数値解析（松尾）　予備資料　　**2008年12月5日**

数値計算のやり方

自宅のパソコン，もしくは教育用計算機センターの端末を使って数値計算する方法について簡単に説明する．なお類似の情報は，講義ＨＰ：http://www.sr3.t.u-tokyo.ac.jp/~matsuo/lecture/08suchi/ にもあるので，そちらも参照すること．

1　数値計算の場所とツール

まずは自宅等の計算機に自分でツールをインストールして数値計算する場合．数値計算・プログラミングのツールには，講義初回に紹介したような各種のものがあるが，無償のもので使いやすいのは，大体次の３つである．ただし後ろの２つは計算機に関する知識・経験をある程度前提とするため，ここでは URL などは載せない.（できる人は自力で検索して開発環境を整えること）

- Scilab: http://www.scilab.org/
  →商用の数値計算ソフト MATLAB にほぼ互換の，フリーの数値計算ソフト．プロの研究には MATLAB を使うが，ちょっとした計算や勉強のため程度であれば十分な性能を持つ．対話型としても使えるため，初心者はこれを使うのがよい．

- Java: 開発環境が無償で提供されている上，入門本も多数出ているため，勉強しやすい言語のひとつ．研究や仕事でプログラミングを行う必要がある場合，現在でも主要言語は C++ だが，最近は Java を使うケースも増えてきている．

- cygwin: cygwin とは本来，Windows 環境で UNIX 関連ツールを動かすためのプロジェクトのことであるが，cygwin をインストールすると，ついでに C++ 言語などの開発環境もインストールして利用可能になる．C++ の無償コンパイラとしてはこれを使うのがよい.（もちろん，Linux などを自力で導入できる人は，そうしてもよい）

大学では，教育用計算機センターの端末が使える．駒場では情報教育棟のほか図書館に分散配置端末がある．本郷では図書館や工学部６号館２階２００号室などに分散配置端末がある．分散配置端末に関するＵＲＬは講義ＨＰに挙げておく．この計算機ではいろいろなツールを使えるが，数値計算的に重要なのは以下である．

- MATLAB: プロの研究者も使う数値計算ソフト．ほぼ Scilab と互換だが，一部コマンドセットが異なる．**同時使用人数が 20 名に制限されているとのことなので，レポート提出間際の数値計算には注意が必要**.

- Mathematica: これもプロが使う数値計算（数式処理）ソフトで，単なる数値計算を超えて，数式処理や任意精度の計算など高度な機能を備えていて根強い人気を誇る．ただしプログラミングという観点からは（慣れるまで）使いづらく,『数値解析』の実習用としては薦めない．

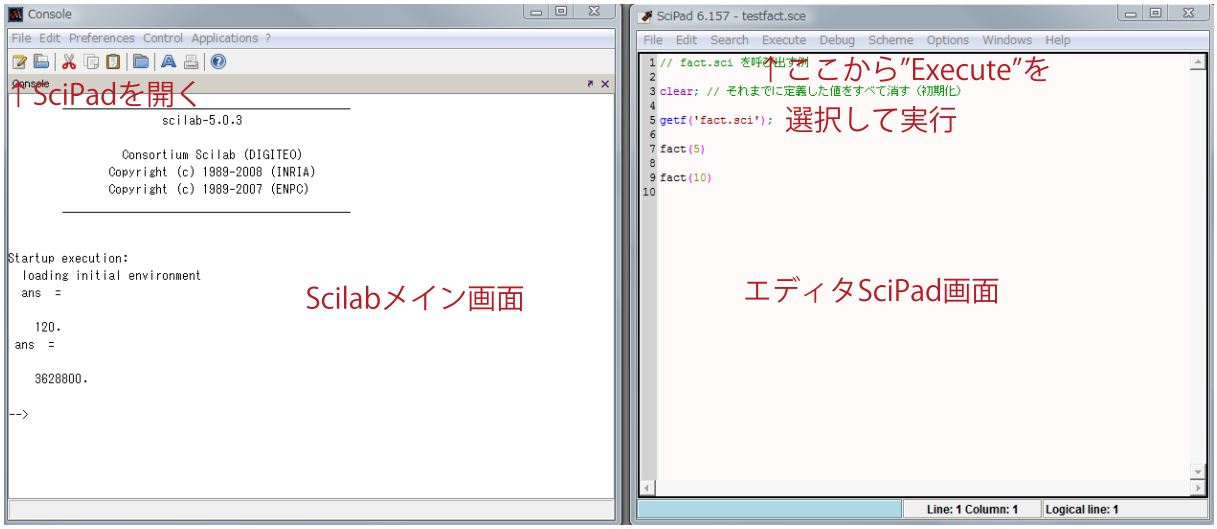
- Java: Java のプログラミング環境が整えられているようである．詳細は，講義ＨＰに挙げたＵＲＬ「はいぱーワークブック」の「26. プログラミング」を参照.

以下では，Scilab，もしくは MATLAB を用いて『数値解析』の実習を進めるための情報を述べる．なお，この２つの両方に対応した解説本として,『MATLAB/Scilab で理解する数値計算』（櫻井鉄也，東京大学出版会）がある．

Scilab をインストールするには，上の Scilab の URL で画面左上の “Download Scilab” というリンクをクリックし，インストールファイルをＰＣにダウンロードして，そのファイルを実行すればよい．途中，ダイアログボックスがたくさん出るが，基本的にはデフォルト設定のままでよい．（何を聞かれているか理解できて，設定を変えたい人は，変えてもよい）

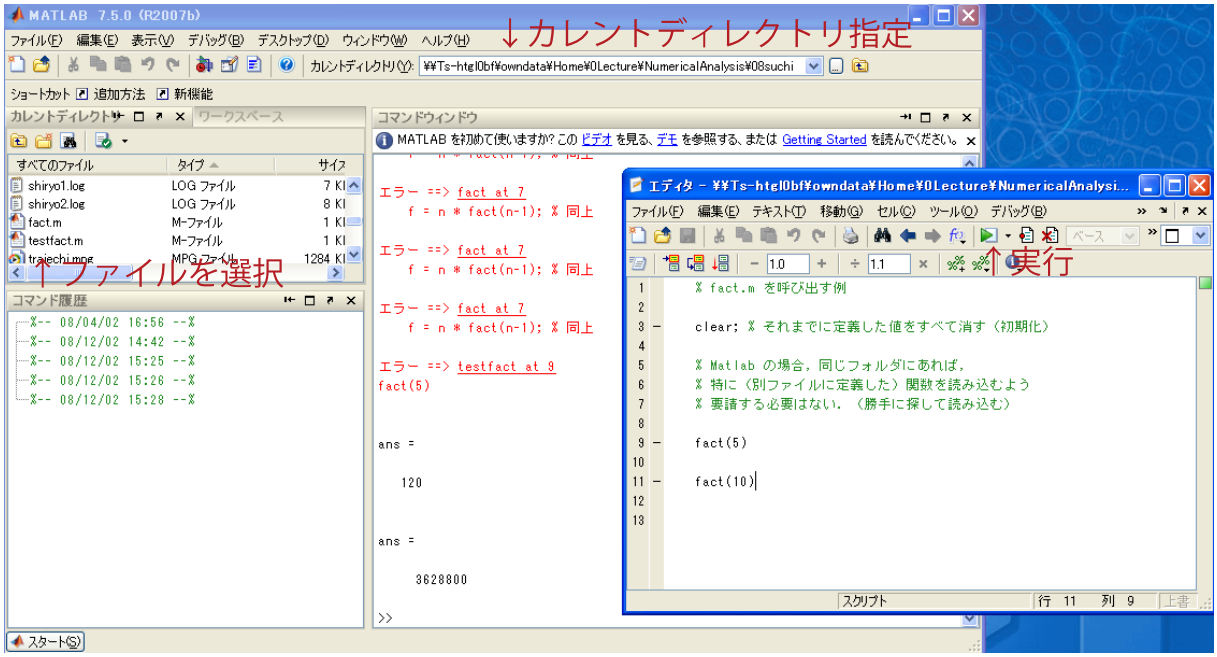
MATLAB/Scilab のいずれを使うにしても，作業ファイルを保存するフォルダをひとつ決めて，そこで作業するとよい．

1.1 Scilab の画面と使い方



Scilab を起動すると，通常では，上図の左側の「Console」ウィンドウだけが開く．ここに直接コマンドを入力すると，電卓的に計算できる（→次節「対話型としての利用）．込み入った計算をするにはプログラムを書くが（→次節「プログラムを書いて実行する」），このときは上図右側の「SciPad」ウィンドウも利用する．SciPad ウィンドウを開くには,「Console」ウィンドウの [File] メニューの下にある「ノートと鉛筆マーク」をクリックする．SciPad にプログラムを記述したら，[Execute]-[Load into Scilab] を選択すると,「Console」ウィンドウで実行される．ファイルは，上で述べた「作業フォルダ」に保存しておこう．すでに Scilab プログラムが存在する場合（*.sce ファイル），それをダブルクリックすると,「Console」画面と「SciPad」画面が両方表示される．

1.2 MATLAB の画面と使い方



！注意！ 上の画面，および次節以降の説明は，Windows 版のものです．教育用計算機センターの端末で利用可能な MacOS 版は操作方法などが若干異なる可能性があります．特に重要な点については，必要に応じて講義ＨＰで情報をアップデートします．

MATLAB は商用ソフトだけあって色々凝った作りになっており，少々分かりにくい点もある．「MATLAB」ウィンドウはデフォルトでは左上の「カレントディレクトリ」ゾーン，左下の「履歴」ゾーン，そして右側の「コマンドウィンドウ」（これがメイン画面）ゾーンの３つに分けられている．上の画像では，それと別に，「エディタ」も開いている．

まずは「MATLAB」ウィンドウの上側にあるドロップボックスで作業フォルダを，自分が決めた場所に変更する．（最初はドロップボックスから，その作業フォルダが選択できないので，右の [...] ボタンをクリックして，作業フォルダを選択する．２回目からはドロップボックスで選べるようになる）．すると，「カレントディレクトリ」ゾーンに，作業フォルダの内容が表示される．

初めて作業するときは，[ファイル] メニューの下の [新規作成] ボタンをクリックすると，「エディタ」が開く．２回目以降は，カレントディレクトリ（作業フォルダ）から*.m ファイルを選択すると，自動的にエディタが開く．

編集中のプログラムファイルを実行するには，「実行」ボタンをクリックする．

2 Scilab/MATLAB による数値計算

Scilab と MATLAB は，「対話型」＝コマンドを逐一入力しながら電卓的に計算していく計算方式と，「プログラム型」＝外部ファイルに予めプログラムを記述しておき，それを呼び出すことで一連の手続きを一気に実行する方式のどちらでも計算が可能である．ちょっとした計算には前者が便利で，込み入った計算や何度も繰り返し（場合によりちょっとずつ内容を変えながら）行う計算には後者が便利である．

2.1 対話型としての利用

2.1.1 基本的な演算

単純に電卓として使うには「計算式」を入力してリターンを押せばよい．MATLAB では `i`，Scilab では `%i` を「虚数単位」として，複素数も扱える．大きな数や小さな数を入力したい場合，たとえば `1.23e-4` のように書ける．

問． `1+2`，`3*4`，`(1+2*i)*(1-2*i)`（もしくは `(1+2*%i)*(1-2*%i)`）などとして結果を確認せよ．

方向キーの上下「↑」「↓」で，過去に入力したコマンドを参照できる（「ヒストリ」と呼ぶ）．コマンドを打ち間違えた場合，似たような作業を繰り返す場合には便利．

基本的な数学演算子として，`+-*/^`が使える．`^`はべき乗を表す（`a^b` は a^b）．数学定数 π は MATLAB では `pi`，Scilab では `%pi` として参照できる．（このように Scilab では，システムで予約されている定数には`%`が付く事が多い）．

電卓より便利なのは，演算結果を**変数**に次々と保存できる点である．(Scilab/MATLAB に限らず，一般にプログラミング言語で「変数名＝右辺」と書いたら，これは**左辺の変数に右辺の計算結果を代入する**という意味なので注意すること．数学の等号とは意味が異なる）．

問． 変数 `a`，`bb`, `c1` などに値を代入し，適当に演算してみよ．たとえば `a=2`，`bb=3`，`c1=a*bb` などとしてみよ．さらに同じ命令セットで，カンマをセミコロン “`;`” に変えたらどうなるか？(注：上で説明した「ヒストリ」機能を使うこと！）

上の問いで例示したように，１行の入力中に複数の演算を書く場合には，カンマかセミコロンで区切る．これらは左から右へ順に実行される．カンマで区切ると，命令ひとつごとに結果が表示されるが，セミコロンの場合表示されない．また最後の演算（１つしか演算がない場合を含む）の末尾にセミコロンを付けると，その結果も表示されなくなる．これはプログラムで大規模な計算をする場合に，途中の計算の表示を抑制するために便利な機能である．

なお，C++ など厳格なプログラミング言語では，変数は事前に，その「型」(実数値か，複素数値か，など）を厳密に宣言しなければならないが，Scilab/MATLAB ではその必要がない．変数に数値を代入すると，自動的にその「型」が判断される．変数名には大文字小文字が使えるが，その両者は区別されるので注意すること（`A` と `a` は別の変数と見なされる）．

2.1.2 行列とベクトル

Scilab/MATLAB では，行列やベクトルを次のように書く．

- 行ベクトル： $(1,2,3)$ は `[1 2 3]`.
- 列ベクトル： $(1,2,3)^T$ は `[1; 2; 3]`.
- 行列： $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ は `[1 2; 3 4]`.

行列，ベクトルの演算では `+-*` が使える．さらに，アポストロフィ ’ で共役転置をとれる（なおスカラー変数は長さ１のベクトルと見なせるため，スカラー変数にアポストロフィを付けると,「複素共役」が出てくる）．たとえば `a`，`b` が長さの等しい列ベクトルの場合， `a'*b` はベクトルの内積になる．

コラム：行列やベクトルの便利な演算

一般のプログラミング言語では，スカラー同士の演算は定義されているが，ベクトルや行列の演算は定義されていない．しかし MATLAB/Scilab はもともと線形計算を指向したソフトのため，豊富な演算が予め用意されており，しかもそれらは，ユーザが後述の `for` 文などを使ってプログラムするより一般にはるかに高速である．たとえば `a` をスカラー変数（あるいはスカラーの値），`b` をベクトル（や行列）とするとき，`a*b`，`b/a` などの演算もでき，予想したとおりの結果になる．ベクトル同士，行列同士の和や差，行列ベクトル積も簡単に記述できる．

さらに，連立一次方程式の解法や固有値問題の解法のルーチンも，もちろん備わっている．たとえば，$Ax = b$ は `A\b` （日本語環境ではバックスラッシュ \ は ¥） とタイプすると答えの x が帰ってくるし（内部では消去法を使っている），`[L, U, P] = lu(A)` とすると，LU 分解が得られる．

問． 適当にベクトルや行列を定義し，基本的な演算を色々試してみよ．行列ベクトル積，内積もやってみること．また適当な大きさの $Ax = b$ を考え，上のコラムで紹介した 2 つの方法でこれを解いてみよ．LU 分解を使う場合，$Ax = b$ を $PAx = Pb$ として解くことになるので注意すること．余力のある人は，プログラムの練習がてら，自分で LU 分解のルーチンを書いてみるとなおよい（プログラムについては次節を参照）．

数値計算では，行列やベクトルの要素を取り出すことが多い．`A(2,3)` は行列 `A` の第 2 行第 3 列要素，`b(2)` はベクトル `b` の第 2 要素を表す（行ベクトルと列ベクトルのどちらの場合でも）．

大きな行列やベクトルを機械的に生成したい場合，`zeros`（零行列），`ones`（要素がすべて 1），`eye`（単位行列），`diag`（対角行列），`rand`（乱数行列）なども便利である．`diag` 以外は，`zeros(2,3)` などと書くと，大きさ $(2, 3)$ （2×3）の行列ができる．`diag` は `diag([1 2 3])` のように書く．

Scilab や MATLAB には，これら以外にも，便利な演算や関数がたくさん定義されている．ヘルプや各種の解説本，解説ページを自分なりに検索して，どんなことができるのか調べてみるとよい．

2.2 プログラムを書いて実行する

ごく簡単な演算は対話型でやると速いが，込み入った計算は，予め別途プログラムファイルに書いておき，それを読み込む方が効率がよい．MATLAB/Scilab それぞれ，前述の方法でエディタを開いて，そこで作業してみよう．書いたプログラムは，Scilab の場合は拡張子 `sce`，MATLAB の場合は拡張子 `m` を付ける約束になっている．（こうしておくと，Windows でそのファイルをダブルクリックすると，自動的に MATLAB/Scilab とエディタが開く．）

問． 次の簡単なプログラムを MATLAB/Scilab のエディタで編集し，保存せよ．そしてそれらをエディタから実行してみよ．

```
a=1;
b=2;
a*b
```

込み入ったプログラムには，「コメント」を付ける習慣をつけるとよい．MATLAB では `%`，Scilab では `//` 以降がコメントと見なされプログラム的には無視される（本資料後半に載っている例を参照のこと）．

以下，プログラムを書く際に特徴的に現れる要素について説明する．

2.2.1 制御構造

条件分岐は if 文で行う．下は Scilab の書式である．MATLAB では then を省く．

```
if 条件式 then          ※注：MATLAB では then は省く！
  命令
elseif 条件式 then
  命令
  ・
elseif 条件式 then
  命令
else
  命令
end
```

「条件式」は上から順番に評価され，成立した条件式直下の「命令」（複数の命令があってもよい）だけが実行される．どの条件式も満たされない場合，else 節があれば，その内容が実行される．elseif 節は何個あってもよい（0でもよい）．else 節は省略でき，その場合，どの条件式も満たされない場合は何も実行されない．

条件式は，まず MATLAB/Scilab に共通のものとして，スカラー変数 a，b に対して，a<b，a>b，a<=b，a>=b，a==b などがある．これらの意味は見て想像したとおりであるが，特に== が数学的な意味での「等号」であるので，改めて「代入」の意味での = との区別に注意すること.「等しくない $a \neq b$」を表すのには，a~=b（MATLAB），a!=b（Scilab）を使う．複数の条件式を結合したい場合，各条件を () で括り，&（AND)，もしくは |（OR）で繋げる．例えば，if ((a > b+1) & (c != 0)) then のように書く．

決まった回数の繰り返しには for 文を使う．

```
for ループ変数 = ループ変数の範囲
  命令
end
```

たとえば for i = 1:10 と書くと，i を 1 から始めて,「命令」を 1 回実行するごとに i に 1 を加算して，合計 10 回実行する．また増分・減分を陽に指定して，for i = 0:2:10（0 から始めて 2 ずつ増やして 10 まで），for i = 10:-1:1（10 から始めて 1 ずつ減らして 1 まで）などとも書ける．

ある状況が満たされている間，ずっと繰り返したい場合は，while 文を使う．

```
while 条件式
  命令
end
```

たとえば while (1.0+r > 1.0) と書くと，r が machine epsilon より大きい間は繰り返し「命令」が実行される．

繰り返し制御文の for，while 文において,「命令」の途中でそのループを終了し，次のループに移るには continue を，そもそも繰り返し自体をやめて，その先（プログラムで言うと「下」）に進むには break を使う．

2.2.2 関数の定義

大きなプログラムになると，全体をいくつかのパーツに分解して記述する方がわかりやすく間違いが少ない．一般にプログラム言語で，一連の命令をまとめてひとつの機能を実現したものを「関数」や「手続き」などと呼ぶ（言語により異なる．また数学的な意味での関数とは少しニュアンスが異なるので注意すること．数学の「関数」とは，ある値を他の値へと写す「写像」であるが，プログラミング言語における「関数」は，値を返すことよりもむしろ「その中で一連の手続きを行う」ことに重きが置かれることが多い）．MATLAB/Scilab では「関数」を定義できる．

関数は，メインプログラムと同じファイルに書くこともできるが，特に MATLAB では関数ごとに１つのファイルを作成することが奨励されているため，ここではその方法を説明する．

関数のファイルの一行目（それより前にコメント行はあってもよい）に，

`function [出力変数のリスト] = 関数名（入力引数のリスト）`

と記述し，それ以降に，関数の具体的な内容を記述する．ファイルは，関数名と同じ名前に拡張子 sci（Scilab の場合），もしくは m（MATLAB の場合）を付けたもので保存する．

たとえば，$n!$ を計算する関数 `fact` を定義して使ってみよう．このとき，Scilab/MATLAB のそれぞれで，次のようなファイルを用意する．

Scilab: fact.sci

```
// 関数定義のサンプル：　階乗
function [f] = fact(n)
  if (n==1) then
    f = 1;  // 関数実行中に結果が表示されないよう ; で抑制
  else
    f = n * fact(n-1); // 同上
  end
```

MATLAB: fact.m

```
% 関数定義のサンプル：　階乗
function [f] = fact(n)
  if (n==1) % MATLAB は "then" 不要
    f = 1;   % 関数実行中に結果が表示されないよう ; で抑制
  else
    f = n * fact(n-1); % 同上
  end
```

（注意：ここでコメントの付け方，`then` の有無などに２つのソフトで違いがあることに注意しよう）．

これらの関数を呼び出すメインプログラムは，例えば次のように書く．

Scilab: testfact.sce

```
// fact.sci を呼び出す例
getf('fact.sci'); // Scilab では必要
fact(5)
fact(10)
```

Scilab の場合，関数の書かれた外部ファイルを `getf` で明示的に読み込んでから，作った関数 `fact` を使う．一方，MATLAB では（同じフォルダにあれば）いきなり呼び出して構わない．

MATLAB: testfact.m

```
% fact.m を呼び出す例
fact(5)
fact(10)
```

問. 自分の環境でこれらのファイルを作成し，実行してみよ.（なお，参考までにこれらのファイルは講義ページにも挙げておく）.

2.3 グラフを描く

数値結果を表や図の形で整理するには，データをファイルとして出力して何らかのツールを使う，もしくは Scilab/MATLAB 自身でグラフを作成する，のいずれかを行う必要がある．数学系の研究者のほとんどは，グラフ作成には gnuplot というツールを使っている．これは無償でありながら（3 D描画を除けば）柔軟で強力なツールで，早い段階で使い方を覚えて損はない．しかし「データをファイルで出力し」「gnuplot でそれを描画する」手順をここですべて説明するのが難しいため，ここでは Scilab/MATLAB 内で最低限のグラフ描画を行う方法を説明する．

2.3.1 Scilab におけるグラフ描画

Scilab には多彩なグラフ描画コマンドがあるが，最も基本的なものは，点 (x, y) の列が与えられたときに，それを結ぶ線を描く `fplot2d` であろう．例で使い方を示す．

Scilab: graphsample.sce

```
// グラフ描画のサンプルファイル
xr = (0:0.1:6*%pi);  // [0,6 π] を 0.1 刻みの格子に
n = length(xr); // xr の要素数を求める
for i=1:n
  y(i) = sin(i*0.1);  // sin(x) を計算
end
plot2d(xr, y) // グラフ本体を描画
xtitle('graph of sin(x)', 'x', 'sin(x)');
              // グラフタイトル，X 軸名，Y 軸名
```

このプログラムを実行すると，グラフウィンドウが別に開いて，そこにグラフが表示される．グラフウィンドウでは，表示されたグラフを印刷（[File]-[Print...]），またはワープロ，TeX などに張り込める画像を出力（[File]-[Export to...]）できる（レポートに貼り付ける場合，Word などを使うなら BMP や JPG を選択するとよい．TeX の場合は EPS がよい）.

注意. 上で `xr=(0:0.1:6*%pi)` という命令があるが，これは「0 から 0.1 刻みで 6π を超えない範囲で数列（＝ベクトル）を生成せよ」の意味．Scilab や MATLAB にはこのような便利な表現がたくさんあるが，わかりにくければ，もちろん `for` 文や `while` 文でベクトルを生成してもよい.（`n=1000; dx=6*%pi/n; for i=1:n xr(i)=dx*i; end`）

2.3.2 MATLAB におけるグラフ描画

上の Scilab の例とほとんど同じことが MATLAB でもできる．微妙にコマンド等が異なるのでよく確認しよう．

MATLAB: graphsample.m

```
% グラフ描画のサンプルファイル
xr = (0:0.1:6*pi);  % [0,6 π] を 0.1 刻みの格子に
n = length(xr); % xr の要素数を求める
for i=1:n
  y(i) = sin(i*0.1);  % sin(x) を計算
end
plot(xr, y) % グラフ本体を描画
title('graph of sin(x)'); % グラフタイトル
xlabel('x'); % X 軸名
ylabel('sin(x)'); % Y 軸名
```

MATALB は商用なだけあって，グラフ描画に関しても強力な機能を備えている．たとえば上ではプログラム中でラベルなどを設定しているが，MATLAB の場合，plot 命令で開いたグラフウィンドウで，メニューから対話的に軸名を設定したり，凡例を追加したりなどが可能である．

グラフの印刷は [ファイル]-[印刷...] から，画像ファイルの出力は [ファイル]-[別名で保存...] でファイル形式を選択しながら行える．

2.4 時間の計測

プログラムしたアルゴリズムの実行時間を測定するには，次のようにする．

まず Scilab の場合は timer() 関数を使うのが標準的である．これは最後に呼び出してからの実行時間（秒）を与える．例を示す．

Scilab: timer.sce

```
// timer 関数で実行時間を測定する例
// 行列ベクトル積を 2 通りのやり方で時間測定する
n=100;  // ←ここをいろいろに変えて試してみよう
A=rand(n,n); // n x n の乱数行列
b=rand(n,1); // 長さ n の乱数ベクトル
// まずは内部ルーチンを使って
timer();  // 時間測定開始
c=A*b; // 内部ルーチンで A*b を計算
t1=timer(); // 前回の timer() からの実行時間を測定
// 次に for ループで
c=zeros(n,1); // ベクトル c を 0 に初期化
timer();
for i=1:n
  for j=1:n
    c(i)=c(i)+A(i,j)*b(j);
  end
end
t2=timer(); // 実行時間測定
// さて時間を比較すると...
t1
t2
```

MATLAB の場合，tic，toc を使う．

MATLAB: tictoc.m

```
% tic, toc 関数で実行時間を測定する例
% 行列ベクトル積を２通りのやり方で時間測定する
n=100;  % ←ここをいろいろに変えて試してみよう
A=rand(n,n); % n x n の乱数行列
b=rand(n,1); % 長さ n の乱数ベクトル
% まずは内部ルーチンを使って
tic;  % 時間測定開始
c=A*b; % 内部ルーチンで A*b を計算
t1=toc; % tic からの実行時間を測定
% 次に for ループで
c=zeros(n,1); % ベクトル c を 0 に初期化
tic;
for i=1:n
  for j=1:n
    c(i)=c(i)+A(i,j)*b(j);
  end
end
t2=toc; % 実行時間測定
% さて時間を比較すると...
t1
t2
```

3　さらに発展的な使い方を学ぶには

以上,『数値解析』のレポートで必要になるであろう範囲の最低限の使い方を駆け足で説明した.

より発展的な使い方は，それぞれのソフトのヘルプを参照するとよい．特に Scilab の [?]-[Scilab Demonstrations] を選択すると，さまざまなデモを見ることができる．また MATLAB にもデモやビデオ，マニュアルの中の “Getting Started” 節などに初心者向けの説明がある（これらはコマンドウィンドウの最初の行にリンクがある）.