Octaveの精義

【第二版】

フリーの
高機能数値計算ツールを
使いこなす

松田七美男◉著

CUTT
カットシステム

■**サンプルファイルのダウンロードについて**

本書掲載のサンプルファイルは、一部を除いてインターネット上のダウンロードサービスからダウンロードすることができます。詳しい手順については、本書の巻末にある袋とじの内容をご覧ください。

なお、ダウンロードサービスのご利用にはユーザー登録と袋とじ内に記されている番号が必要です。そのため、本書を中古書店から購入されたり、他者から貸与、譲渡された場合にはサービスをご利用いただけないことがあります。あらかじめご承知おきください。

・本書の内容についてのご意見、ご質問は、お名前、ご連絡先を明記のうえ、小社出版部宛文書（郵送またはE-mail）でお送りください。
・電話によるお問い合わせはお受けできません。
・本書の解説範囲を越える内容のご質問や、本書の内容と無関係なご質問にはお答えできません。
・匿名のフリーメールアドレスからのお問い合わせには返信しかねます。

はじめに

筆者は，工学部の物質科学系の学科でいわゆる「計算物理」の講義を担当していました．プログラミング言語としては最初 C 言語を採用していたのですが [1,2]，高校に教科として『情報』がある今日においても，一般の学生がある程度自由に使えるようになるまで時間がかかり過ぎ，何かもっと良い言語はないものかと悩んでいた頃，Octave を知り，直ぐに気に入って使うようになりました [3]．Octave が元々学部レベルの学生への化学反応論の講義の補助ソフトとして書かれたものであることを考えると，筆者が直感的に『これだ』と思ったのも無理からぬことと思います．Octave はその後 GNU のプロジェクトとなり大きな発展を遂げつつありますが，当初の性格は引き継がれ，いわゆるプログラム言語に比べて格段に使いやすいツールであることは間違いありません．[*1]

数値計算ツールといえば，商用のものでは，Mathematica, Maple, MATLAB がよく知られていますが，Octave は行列計算に優れた MATLAB との互換性を考慮して開発が進められています．その互換性は入門レベルのコマンドについてはかなり高いです．Octave に関する書籍 [4–8] はまだあまり多くないのですが（そこに本書を出版する意義があるともいえます），MATLAB については優れた入門書が出版されておりたいへん参考になります [9–18]．さらに，MathWorks 社のウェブサイトからは，MATLAB の計 5000 ページに及ぶマニュアルをネットを通じて入手することができます．筆者は，今回 Octave のマニュアルを読んで疑問に思った事柄を MATLAB のマニュアルを調べて解決した経験を随分としました．

Octave の標準配布には，以下のような項目に対する数値解法が提供されています．この書籍はバージョン 5.1.0 に基づいてますが，MATLAB よりも少ないのはやむを得ないところでしょう．

> 複素数行列の計算・関数，多項式，非線形方程式，線形連立方程式，常微分方程式，数値積分，最小二乗法，数理統計，スパース行列，グラフ理論

残念ながら，シンボリック計算は標準配布されていません．OctaveForge というパッケージの開発版を集めているサイトがあり，そこにシンボリック計算のパッケージがありますが，実用的とはいえないレベルです．どうしてもという方は，オープンソースなシンボリック計算ツール MAXIMA [19–21]，REDUCE [22] などを試してみるとよいでしょう．

それにしても，Octave の守備範囲は十分広くマニュアルは 1000 ページを越します．したがって，いくら『精義』といっても全てを紹介はできません．また，筆者は Octave を主として常微分方程式の計算とその結果のグラフ化に使っているだけで，他の項目については詳しく説明できそうもありません．初版では「一通り関数の説明をする」と豪語したの

[*1] Octave の歴史については https://www.gnu.org/software/octave/about.html をご覧ください．

ですが，今回はもう手に負えないと諦めて，筆者が最も関心のある gnuplot との連携という観点から，**1. 結果のグラフ化についてはできる限り説明する**．さらに，煩わしいと感ずるかもしれませんが **2. 原則的にはスクリプトを必ず示すことにする**．なぜなら「**百聞は一見に如かず**」のとおり，私の経験では，スクリプトの現物が役立つことが多かったからです．以上 2 つの大方針を初版から引き継いで改訂を試みました．更に，この改訂版では筆者の個人的な事情ですが，講義を担当することになった基礎物理学（主として力学）での利用例を追加しました．

本書で用いた記号や表現について

上記の方針にしたがって，本書ではスクリプトのリストを以下のような**左側に横線を引いた体裁**で示します．スクリプト以外に関数などの書式を示す場合にも用いて強調します．このスクリプトは graphics のデモ関数 `peaks` の等高線と等高線の値を表

```
graphics_toolkit("gnuplot");

[x,y,z] = peaks(96);
[C,H] = contourf(x,y,z,[-7:1:7]);
colormap("summer")
clabel(C,H)
axis([-3,3,-3,3], "square")
title("Filled contour", "fontsize", 18)
xlabel("x", "fontsize", 14, "fontangle", "italic")
ylabel("y", "fontsize", 14, "fontangle", "italic")

print("contour_g.pdf","-S360,360")
system("evince contour_g.pdf")
```

contour_g.ovs

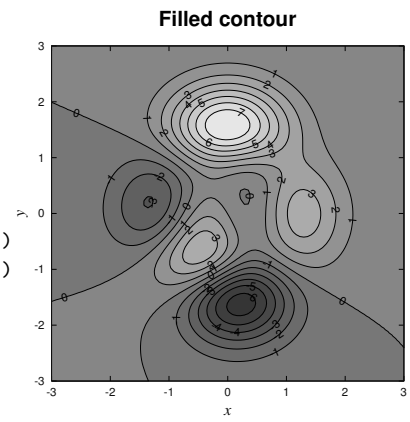

図 1　地図によく利用されている等高線図を描くスクリプトと得られた図面

示し，印刷用の PDF（Portable Document Format）形式のファイルも作成するものです．PDF 形式の図面ファイルは，拡大縮小によって図の品質の劣化がないよう設計されており，一つのファイルだけで，後に色々な（表示される際の大きさが異なる）場面で利用できるので筆者はとても重要視しています．本書に掲載されるスクリプト例には最後の 2 行にそのための印刷命令と出来上がった PDF を画面表示して確認する命令

```
print("***.pdf", "-Sxsize,ysize")
system("evince ***.pdf")
```

pdf ファイルの作成と，evince を起動して画面確認を行う命令

が含まれます．不要な場合は，**行頭に#記号を追記**して最後の行あるいは 2 行ともコメント

アウトし，替わりに以下の pause を加えてください．

```
pause
```

一時停止命令

すると Octave は一時停止して図が画面表示されたままになり，何らかのキー入力をもって終了するようになります．また，冒頭にも次のようなコマンドが記されている場合がありますが，それは以下の事情（と筆者のこだわり）によるものです．

```
graphics_toolkit "gnuplot";
```

グラフィックエンジンとして gnuplot を指定する命令

即ち，Octave は図面作成のエンジン（Octave では graphics toolkit と呼ばれる）として永らく gnuplot を利用していましたが，バージョン 4 からは，Qt ライブラリ（ノルウェーの Toroltech 社が開発元，Nokia 社に吸収された）の上の独自エンジンを既定にしました．Qt を用いることで画面上の像と種々の形式で保存（印刷）した像との一致が良くなりました．しかし，いまだに gnuplot を利用した方が品質の高い印刷（フォント制御や塗り潰しの細かさ）が得られる場合があるので（PDF の仕上がりを見て判断します），その場合には gnuplot を描画エンジンに指定するためのコマンドを最初に置くことにしたのです．

煩わしいというのであればコメントアウト，或いは削除してください．すると，Qt を用いた図が作成されます．スクリプトは，画面上の端末の**コマンドライン**から

```
$ octave contour.ovs
```

とキー入力して実行させます．作成された PDF ファイルがグラフィックエンジンが Qt か gnuplot かによって，若干異なることを図 2 に示します．

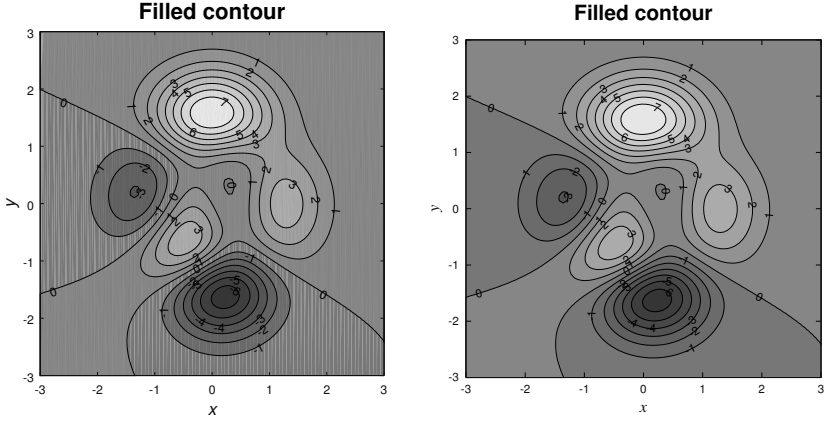

図 2　使用する graphics toolkit に応じて作成される PDF の相違：既定の Qt （左図）に比べて gnuplot（右図）の方が品質が高い場合がある．

■ **紙面節約のための空行の削除**　Octave の端末画面出力は見易さを考慮して空行が含まれます．しかし，それをそのまま紙面に掲載すると，ちょっと間延びして見えます．そこで，表示を簡素にするために書式を以下のように指定しています．

```
format compact
```

画面表示書式を簡素にする命令

こうして紙面の節約を果たしたという訳です（図 3）．

```
octave:46> A = eye(3)
A =

Diagonal Matrix

   1   0   0
   0   1   0
   0   0   1

octave:47> A = sin(A)
A =

   0.84147   0.00000   0.00000
   0.00000   0.84147   0.00000
   0.00000   0.00000   0.84147

octave:48>
```

```
octave:> A = eye(3)
A =
Diagonal Matrix
   1   0   0
   0   1   0
   0   0   1
octave:> A = sin(A)
A =
   0.84147   0.00000   0.00000
   0.00000   0.84147   0.00000
   0.00000   0.00000   0.84147
octave:>
```

図 3　画面出力書式指定 compact による，紙面節約効果：既定（左図），compact 指定後（右図）．

■ **ベクトルの記号**　本文中で行列は A, P, U のように大文字のイタリック体で表現し，縦ベクトルあるいは列ベクトル $\boldsymbol{b} = (b_1, b_2, \cdots, b_n)^T$ をイタリックボールド体で表現します．この表現を用いて，$m \times n$ 行列を長さ m の列ベクトルを要素とする長さ n の横ベクトルとして以下のように表現する場合があります．

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \Big(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n \Big)$$

また，横ベクトルあるいは行ベクトルは縦ベクトルの転置として $\boldsymbol{a}^T$ のように表現します．

バージョン 5 の気になる新しい機能について

■ **日本語表示：ユニコード対応**　ユニコードによる日本語の表示については，ディスプレイに現れる表示窓では図 4 のようにバージョン 4 から対応がなされていました．print に

よる印刷物にはそれが反映されず，図 5 の左図のように文字化けを起こしていましたが，バージョン 5 では jpg などのビットマップでは図 5 の右図のように文字化けしなくなりました． ただし，筆者が重要視している pdf やその前身である ps は対象外であり，相変わ

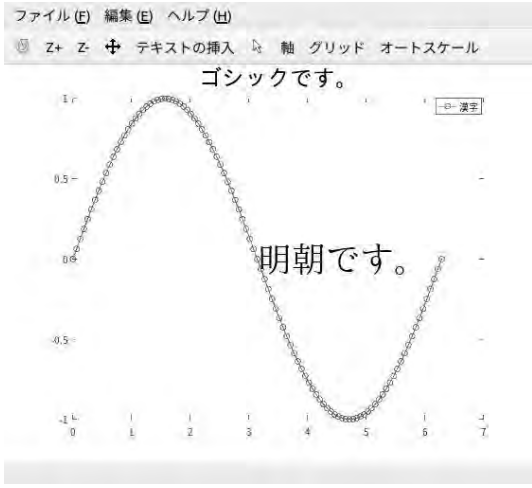

図 4　画面に現れるグラフィックダイアログについては，ユニコードによる日本語の表示がバージョン 4 から実現しています．利用できるフォント名はテキスト挿入のダイアログを立ち上げリストボックスで確認してください．

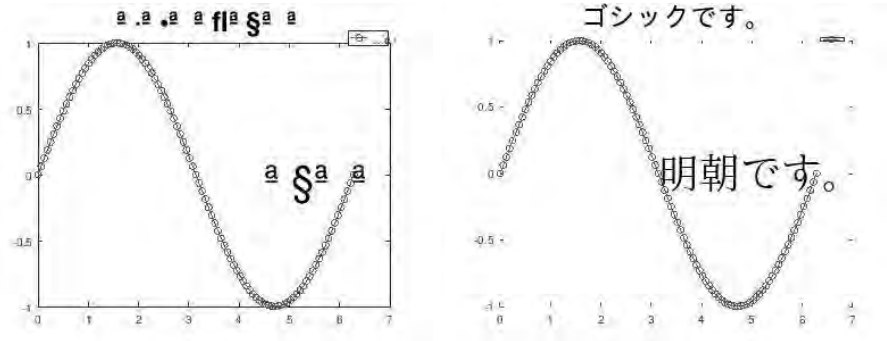

図 5　`print` によるビットマップの印刷物は，バージョン 4 では日本語は文字化けしてましたが（左図），バージョン 5 からは正常にレンダリングされています（右図）．

らずグラフィックエンジンを gnuplot にしてフォント名も ghostscript が理解するものに指定しないとなりません．この書籍では，日本語の表示は原則させてませんが，例外的に日本語を表示させる場合にはバージョン 4 と共通である，gnuplot を利用するコードを例示しています．

■ **統合環境 GUI**　この本の初版を著した頃の Octave のメジャーバージョンは 3 でした．現在メジャーバージョンは 5 となり，図 6 のような GUI 統合環境版が起動できるようになりました．実行，編集，ヘルプの 3 つのタブを，一つの窓の中で切り替えて作業をすることができ，初心者には大変便利な環境が整備されました（図 7 参照）．

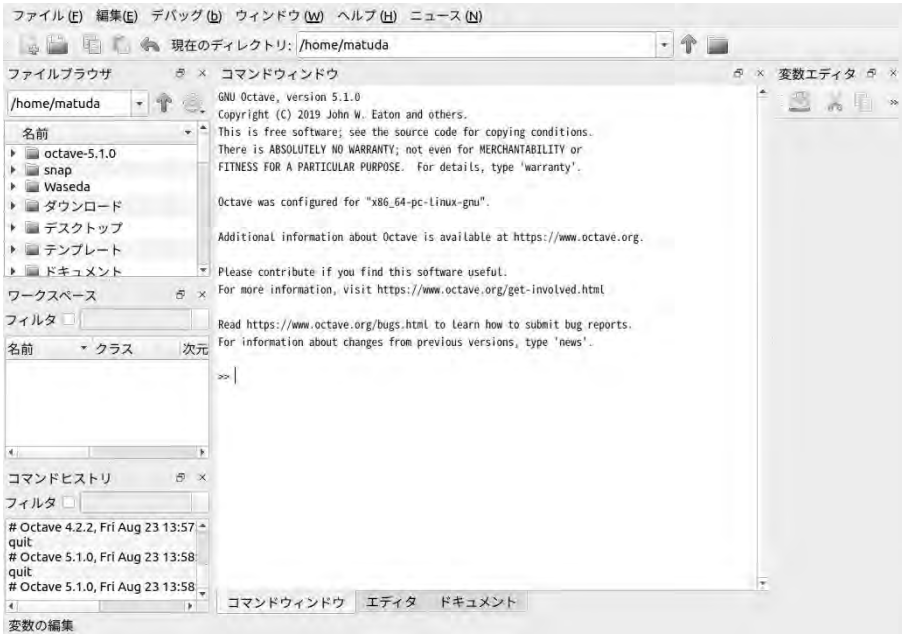

図 6　統合環境の起動時画面．既定ではコマンドラインタブが前面に現れている．

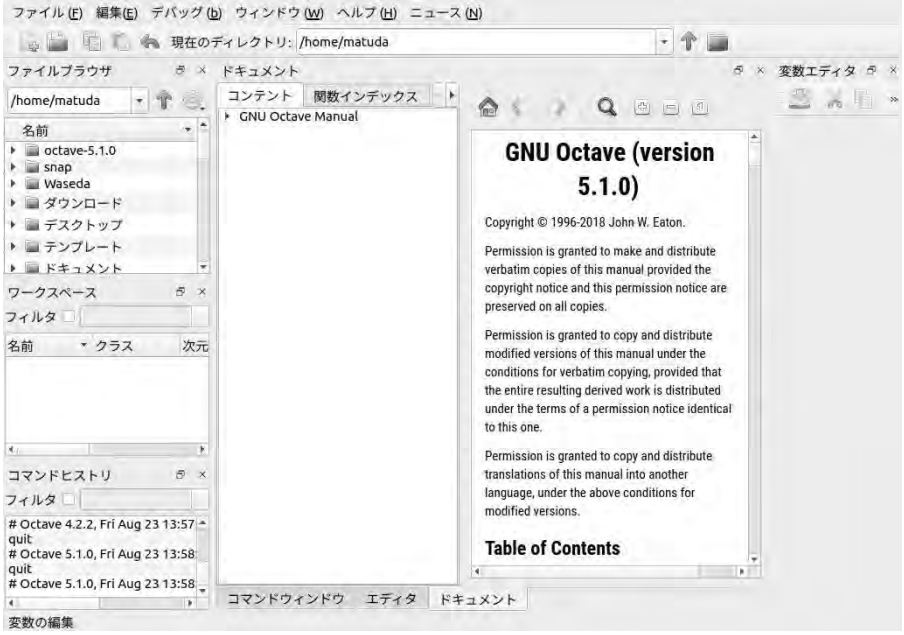

図 7　統合環境のドキュメントタブの切り替えの様子．目次や索引付きのドキュメントを直ぐに読めるので，とても便利になりました．

ところでこのような進歩を素直に喜べないのが，筆者の性格でして，余分な窓は開かず，古えの文字端末インターフェースで作業をする習慣から抜け切れません．そこで CLI を指定するオプション--no-gui を付けて起動しています．

```
$ octave --no-gui
```

実はバージョン 5 では，既定が CLI の起動というバージョン 3 の挙動に戻りました．GUI を起動する場合には，--gui をオプション指定する必要があります．

本書の初版が Octave の解説書として発刊されたのは平成 23 年の 1 月でした．以来 8 年の間 Octave は進化を続け，メジャーバージョンも 3 から 5 へと変化しました．筆者といえば，化学・物理・生物を学ぶ学科から共通教育系列に移って，初年次の基礎物理だけを担当することになってしまい，専門科目で Octave を使った授業展開ができなくなってしまいました．専ら隔年の大学院の講義や色々な問題作成の場面での検算（もう 3 桁の割り算も危ない年齢となりましたので）にしか Octave を利用することがなくなり，改訂版の要望を再三再四，カットシステムの石塚勝敏社長から受けながらもなかなか執筆が進みませんでした．しかし，初版には間違いが散見されることも承知していましたので，その箇所を修正した改訂版を一回くらい出さないでは，執筆者としての責務が果たされないであろうと決断し，本書が出来上がりました．

以上のような経緯ですので，バージョン 5 の新しい機能への記述はあまりありません．それは単に筆者が怠けていたということよりも，必要とする数値計算ツールとしての機能について大きな変更がなかったからというのが根本的な理由です．それではあまりなので，基礎物理学や他大学での C 言語の講義で使ってみた具体例を追加しました．初版でも述べましたが，一介のユーザーに過ぎない筆者がこのような書を執筆したのは，なんといっても Octave に魅力があり，是非普及して欲しいと願ったからです．そのような魅力ある Octave を開発し続けている John W. Eaton 氏および多くの開発者の方々，そして開発の場を提供してくれている GNU プロジェクトに敬意の念を表するとともに感謝申し上げます．

令和元年 松田七美男

目次

第 1 章

基礎

Octave はコマンドひとつあるいは 1 行（80 文字くらい）で数値解を求めるという究極の使い方も可能です．しかし，一般的な状況から少しはずれたような問題に対しては自分でプログラムを作成しなければなりません．そのためには Octave の文法の知識とその実習が必要です（あらゆるプログラミング言語において，避けて通れない必須事項ですね）．そこで，やはり第 1 章に基礎的な事柄を，なるべく実行例を交えて整理することにしました．実行例のない瑣末な関数の説明も所々混じっていますが，それは辞典としても使えるようにと意図したからです．

1.1 入門

手軽に数値計算ができるということが最大の特徴ですから，どのくらい簡単なのかを Octave のマニュアルの『A Brief Introduction to Octave』に準じて紹介します．

1.1.1 起動と終了

Octave を**対話的に使う**には，GUI ならばメニューやデスクトップ上のアイコンをクリックします．すると統合環境版が起動します．しかし，「はじめに」でも述べましたが，この本では**端末**エミュレータ上のコマンドライン上で動かすこと（Command Line Interface: CLI）を想定しています．バージョン 5 では既定が CLI 起動というバージョン 3 の挙動に戻りましたが，明示的に CLI を起動するため，

```
$ octave --no-gui
```

と入力します．

```
$ octave --no-gui     <- キーボードを使って入力して Enter キーをおす
GNU OcGNU Octave, version 5.1.0
Copyright (C) 2019 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1>             <- プロンプトの表示：命令の入力を待っている状態
```

ここで，実行画面例の最初の行頭の $ はシェルのプロンプトで既に表示されているはずですから，キー入力はしないでください．シェルのプロンプトはもう少し長いかもしれませんし，もしかしたら # となっているかもしれません．すると，Octave が長々とメッセージを表示して，プロンプトを表示します（統合環境版ではプロンプトは単に '>>' となります）．この状態になったら命令を入力していきます．まだ，何もしていませんが，プロンプトが表示されていれば，quit で正常に終了できます．

```
octave:1> quit        <- 終了の命令   quit の入力

$                     <- シェルのプロンプトが表示され，シェルへの命令待ち
```

さて先ほどのメッセージは 1 度は読んでもいいですが，あまり見たくないでしょうから，抑制するオプション -q を付けて起動してみましょう．

```
$ octave -q --no-gui
octave:1>              <- グリーティングメッセージなしですぐに命令待ち
```

ところで，何か操作を誤って（あるいは滅多にないと思いますが Octave のバグのせいで）プロンプトが表示されない状態に陥ることがあるかもしれません．その時には，**強制終了**を行うしかありません．GUI になれると，窓枠にある終了ボタンをおす（これは端末の強制終了です）癖がついていると思いますが，それはできれば最後の手段に取っておきましょう．端末が活きている間は，Ctrl キー と C キーを同時におすことで，端末上で動いているアプリケーション（ジョブ）を終了させることが可能です．

1.1.2 簡単な例

1.1.2.1 行列の生成

新たに行列を生成して，ある変数に内容を保存（代入）し，後で呼び出せるようにするには，プロンプトに対して以下のように命令します．

```
octave:1> A = [ 1, 2, 3; 4, 6, 8; 9, 10, 12]
A =
    1    2    3
    4    6    8
    9   10   12
```

3×3 の行列が出来上がり変数 A に代入されました．さらに，すぐに結果が表示されます．もし結果の表示が不要ならば**セミコロン** ； を行末に付けてください．

```
octave:2> B = eye(3);
```

行列 B は生成されますが結果は表示されませんでした．B を呼び出すには，単に B と入力します．

```
octave:3> B
B =
   1   0   0
   0   1   0
   0   0   1
```

確かに，B が生成されていました．生成時に呼び出した関数 eye() は結果を見てわかるように，単位行列を生成するものです．

1.1.2.2 行列の演算

Octave は行列の演算を教科書にあるような記述で命令することができます．

```
octave:4> 2*A
ans =
    2    4    6
    8   12   16
   18   20   24
octave:5> A*B
ans =
    1    2    3
    4    6    8
    9   10   12
```

```
octave:6> A'*A
ans =
    98   116   143
   116   140   174
   143   174   217
octave:7> A^(-1)
ans =
    4.0000   -3.0000    1.0000
  -12.0000    7.5000   -2.0000
    7.0000   -4.0000    1.0000
```

教科書の記述法であれば，それぞれ，$2A$，AB，A^TA，A^{-1} を表しています．もちろん記述法は完全には一致しません．たとえば掛け算のつもりで単に AB と書くと Octave には変数 AB と解釈されてしまいますから，どうしても掛け算を表す記号（演算子 * ）が必要です．これはプログラミング言語共通の記法です．

1.1.2.3 連立一次方程式

線形連立方程式とも呼ばれる 1 次方程式の連立問題は係数行列 A と定数ベクトル $\boldsymbol{b}$ を用いて表現され，解ベクトル $\boldsymbol{x}$ が以下のように求まることになっています．

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

もちろんこれは理論上の大変美しい結果ですが，実際の計算はこの通りにはいきません．この単純な解を求めるための数値計算上の研究が今でも続けられています．数値計算では，『逆行列を左から掛ける』という理論上の方法をそのまま実行しません（例え逆行列が求まるとしても）．そのかわりに，工夫された（安定，効率的な）数値計算を行って解を求めます．それは**左除算**と呼ばれ `A\b` のようにバックスラッシュ記号 \ を用いて表わされます．

```
octave:8> b = [2; 1; 4];
octave:9> x = A\b
x =
    9.0000
  -24.5000
   14.0000
```

```
octave:10> A*x
ans =
   2.00000
   1.00000
   4.00000
```

1.1.2.4 常微分方程式

次のような**陽形式常微分方程式**の初期値問題のためのソルバーは，MATLAB と異なり Octave ではたった一つの関数，`lsode()` が標準実装されています．

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t), \quad \boldsymbol{x}(t = t_0) = \boldsymbol{x}_0$$

`lsode()` を使うためには，常微分方程式の定義関数と初期値，計算値を出力する `t` の値のベクトルなどを予め定義しなければなりません．かなり大変ですが，間違えないようにキー入力してください．キーワード `function` で始まる関数の定義では，キーワード `endfunction` が入力されるまでは改行しても，継続のプロンプト `>` が表示され続けます．

```
octave:11> function xdot = f (x, t)
> r = 0.25; k = 1.4; a = 1.5; b = 0.16; c = 0.9; d = 0.8;
> xdot(1) = r*x(1)*(1- x(1)/k) - a*x(1)*x(2)/(1 + b*x(1));
> xdot(2) = c*a*x(1)*x(2)/(1 + b*x(1)) - d*x(2);
> endfunction
octave:12> x0 = [1; 2];
octave:13> t = linspace(0, 50, 200)';
octave:14> x = lsode(@f, x0, t);          <- 旧形式 lsode("f", x0, t) も可
```

もし，関数定義に入力ミスがあった場合には最初の 5 行の定義をやり直さなければなりません．これは対話的な使い方で最も悲しい結末ですが，あとで述べるように既に入力した行を呼び出して編集できるので，全てを打ち直すという最悪の事態は避けられます．最後

の行で常微分方程式を解き，その 200 × 2 の行列は変数 x に保存されています．数値を見てもさっぱりなので非表示にしました．結果をグラフに表す方法は次の節で紹介します．

なお，lsode() は A. C. Hindmarsh 氏が開発した，“the Livermore Solver for Oridinary Differential Equations” を採用しています [23].

1.1.2.5 グラフィカル出力

先ほどの常微分方程式の解を画面に表示するには plot() を用います．

```
octave:15> plot(t, x);
```

GUI 環境であれば，図 1.1 のように別窓が開いてグラフが表示されます．

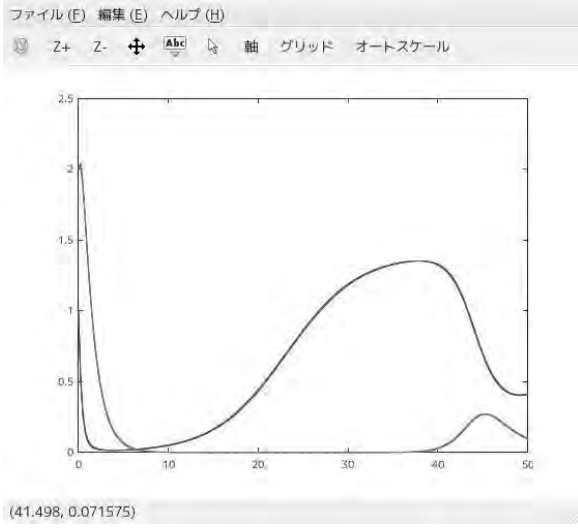

図 1.1 plot() により画面上に現れた，常微分方程式の解を図示している窓．

画面上に表示された図を印刷（ファイルに出力）するには，print を使います．

```
octave:16> print("foo.pdf","-S400,300");
```

すると，foo.pdf という PDF(Portable Documnt Format) 形式のファイルが出来上がります．大きさを"-S*xsize,ysize*"などと指定しないと，A4 用紙の中央にやや余白たっぷりの図が作成されます．『はじめに』で説明したようにこの単純な命令では筆者の望みに叶った図はできません．気に入った図を得るには，オプションをいろいろ試す必要があります．help print としてオンラインヘルプを読んでみましょう．

```
octave:17> help print
'print' is a function from the file /usr/local/share/octave/3.2.4/m/plot/print.m

 -- Function File:  print ()
 -- Function File:  print (OPTIONS)
 -- Function File:  print (FILENAME, OPTIONS)
 -- Function File:  print (H, FILENAME, OPTIONS)
     Print a graph, or save it to a file

     FILENAME defines the file name of the output file.  If no filename
     is specified, the output is sent to the printer.

...(以下略)...
```

Enter キーで 1 行送り，Space キーで 1 画面送り，Q キーでオンラインヘルプが終了し，Octave のプロンプトに戻ります．

1.1.2.6 固有値問題

以下のような行列の固有値問題に対しては `eig()` が実装されています．

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

`eig()` の戻り値は 2 つあって，固有ベクトルと固有値を対角要素とする対角行列です．Octave では関数からの複数の戻り値をブラケット `[…]` で囲って受けとることができます．これも大変柔軟性の高い機能です．

```
octave:18> [V L] = eig(A)
V =
  -0.18420  -0.47541   0.31987
  -0.52048  -0.51818  -0.83086
  -0.83377   0.71097   0.45535
L =
Diagonal Matrix
   20.230850           0           0
           0   -1.306516           0
           0           0    0.075666
```

```
octave:19> A*V(:,1), L(1,1)*V(:,1)
ans =
   -3.7265
  -10.5298
  -16.8678
ans =
   -3.7265
  -10.5298
  -16.8678
```

固有値対角行列 `L` の i 番目の対角要素，すなわち固有値に対応する固有値ベクトルが `V` の第 i 番目の列ベクトルとなっています．

1.1.2.7 非線形方程式

人が手で解ける方程式は 2 次方程式くらいのもので，三角関数や指数・対数関数などが含まれる超越方程式は特別な場合を除き解を求めることはできません．かなり複雑な方程式であっても関数表現されている場合には `fsolve()` が（反復法を用いて）数値的に解を求めてくれます．関数定義の他に適切な初期推定値が必要です．

```
octave:20> fsolve(@(x) 2*sin(x) - x, 3)
ans =  1.8955
octave:21> 2*sin(ans) - ans
ans = -4.3012e-08
```

`@(x)` は匿名関数と呼ばれ，その場で関数を定義する場合に用います．もちろん，1 行で定義できるような簡単な構造のものならばこの記法が非常に役立ちます．複雑なものは `function ... endfunction` で関数定義せざるを得ません．得られた結果は保存先の変数を指示しなかったので変数 `ans` に保存されますので，それを使った検算を実行してみました．マニュアルによれば打ち切り誤差の既定値は 1×10^{-7} ですが，得られた解の $f(x)$ 値はそれを下回っており条件を満足しています．

1.1.2.8 複素数の計算

複素数を数学の教科書にあるような記述形式で表すことができますし，もちろん数学関数も原則的に複素数に対応しています．

```
octave:22> z = 3 + 4i
z =  3 + 4i
octave:23> [conj(z), z^2, sqrt(z); sin(z), exp(z), log(z)]
ans =
    3.0000 -  4.0000i   -7.0000 + 24.0000i    2.0000 +  1.0000i
    3.8537 - 27.0168i  -13.1288 - 15.2008i    1.6094 +  0.9273i
octave:23> [real(z), imag(z), arg(z), arg(z^2), abs(z), abs(z^2)]
ans =
    3.00000    4.00000    0.92730    1.85459    5.00000   25.00000
```

1.1.3 コマンドの行編集

打ち込んだコマンドは履歴として残り（history 機能），Unix の 2 大エディター **Emacs**, **vi** に準じたライン編集ができます．キー割り当ては Emacs のそれが既定となっています．すなわち，上下のカーソルキー（ Ctrl – P , Ctrl – N キーも可）でコマンドを呼び出すことができます．さらに，呼び出された行の内容は左右カーソルキー（ Ctrl – B , Ctrl – F キーも可）で移動して編集することができます（編集を確定するにはエンターキーの入力が必要です）．

1.1.3.1 ヘルプ機能

もう既に述べた help の他に，オンラインヘルプとしては lookfor，doc があります．lookfor XXX は文字列 XXX を含む項目を検索してとても簡単な説明を一覧します．doc はマニュアルと同じ内容を詳細表示します（Unix 系ならば info を使います）．GUI 統合環境版では「ドキュメント」タブが用意されており，目次もありますから参照が楽になりました．筆者も，この点についてだけは統合環境版の優位性を認めざるを得ません．

■ example　ある関数のコードの見本を表示してくれます．例えば，2 次元グラフを描く基本的な関数 plot() については，以下の実行画面のように表示されます．

```
octave:14> example plot
plot example 1:
 x = 1:5;  y = 1:5;
 plot (x,y,"g");
 title ("plot() of green line at 45 degrees");

... (中略) ...

plot example 8:
 x = 0:10;
 shape = [1, 1, numel(x), 2];
 x = reshape (repmat (x(:), 1, 2), shape);
 y = rand (shape);
 plot (x, y);
 axis ([0 10 0 1]);
 title ({"Two random variables", "squeezed from 4-D arrays"});
```

■ demo　文字通り，コードの見本をデモしてくれるコマンドが demo です．コードの見本自身も表示されます．3 次元メッシュの表面を描画する関数 surf の場合の実行画面を以下に示します．

```
octave:15> demo surf
 clf;
 colormap ("default");
 Z = peaks ();
 surf (Z);
 title ({"surf() plot of peaks() function"; "color determined by height Z"});

Press <enter> to continue:
```

画面には，次のようなグラフが順次表示されます．

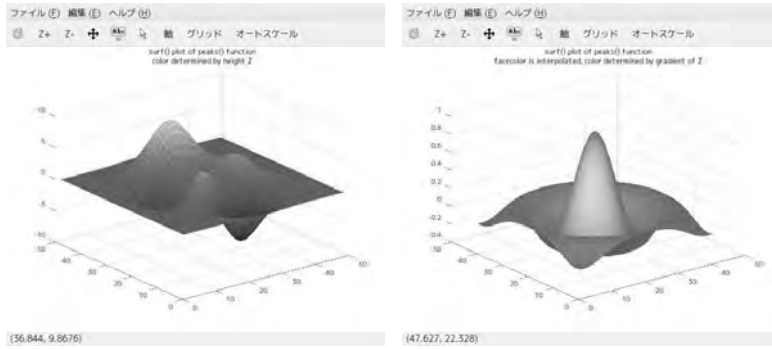

図 1.2　`demo surf` を実行した際に現れるデモグラフ.

1.1.4 非対話モード

1.1.4.1 一行コマンド

例えば $\sin(15°)$ を計算したいなどというとき，一々 Octave を起動させるのは億劫です．そのような場合には，評価する命令文を与えるオプション`--eval` を用いて以下のように，所謂「一行コマンド」として実行させることができます．

```
$ octave --eval 'sind(15)'  <- シェルから起動
ans =  0.25882
$                           <- 直ぐにシェルに戻る
```

もう少し複雑な問題の解，例えば超越方程式 $\cos(x) - x = 0$ の数値解なども

```
$ octave --eval 'fsolve(@(x)cos(x)-x,0)'
ans =  0.73909
```

と簡単に求まります．筆者が CLI に拘る最も重要な理由の一つがここにあります．

1.1.4.2 スクリプトファイル

一般に，複雑な問題を解く場合には長いプログラムを書く必要があります．その場合には，対話モード上で修正を繰り返して完成させることは相当難しいです．長いプログラム *`script`*`.ovs`[*1] をスクリーンエディター上で編集し，以下のように命令して，Octave に実行させることができます．

```
$ octave script.ovs
```

そして，「不具合があったら，エディターでその箇所を修正し，再実行する」の繰り返しで

[*1] 拡張子は `m` を用いるのが普通ですが，それは汎用性が高く関数化する意味のあるものに留めるがよいと考えられます．本書では，その場限りのものには `ovs` を拡張子として用いています．

プログラムを完成させます.

■ edit　対話モードから EDITOR() で登録された外部エディタを起動して，スクリプトファイルを編集することができます．一般には emacs が登録されていますが，軽量なスクリーンエディタ，例えば leafpad を使いたければ，EDITOR('leafpad') と登録をやり直してください.

■ source，run　対話モードにおいては，拡張子.m のついたいわゆる **M–ファイル**は，拡張子を除いた名前だけで呼んで実行させることができます．異なる拡張子，例えば本書のように.ovs のついたスクリプトも，source や run の引数に PATH を含めた完全な名前を与えて実行することができます.

```
source("name.anyext")
source "name.anyext"
run("name.anyext")
run "name.anyext"
```

小括弧をつけた関数形式で呼び出すよりも，コマンド形式で実行する方がキー入力が少なくて済みますから楽です.

1.1.5　コマンドと関数

run は，run "*script*" のように記述して，引数として与えられたスクリプト *script* を実行するコマンドですが，括弧を付けて関数呼び出し run("*script*") もできます．Octave の組み込み関数にはコマンド風の記述が可能なものが数多くあります．一方数学関数は，コマンド風の呼び出し sin 1 はエラーとなります．またユーザー定義関数も関数呼び出ししかできません．本書では関数呼び出ししかできない場合には名前の後ろに () を付けて区別することにしました．なお，コマンド風の呼び出しが可能ではあるが（エラーとはならない），関数と同じ機能をしないものもあります．例えば，plot() は

```
octave:> plot x, sin(x))
```

とすると，図面窓（fig と呼ばれるオブジェクト）が現われますが，データがないというメッセージも出力され，$(x, \sin(x))$ のプロットは描かれません．関数呼び出しでしか機能が発現しない場合も名前の後ろに () を付けます.