ニューラルネットワーク：人工ニューロンから構成したネットワークが，学習によってシナプスの結合強度を変化させ，問題解決能力を持つようにしたモデル（図 16.2右）．

- 深層学習：多層のニューラルネットワークによる機械学習．

16.3　MATLAB に用意されている深層学習

かなりのことは Deep Learning Toolbox を用いて実行できます．

1. MATLAB は，種々の入力に対する機械学習，深層学習の多くのプログラムを提供しています．それらを動かし，また入力データを自前のものに取り換え，あるいは書き換えるなどしながら具体的に学ぶことが可能です．

2. Deep Learning Toolbox では，Simulink を用いてニューラルネットワークを作ることができるようですので，便利だと思います．

MATLAB が提供する具体的な例を見ることにしましょう．既に用意され調整されているパッケージや関数を用いますが，それらに慣れた上で自分で少しずつ変更を加えたり，新たに書き加えするのが良いでしょう．その場合に，関数のコードがどのように書かれているかを見るのには，type コマンドを使うのが便利です．

```
type file_name
```

16.3.1 例：機械学習による 5 種類の「木の実」の識別・分類

MathWorks 社提供による NutMeasurement.mlx の解説を与えることにしましょう．画像に写っているものを識別・分類する機械学習のプロセスとして，大変具体的かつ詳細に実行していますので，良い教材といえるでしょう．
以下の例は MathWorks 社に提供して頂いた機械学習を用いて画像の中の木の実を分類するものです．入力データは画像として与えられています．総てのコマンドは MATLAB で与えられていますので，マニュアルで確認してください．またプログラム中の図の出力は，一部のみここに示してあります．実際に自分で動かしてみてどのような出力が得られるのか確かめてください．

- 画像データを読込み

```
I = imread('DSC_1895cr.png');
figure; imshow(I);
```

図 16.3: 入力画像 （DSC_1895cr.png）.

- データの準備（情報の抽出=ナッツ領域の切り出し）
  - 輝度変換 （RGB 値を人の色空間を近似する CIE1976 L*a*b* 色空間 (CIELAB) に変換.

```
Ilab = rgb2lab(I);
[L, a, b] = imsplit(Ilab);
```

- エッジ検出 （強度の微分強度が大きなところを検出）.

```
[Iedge, th] = edge(L, 'canny', 0.06);
imshowpair(L,Iedge, 'montage')：二つの像を並べて比較.
```

- Morphological Closing という手法を用いて作った閉じた図形（穴埋めされた図形）の像を返す.

```
bw = imclose(Iedge, strel('disk', 3));
bwNuts = imfill(bw, 'holes');
imshow(bwNuts)
```

以上で ナッツ領域の切り出しが完了（それを確認のため imshow）.

```
figure, imshow(I .* repmat(uint8(bwNuts), [1,1,3]));
```

- 分類

- 形状で分類: Area（面積） と Eccentricity（楕円度） で「ジャイアントコーン」と小さいグループ大きいグループに分類する．次のコマンドでイメージの領域解析 アプリを開く.

```
imageRegionAnalyzer(bwNuts);
```

「stats = regionprops(BW,properties)」 はバイナリイメージファイル BW 内の各 8 連結要素 (オブジェクト) について properties で指定されるプロパティの測定値を返す．properties は多種類が用意されているので，マニュアルを参照.
Area と Eccentricity でジャイアントコーンと小さいグループ大きいグループに分類可能.

```
stats = regionprops(bwNuts, 'Area', 'Eccentricity', ···
  'Circularity', 'MaxFeretProperties', 'BoundingBox');
  areas = ［stats.Area］;
  eccentricity = ［stats.Eccentricity］;
```

ヒストグラムで確認

```
figure, h=histogram(areas, 10); title(' 面積');
```

```
figure, h=histogram(eccentricity, 10); title(' 楕円度');
```

楕円度で「ジャイアントコーン」を分類，インデックスを付与.

```
idx_GiantCorn = [stats.Eccentricity] < 0.5;
Idetect = insertObjectAnnotation(I, 'rectangle',···
  [cat(1,stats(idx_GiantCorn).BoundingBox)], 'CORN',···
  'Color', 'blue', 'TextBoxOpacity', 1, 'TextColor', 'white');
figure, imshow(Idetect);
```

面積で 2 グループに分け（~ は論理否定の記号），インデックスを付与.

```
idx_Large = ［stats.Area］ > 1500;
idx_Small = ~ idx_Large;
```

– 色による分類小さな面積のグループに対して、色による分類.
各領域の色 (a の値) の平均値を計算，ヒストグラムで確認.

```
stats_a = regionprops(bwNuts, a, 'MeanIntensity');
h = histogram([stats_a(idx_Small).MeanIntensity], 10); ···
  title('a の値')
```

赤と緑で分類し，インデックスを付与.

```
idx_Red= ［stats_a.MeanIntensity］ > 15 & idx_Small;
idx_Green= ［stats_a.MeanIntensity］ < 15 & idx_Small;
Idetect = insertObjectAnnotation(Idetect, 'rectangle',···
 ［cat(1,stats(idx_Red).BoundingBox)］, 'RED', ···
  'Color', 'red', 'TextBoxOpacity', 1);
Idetect = insertObjectAnnotation(Idetect, ···
  'rectangle', ［cat(1,stats(idx_Green).BoundingBox)］,···
  'GREEN', 'Color', 'green', 'TextBoxOpacity', 1);
figure, imshow(Idetect)；確認のため描図.
```

ここまでで 3/5 種類の分類ができた.

– 円形度（Circularity）による分類
大きなグループに対して、円形度での分類を行う.

```
h = histogram([stats(idx_Large & ···
  ~ idx_GiantCorn).Circularity] , 9); title(' 円形度');
```

きれいに分かれていることが確認できるので円形度で分類，インデックスを付与.

```
idx_Almond = [stats.Circularity] > 0.8 & ...
  ~ idx_GiantCorn & idx_Large;
idx_Cashew = [stats.Circularity] < 0.8 & ...
  ~ idx_GiantCorn & idx_Large;
Idetect = insertObjectAnnotation(Idetect,...
  'rectangle',[cat(1,stats(idx_Almond).BoundingBox)],...
  'ALMOND', 'Color', 'magenta', 'TextBoxOpacity', 1);
Idetect = insertObjectAnnotation(Idetect, ...
  'rectangle',[cat(1,stats(idx_Cashew).BoundingBox)], ...
  'CASHEW', 'Color', 'yellow', 'TextBoxOpacity',1);
figure, imshow(Idetect)
```

以上で，形，大きさ，色，円形度，などの 5 種類を分類した.

- 解析：抽出した情報をグループごとに可視化．ラベル行列の作成.
  %以降は書式指定.

```
labels = cell(length(stats), 1);
labels(idx_GiantCorn,1) = {sprintf('%-6s','Corn')};
labels(idx_Red,1) = {sprintf('%-6s','Red')};
labels(idx_Green,1) = {sprintf('%-6s','Green')};
labels(idx_Almond,1) = {sprintf('%-6s','Almond')};
labels(idx_Cashew,1) = {sprintf('%-6s','Cashew')};
labels = cell2mat(labels);
```

データをテーブル型に

```
data_table = struct2table(stats);
label_table = table(labels);
table_all =  [data_table, label_table] ;
```

平均値を表で

```
statarray = grpstats(table_all(:, [1 3:6 end] ), 'labels')
```

表 16.1: データ平均値.

	labels	Group_ Count	mean_ Area	mean_ Eccentricity	mean_ Circularity	mean_Max FereDiam	mean_Min FereDiam
1 Red	Red	15	656.4000	0.8825	0.8242	44.5620	-124.3938
2 Green	Green	8	693.8750	0.8072	0.8393	42.3641	11.5974
3 Almond	Almond	4	2.3623e+03	0.8397	0.8904	75.3938	-67.5016
4 Cashew	Cashew	6	2.4475e+03	0.8093	0.7349	74.0489	-56.9131
5 Corn	Corn	4	3.0768e+03	0.4111	0.9981	67.9186	-92.5418

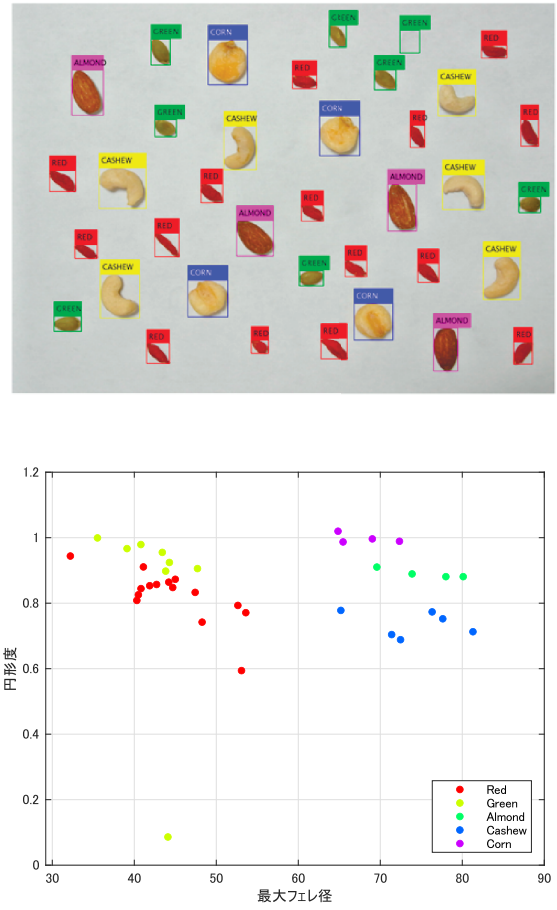


図 16.4: 出力：分類結果．上の図中のラベルの色と下の図のデータ点の色は対応していないことに注意．

グループごとの散布図（図 16.4(下)）

```
figure, gscatter([stats.MaxFeretDiameter], ···
    [stats.Circularity], labels);
xlabel('最大フェレ径'); ylabel('円形度')；
grid on;
```

パラメータごとの相関

```
plotmatrix(table2array(table_all(:, [1 3:6] )))
```

16.3.2　例：ニューラルネットワークによる分類の深層学習（TrainNetwork）

先の例では，画像中のナッツを機械学習で分類しましたが，分類の過程はユーザーがステップごとに指示をしました．次は，手書き数字の認識・分類を（「畳み込みニューラルネットワーク」を用いて）深層学習で行うための，MATLAB が提供するプログラムです．これを用いて説明します．

なお，深層学習についてとりあえず概要を学びたいという人には，
「**ゼロから作る Deep Learning**」斎藤康毅著 オライリー・ジャパン
がお勧めです．これは Python を前提に書かれていますが，Python を知らなくても，全く障害なしに理解することができます．書かれている内容も，特に難しいということはありません．

ここでの深層学習のプロセスは次のような順序で行われます．
1. 入力用画像データの読み込みと確認．
2. ネットワーク アーキテクチャの定義．
3. 学習オプションの指定．
4. ネットワークの学習．
5. 新しいデータのラベルの予測と分類精度の計算．

- **画像データの読み込みと確認**
  `imageDatastore` (イメージ データストア) は，数字の標本データをフォルダー名に基づいて，自動的にラベルを付け，`ImageDatastore` オブジェクトとして格納する．こうすることでデータをバッチ（束）単位で処理できる．

```
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet',...
  'nndemos', 'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
  'IncludeSubfolders',true,'LabelSource','foldernames');
```

  データストアにある 0〜9 の数字 10000 個の内 20 個のイメージ imds を（4 × 5 の形で）表示．

```
figure;
perm = randperm(10000,20);
for i = 1:20
  subplot(4,5,i);
  imshow(imds.Files{perm(i)});
end
```

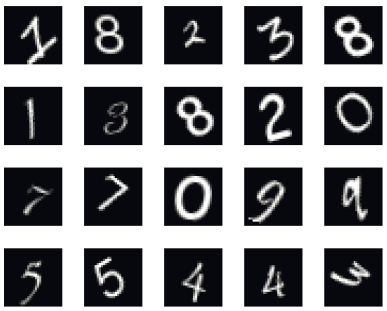


図 16.5: 手書き数字の入力データの例

`labelCount` は，ラベル，およびそれぞれのラベルが付いているイメージの数を格納する table．データストアには 0 〜 9 の数字それぞれについて 1000 個のイメージ，合計で 10000 個のイメージが含まれている．

```
labelCount = countEachLabel(imds)
```

	Label	Count
1	0	1000
2	1	1000
3	2	1000
4	3	1000
5	4	1000
6	5	1000
7	6	1000
8	7	1000
9	8	1000
10	9	1000

ネットワークの入力層にイメージのサイズを指定する必要がある．`digitData` の最初のイメージのサイズを確認（イメージはグレースケールなので各 $28 \times 28 \times 1$ ピクセル）．

```
img = readimage(imds,1);
size(img)
```

ans = 1 × 2
　　28　28

imds を学習セットと検証セットに分割する指定
`splitEachLabel` は，データストアにある `imds` を，750 の学習データセット `imdsTrain` と残り 250 個の検証データセット `imdsValidation` に分割する．

```
numTrainFiles = 750;
[imdsTrain, imdsValidation]=splitEachLabel(imds, ···
  numTrainFiles, 'randomize');
```

- **ネットワーク アーキテクチャの定義**
畳み込みニューラル ネットワークの層の指定をする核心部．

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer ];
```

Input Layer (**イメージ入力層**)：イメージ サイズ（$28 \times 28 \times 1$）を指定．1 はチャネル サイズ (グレースケールのため 1).

Convolution Layer （**畳み込み層**）： 深層学習の多くの手法が畳み込みニューラルネットワーク (CNN) を基礎に置いている．ここで「**畳み込み演算**」(「**積和演算**」．積和演算は，画像の平滑化などにも用いられている.）を行う．これは元の関数（行列）と畳み込み関数（行列）（フィルター）との積和をとるもので，最初の引数は `filterSize` といい，その値 3 は畳み込み行列のサイズが 3×3 であることを示す．フィルターを縦横にずらしながら走査するステップサイズ（Stride）のデフォルト値は［1　1］となる．'Padding' とすると行列の端に 0 要素の行および列を加えて畳み込み演算が行われる．
1 ステップ毎のスライドで以下の畳み込み演算を行うと

0	**1**	2	3
3	**0**	1	2
2	3	0	1
1	2	3	0

$*$

0	**1**
2	**0**

$\Rightarrow$

7	2	5
4	7	2
5	4	7

左辺の太字部分の 2×2 行列*2×2 行列の積和計算が矢印右辺の 7 を与える．2 つ目の引数 `numFilters` はフィルターの数（入力の同じ領域に結合するニューロンの数）を示す.

batchNormalizationLayer (**バッチ正規化層**)： 大量のデータ（具体的には配列データ）の**束**（**バッチ batch**）をまとめて処理をする．「**正規化（normalize）**」とは変数を標準化すること，例えば平均値 0，偏差 1 にそろえることをいう．ニューラルネットワークを通じて伝播される活性化と勾配を正規化することにより，ネットワークの学習は簡単な最適化問題になり，学習速度を上げる等の効果がある.

reluLayer (**ReLU 層**) ：ReLU(Rectified Linear Unit) 関数は現在広く用いられている非線形活性化関数.

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases}$$

maxPooling2dLayer (**最大プーリング層**)：畳み込み層 (と活性化関数) の後で, データの縮小処理を行うことがある（Pooling とは,

プログラムのパフォーマンスを向上させるために，オブジェクトを pool に蓄積しておき必要なときに取り出して利用するという技術）．これにより特徴マップの空間サイズが縮小され、冗長な情報が削除される．データ縮小法の 1 つの方法が最大プーリング．最大プーリング層では，最初の引数 `poolSize` によって指定された矩形領域サイズの最大値を返す．例えばこれが [2,2] ならば，高さ 2，幅 2 の領域内の最大値を返す．

0	**1**	2	3
3	**0**	1	2
2	3	0	1
1	2	3	0

⇒

3	2	3
3	3	2
3	3	3

`dropoutLayer` (**ドロップアウト層**) ：例にはないが，layers={　} の中の適当な場所に dropoutLayer(____,'Name',Name) を挟むことがしばしば行われる．ドロップアウト層は，与えられた確率でランダムに入力要素をゼロに設定する．これにより「過学習」を抑制することができる．

`fullyConnectedLayer` (**全結合層**) ：畳み込み層とダウンサンプリング層の後には，1 つ以上の全結合層を配置する．全結合層はニューロンが前の層のすべてのニューロンに結合している層．この層は，前の層によってイメージ全体で学習されたすべての特徴を組み合わせて，より大きなパターンを特定する．最後の全結合層は，これらの特徴を組み合わせてイメージを分類する (最後の全結合層の `OutputSize` パラメーターは，ターゲット データのクラスの数と等しくなる).

`softmaxLayer` (**ソフトマックス層**) ：最後の全結合層の後にソフトマックス層を作成する．これは各結合層の出力 y_i を，全結合層に関して正規化するもので，使用されるのはソフトマックス活性化関数

$$S(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$$

である．

`classificationLayer` (**分類層**) ：最後に分類層を置く．これはソフトマックス活性化関数によって各入力について返された確率を使用して，(交差エントロピー) 誤差を計算し，また出力結果からクラスの数を推測する．ここでいう誤差とは「出力が期待する出力とど

れだけ乖離しているか」の目安で，重みやバイアスなどのパラメータを調整する．

- **学習オプションの指定**
  ネットワーク構造の定義後，学習オプションを指定する．深層学習のネットワークに学習させる場合，学習の進行状況を監視するためのオプションを指定する．

  ここでは，初期学習率 0.01 を指定し， モーメンタム項（過学習を抑制するために入れる項）付き確率的勾配降下法 (SGDM) を用いて，ネットワークに学習させる．学習率とは，勾配降下法においてどのくらい勾配方向に比例した成分を混ぜるかという，ミクシングの割合．最初は小さく混ぜ，全体の変化に応じて割合を調整する．

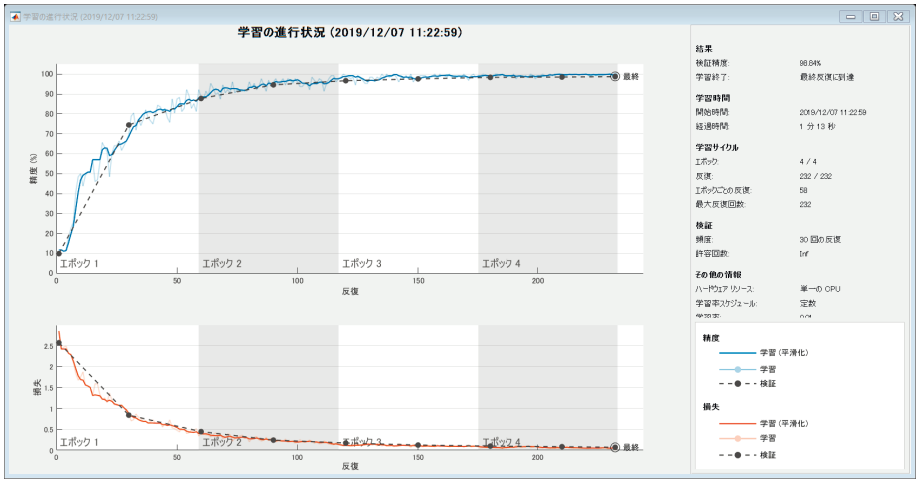
  学習データセット全体の学習サイクル（エポック）の最大回数を 4 に設定．検証データと検証頻度を指定して，学習中にネットワークの精度を監視する．学習データでネットワークに学習させ，学習中に一定間隔で検証データの精度を計算する指定.

```
options = trainingOptions('sgdm','InitialLearnRate',0.01, ...
    'MaxEpochs',4,'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, 'Verbose',false,'Plots',...
    'training-progress');
```

- **学習データを使用したネットワークの学習**
  `layers`，学習データおよび学習オプションによって定義されたアーキテクチャを使用して，ネットワークに学習させる．

  学習の進行状況プロットには，ミニバッチの損失と精度および検証の損失と精度が表示される． 深層学習における学習の進行状況の監視を参照.

```
net = trainNetwork(imdsTrain,layers,options);
```



- **検証イメージの分類と精度の計算**
  学習済みネットワークを使用して検証データのラベルを予測し，最終検証精度（ネットワークによって予測が正しく行われるラベルの割合）を計算する．

```
YPred = classify(net,imdsValidation);
YValidation=imdsValidation.Labels;
accuracy=sum(YPred == YValidation)/numel(YValidation)
```

```
accuracy = 0.9888
```

16.3.3　深層学習ネットワーク アーキテクチャの解析（analyzeNetwork ）

ここまでで，深層学習の実際を見てきました．
analyzeNetwork(layers) は，layers によって指定された深層学習ネットワーク アーキテクチャを解析します．

```
analyzeNetwork(layers)
```

関数 analyzeNetwork は、ネットワーク アーキテクチャを対話的に可視化し，各層の詳細情報を提供します．

第VII部

付録

付録１：文法について

A1.1 初歩文法の補足

A1.1.1 整数型と倍精度浮動小数点数

MATLAB では，デフォルトが倍精度浮動小数点数となっています．しかし，任意の数や数の配列を，整数あるいは単精度浮動小数点とすることもできます．そうすることにより，倍精度浮動小数点数に比べてメモリの節約になるからです．
・統計の項では外部データを読み込む際に，[double] を用いて倍精度浮動小数点数に変更しました．
・画像圧縮の項では実際，符号なし整数型から倍精度浮動小数点数に double を用いて変更しました．最後に結果を画像ファイルに直す際には，[uint8] を用いて符号なし整数型に変換しました．

A1.1.2 無名関数（anonymous function）

名前付けしないで定義された関数を，無名関数といいます（いろいろなプログラム言語で用いられます）．@演算子でひとまとめの関数（関数ハンドル）を作成します．@演算子の直後にある () で関数を定義する変数を定義します．@演算子の後ろには，実行可能ステートメントを 1 つしか含めることはできません．従って，そこで 1 度だけ使う関数に名前を付けなくて済みます．変数の引き渡しも直接行えます．

A1.2 グラフ上級編

A1.2.1 流れの表現

付録２：MATLABの有効な利用のために

A2.1 フォントの設定を変える

ダイアログ ボックスを使用し，デスクトップ ツールのフォントを変更するすることができます．このダイアログ ボックスにアクセスするには，[ホーム] タブの [環境] セクションで [設定] をクリックします．そこで [MATLAB]，[フォント] を選択します．

A2.2 MATLAB関数のプログラムの中身を見る

タイプコマンドを使い

type File_Name

とすると、MATLAB関数を含むプログラムファイルFile_Nameの中を全てる事ができます．

付録 3：教室での教育用ツール - MATLAB Drive と Live Scripts

私たちは，MATLAB を用いて，文科系の学生に対する数学に基礎講義をすることを試みています．工学部や理学部の学生に対しては，数学の講義にはおおむね演習の時間が付いています．一方，文科系の学部では，そこまでする時間的余裕がありません．
しかし，数学などの科目は，講義を聞いて後は自分で勉強しろというのでは，普通は無理があります．それを補助するためには，教師が講義で示す計算プログラムを学生が共有して，“追体験” する必要があります．そのための MATLAB の機能を紹介します．MATLAB Drive を利用して，プログラムコードの共有，教材の教室内での共有などができます．
共用ファイルの作成にはライブスクリプトを用います．

A3.1 MATLAB Drive の利用

プログラムコードの共有，教材の教室内での共有などを目的として，ファイルやドキュメントを共有するためには MATLAB Drive 上のクラウドサービスを利用するのが良いでしょう．

- **MATLAB Drive** を利用するためには，MATLAB Drive Connector をインストールします．あらかじめしておく準備はこれだけです．
- 接続には **MATLAB Drive Online** を利用します．サイト https://jp.mathworks.com/products/matlab-drive.html#explore-matlab-drive

から,「MATLAB Drive オンライン」のボタンをクリックすると,MATLAB Drive にログインできます.

- 共有フォルダーの所有者（作成者）は共有を許した相手の権限について，書き換え権限 ('Can Edit') または閲覧権限 ('Can View') を指定できます．Invitation はフォルダー内からのみ，フォルダー毎に行います.

- 以上で，PC からクラウド（MATLAB Drive）にファイルをアップロードし，あるいは貴方を招待したユーザーのファイルを共有することができます.

A3.2 MATLAB Live Editor の利用

ライブスクリプト（Live Scripts）とは，プログラムコード，出力およびテキストを一体化した環境を作り出すプログラムファイルです．これにより，教室その他で，プログラムを共有しながら他のユーザー（共同開発者や学生）と対話を行いつつ作業を進めることができます.
ライブスクリプトを作成するために，**ライブエディター（Live Editor）**モードを使います.

- 最初に [ホーム] タブにある [**新規ライブ スクリプト**] をクリックして起動します.

- 次にコマンドウィンドウに NewFile.mlx と入力すると，MATLAB Drive の中に NewFile.mlx というファイルが作成され，同時に**ライブエディター**が　NewFile.mlx　につながります.

- グレーな領域に式や計算手順を書いていけば，そのままスクリプトとなります．またライブエディター内の［**実行**］をクリックすれば，計算が実行されます．実行結果はプログラムコードの右側に出ます.

- テキストなどを書いてブログラムの説明やその他の記述をしたいときには，その場所にカーソルを置いて，ライブエディター内の［**テキスト**］をクリックすれば，テキストを記入する領域が開きます．先頭にテキストを書きたいときにはグレー領域の先頭にカーソルを置いて，以上の操作を行います.

［**Aa**］ボタンにより，平文かあるいはタイトルかその他かを選択できます．

- テキストを書いた後，［**コード**］をクリックすると再びスクリプトを書くためのグレー領域が開かれます．
- テキスト中には式も挿入することができます．式を書き入れるためには［**挿入**］ボタンをクリックし，［Σ 式］から式を対話的に挿入するか，Latex 形式で挿入するかを選択します．

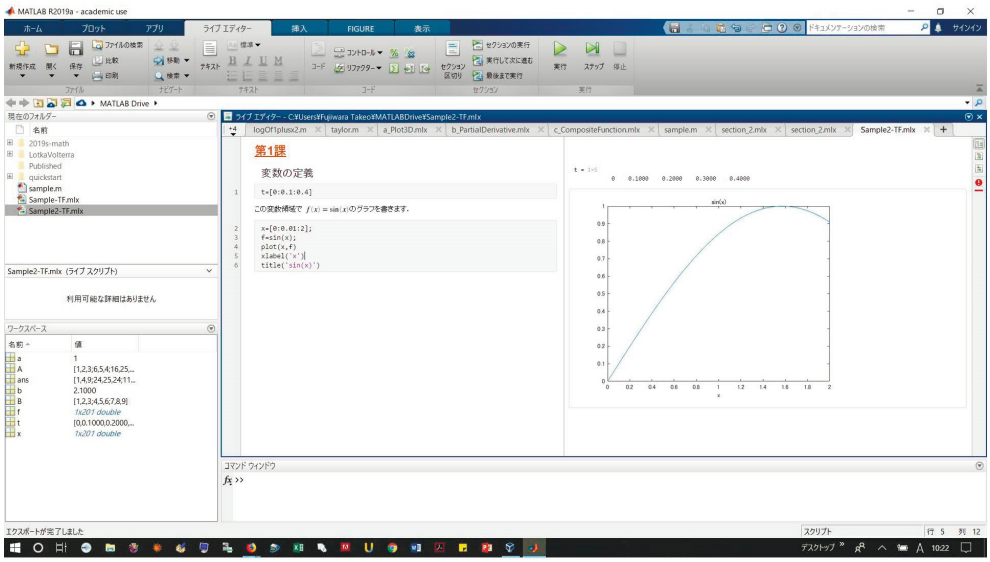


図 16.6: Live Script の利用．

付録 4：自動採点システム - MATLAB Grader

A4.1 MATLAB Graderの概略

MATLAB Grader を用いて，各教員が講義に対応した利用ができます．それらは，各講義（Course）に対応して，以下の構造を持つことができます．学生はMATLABのライセンスは不要です．

- **Course：** 各講義に対応して教師が設定します．
  - **Course Description：** 各講義の概要，内容等を記述します．
  - **Assignment：** 各コース内で設定されるいくつか（複数可能）の課題を設定します．
  - **Problems：** 各課題の中で与えられる問題を設定します．
    ここで教師は模範解答 (Reference Solution) を与えます．準備ができたら，学生に e-mail によりその旨の通知を出すことができます．
    学生が解答をしたら，システムはそれを模範解答と比較し,（自動）評価 (Assessment) します．システムは学生の解答を一元管理し，教師は学生の進捗状況の把握および分析結果を知ることができます．

A4.2 教員が利用するための準備手順

順にこれらの手順を見ていきます．

MATLAB Grader:https://jp.mathworks.com/help/matlabgrader/index.html
⇒

MATLAB Grader product page：
https://jp.mathworks.com/products/matlab-grader.html ⇒

1. **MATLAB Grader の紹介ビデオ**：上のページから “Guided Tour video” をクリック

2. **MATLAB Grader へのログイン**:https://grader.mathworks.com/

   - 「**For Instructors: Create Courses and Coding Problems**」
     https://jp.mathworks.com/help/matlabgrader/matlab-grader-for-instructors.html
     - 「**Create Course**」: コースの設定します．新規の場合には「**ADD COURSE**」を，既存の場合には「**コース名**」をクリック.
       ⇒「Markup in MATLAB Grader」書き方の詳細（字体, 図挿入他）
       (a) 「**Create Assignment**」: アサイントの設定します．新規の場合には「**ADD ASSIGNMENT**」を，既存の場合には「**アサイント名**」をクリック.
       公開 (Visible) 期間の指定は問題ごとではなく，Assignment 毎にここで指定する．
       課題提出回数を（Number of Submission で指定）制限できる．
       * 「**Add Problems**」: 問題作成と評価 (Assessments) のための模範答案作成．新規の場合には「Blank Problem」をクリック．問題をオープン，クローズする日をここで記入.
       * 「**Reference Solution**」に模範解答を記入．学生からは閲覧できない.
         「**Learners Template**」に学生の解答を誘導する枝問などを記入することができる．Lock 機能があり，これを掛けたものは学習者の書き換え禁止.
       * 「**Assessment**」各 **Test** 項目に評価項目を（テスト形式で）記入できる．**Pretest** にチェックマークを入れておけば，学生が事前にテスト項目を実行してそれが正答か誤答かを知ることができる.

∗ 問題の Edit 状態で下に表れるボタン：
(1) Learner Preview：学生からどう見えるかチェック.
(2) Validate Reference Solution：模範解答と評価テストの動作確認.
(3) Save as Draft と Save as Final：Draft の状態では学生には見えない.

(b) 「**Manage People**」：受講者あるいは共同運営者を招待する．mail アドレスを記入．相手先にメールが届く.

以上が MATLAB Grader の概略です.

上の「**For Instructors: Create Courses and Coding Problems**」ページから繋がる「**Add Problem**」ページおよびその下にある以下も参考になります.

(a) 「**Create Script-Based Problems**」：
書き方や「Problems」の仕組みの詳細.

(b) 「**Create Function-Based Problems**」：

(c) 「**Assessments**」：評価のための文の書き方など

- Testing Learner Solutions
- Write Assessments for Script-Based Learner Solutions
- Write Assessments for Function-Based Learner Solutions

以上を参考にして，実際に使ってみることをお勧めします．総ての機能を使うのではなく，それぞれの方が好みの使い方を見つけて下さい.
また私自身が実際に使用してみると，課題の与え方や答え方に規則があり，特に課題の与え方に注意が必要です．何でもできると考えずに，今後のプログラムの整備に期待したいと思います.
マニュアルの構造も，もうすこし整理が必要だと感じました.