信号処理分野のための GNU Octave / MATLAB 入門 [抜粋版]

長谷 芳樹 *

0. はじめに

0.0. 本資料の対象者とコンセプト

• 少しはプログラミング経験があるが GNU Octave / MATLAB は（ほぼ）初めて使う，という人を対象にしています。

• 工学的な話に特化しています。数学的な話は扱っていません。高専の電子工学科５年生を強く意識しています。

• 簡単で実用的なサンプルを集めているつもりですので，眺めるだけではなく実際に手を動かしてやってみて下さい。

※ 本資料は神戸市立工業高等専門学校 総合情報センター広報 第 30 号（2018）pp. 10-17 の抜粋版です。章，節，課題番号等は元の原稿（筆者サイトで公開）から変更していませんので虫食いが存在します。

0.1. インストール

・**GNU Octave**
　`https://www.gnu.org/software/octave/` で GNU Octave 4.2.1 を入手（有用であると感じたらプロジェクトに寄附することを勧める）

0.2. 基本概念

GNU Octave / MATLAB は以下のような特徴を持つ言語（プラットホーム）です。

• 行列演算が得意（Fortran 90 に似ている）

• 関数がとても充実しているので，車輪を再発明する前に，まずは望みの関数が用意されているかどうか探すのがよい（「doc ［コマンド］」でドキュメントを参照できる）

• 基本的に変数の「型」の概念はあまり気にしない（特に明示しなければ全ての数値は倍精度浮動小数点（double）型で扱われる）

• for 文は極力使わず，行列のままでの計算を使う方が良い（for 文は遅く，コードも長くなる）

0.3. 操作系

・命令の最後に「;」を付けると結果を表示しない
・カーソル↑で前のコマンドを呼び出し
・[Ctrl]+[C]で中断

1. 単純な計算

1.1. 電卓として使う

```
>> 1+2*3　【画面に最初から表示されている >> 記号の右に半角で 1+2*3 とだけ入力して [Enter] を押す】
>> sqrt(2)
>> exp(i*pi/6)　　【円周率 pi / 複素数 i】
>> 20*log10(2)　　【比 → dB の計算】
```

◆課題１：dB →比 に戻してみる（べき乗は "^"）。

◆課題２：その他，各自で好みの計算をやってみる。（利用できる演算子は 6.3.参照）

1.2. 変数の定義とクリア

```
>> a=1　　【"=" は代入】
>> a　　【変数の内容の確認】
>> b　　【定義されていないとエラーになる】
>> b=2
>> a+b
>> clear a　【変数 a をクリア（メモリ解放)】
>> a　　【定義されていないのでエラーになる】
>> clear　【全ての変数をクリア】
>> b　　【定義されていないのでエラーになる】
```

2. 行列の扱い

2.1. 行列の基本操作

```
>> c=[10 20 30 40]【 [ ] で行列の生成や連結ができる。四角カッコは行列，と意識。列の区切りはスペース。】
>> c
>> c*2
>> c(1)　【インデックス（添え字）を指定するときは丸括弧。インデックスは 1 からはじまるので注意。従って，c(0)はエラーとなる。】
>> c(2)
>> c(5)　　【定義されていないのでエラーになる】
>> c.'　【転置：1×4 の行列が 4×1 の行列になる】
>> sum(c)
>> mean(c)　【算術平均は average ではなく mean】
>> d=[50 60 70 80]
>> c+d
>> kekka=c+d　【c+d の計算結果を kekka に代入】
>> kekka　　【単に c+d としたのと同じ結果になる】
>> c.*d　　【要素同士の演算は"."を付ける】
>> c*d.'　　【c*d（行列同士の掛け算）が可能かも試す】
>> e=[1 2 3; 4 5 6; 7 8 10]　　【行の区切りは";"】
>> e^2　　【行列×行列（e × e）の演算。c^2 も試してエラーとなることを確認する。】
>> e.^2　　【行列の各要素それぞれの２乗】
>> inv(e)　　【逆行列】
>> e.'　　【転置（ ' だけだと複素共役の転置)】
>> sum(e)　　【各列の和（行列を返す)】
>> sum(sum(e))　【２次元行列の全部の要素の和】
>> g1=[0:9]　　【カッコ [ ] は無くても可】
>> g2=[0:0.1:0.9]　　【増分も指定できる…】
>> g3=[0:9]/10　　【…が，こちらの方がオススメ】
```

◆課題３：1 から 100 までの整数の合計は？

◆課題４：7 ～ 18 までの整数それぞれの自乗（２乗）の合計は？

* 神戸市立工業高等専門学校 電子工学科 准教授

2.2. 行列の基本関数

```
>> size(c)
>> size(c,1)      【行の数】
>> size(c,2)      【列の数】
>> ones(3,4)      【全部 1 の行列】
>> zeros(3,4)     【全部 0 の行列】
>> eye(3,4)       【単位行列】
```

2.3. 行列の一部の切り出し

```
>> c(1:2)         【1 次元行列の一部を切り出す】
>> c_cut=c(1:2)   【切り出した部分を新しい行列に代入】
>> c(2:4)
>> c(2:end)       【最後の要素は end で指定可能】
>> c(end:-1:1)    【1 次元行列を逆順にする】
>> e(1,:)【2次元行列の一部を取り出す（「:」は全て）】
>> e(:,1)
>> e(2:3,1:2)     【行列の一部を取り出す】
>> e(2:end,1:2)
```

2.4. 行列の一部への代入

```
>> c(1)=15      【c の 1 つ目の要素に 15 を代入】
>> c(1:2)=12    【c(1)～ c(2)に 12 を代入】
>> c(1:2)=[13 14]  【c(1)に 13 を，c(2)に 14 を代入】
>> c(1:2)=d(3:4)
```

2.5. 行列同士の結合・ゼロ詰め

```
>> [c d]      【行列の結合（横方向に並べる)】
>> [c; d]     【行列の結合（縦方向に積み上げる)】
>> [c(1:3) d(4)]     【行列の一部分同士の結合】
>> [c zeros(1,5)]    【ゼロの追加（右に)】
>> [c'; zeros(5,1)]  【ゼロの追加（下に)】
>> [c(1:2) zeros(1,5) c(3:end)]  【間にゼロを挿入】
```

3. 2次元グラフの書き方

3.1. グラフの基本操作

```
>> x=[-50:50];【-50 から 50 まで 1 ずつ増やす】
>> y=x.^2;
>> plot(x,y)      【虫眼鏡アイコンで拡大できる】
>> plot(x,y,'o-b')  【 o：プロット有り/ -：直線 /
   --：点線 / r：赤 / g：緑 / b：青 / k：黒 】
>> xlabel('よこじく')  【横軸の名称】
>> ylabel('たてじく')  【縦軸の名称】
```

3.2. 軸の範囲設定・グリッド表示

以下のコマンドはグラフのウィンドウが開いた状態で入力する。

```
>> axis([-60 60 0 3000]) 【[左 右 下 上]の順に指定】
   【xlim([-60 60]); ylim([0 3000]); でも同じ】
>> grid on 【グリッドの表示】
```

3.3. 複数のグラフを重ねる

```
>> y2=40*x-400;
>> plot(x,y,'o-b')  【まずは 1 本目（閉じないで)】
>> hold on  【既に描いたグラフを保持するように設定
   （hold off で解除)。】
>> plot(x,y2,'-r')     【2本目】
>> legend('1本目','2本目');  【凡例の表示】
```

3.4. 複数のグラフを並べる

```
>> subplot(2,1,1)  【2 行 1 列に並べて 1 個目の枠に】
>> plot(x,y,'o-b')  【まずは 1 本目を描く】
>> title('1本目')
>> subplot(2,1,2)  【2 行 1 列に並べて 2 個目の枠に】
>> plot(x,y2,'-r')     【2本目を描く】
>> title('2本目')
```

例：2 行 3 列の場合の通し番号（ subplot(2,3,通し番号)):

1	2	3
4	5	6

◆課題５：$-\pi \leqq x < \pi$ の範囲で $y = x$ と $y = \sin(x)$ の２つのグラフを重ねて表示してみる。（２つの関数の特徴が明確になるように x の増分を充分細かく取る。また，凡例も付ける。）

◆課題６：$x > 0$ の範囲で $y = \mathrm{sqrt}(x)$，$y = x$，$y = x^2$，$y = \log(x)$ の４つのグラフを並べて表示してみる。（４つのグラフの特徴の違いが明確になるように x の範囲と増分を調整する。また，各グラフにタイトルも付ける。）

4.「時刻 vs. 振幅」の信号の扱い

4.1. 正弦波の作成（これ以降は値はすべて SI 単位で扱う）

```
>> fs=8000;  【サンプリングレート(標本化周波数) [Hz]】
>> T=0.2;    【生成する信号の時間長 [s]】
>> t=[0:T*fs-1]/fs;【時間軸の生成 [s] (11.4.節参照)】
>> f=440;    【信号の周波数 [Hz]（440 Hz は♪ラ)】
>> sig=sin(2*pi*f*t);  【正弦波関数 sin(2πft) をそ
   のまま書いただけ（ sin の中身はラジアン [rad])】
```

4.2. 正弦波波形の表示

```
>> plot(t,sig,'o-')
>> xlabel('Time [s]')  【横軸の名前】
>> ylabel('Amplitude [arb. unit]')  【縦軸の名前】
>> title('440 Hz の正弦波')
```

◆課題７：♪ミの音などを作成し，sig（♪ラ）と比べてみる。(sig は残しておく)

◆課題８：横軸をミリ秒にして課題７のグラフを描く（横軸の範囲も見やすく調整する：3.2.参照)。

◆課題９：同じサンプリングレート（fs = 8000 Hz）で 3900 Hz の正弦波をサンプリングしたときに波形がどうなるか，グラフを描いて確認する。(sig は残しておく)

◆課題１０：sig の最大値，最小値，平均値，実効値をそれぞれ求めてみる。(いずれも一行で書ける)（ヒント：実効値(RMS) = 信号を **S**quare(自乗)して **M**ean(平均)して **R**oot(ルート)するだけ）

4.3. 波形の再生（MATLAB Mobile は非対応）

```
>> sound(sig,fs)  【ヘッドホンの音量に注意】
>> sound(sig*2,fs)  【聞き比べて違いの原因を考える】
>> soundsc(sig*2,fs)  【信号が -1~1 の範囲に入るよ
   うに自動でスケーリング】
```

4.4. ノイズ(乱数)(一様分布版)

```
>> r=rand(1,fs);  【1×8000 点の乱数行列】
>> plot(r)
>> mean(r)
>> std(r)  【標準偏差】
>> median(r)  【中央値】
>> histogram(r,20)
>> sound(r-0.5,fs)  【直流分をカットして再生】
```

4.5. ノイズ(乱数)(正規分布版)

```
>> r2=randn(1,fs)*0.1+0.5;【正規分布の乱数(randn 関数自体は平均 0.0，標準偏差 1.0 の乱数を返す)】
>> mean(r2)
>> std(r2)
>> histogram(r2,20)
>> sound(r2-0.5,fs)
```

◆課題１１：一様分布乱数から疑似正規分布風の乱数を作ってみる。

5. wav ファイルの読み書き

5.1. wav ファイルの書き出し

```
>> cd 'c:¥hoge'【保存したいフォルダーを作成してから移動(この例は Windows の場合。macOS や Linux では /Users/namae/hoge/ や /home/namae/hoge/ など)】
>> audiowrite('440Hz.wav',sig,fs)
>> audiowrite('440Hz_clip.wav',sig*2,fs)
>> audiowrite('white.wav',r2-0.5,fs);【0 を中心に】
```

◆課題１２：各ファイルに保存された波形を Audacity / SoundEngine 等の波形編集ソフトで確認する。

5.2. wav ファイルの読み込み

```
>> [sig2,fs2]=audioread('440Hz_clip.wav');【MATLAB では.mp3，.ogg，.flac 等も読める。】
>> size(sig2)
>> plot([1:size(sig2,1)]/fs2,sig2)
>> sound(sig2,fs2)
```

◆課題１３：任意の wav ファイルを読み込んで聞いてみる(ファイルがない場合は筆者のウェブサイトからダウンロードする(https://ultrasonics.jp/nagatani/))。また，サンプリングレートを変えて再生してみる。例えば，半音上げて再生するにはどうすればよいか？

6. 関数いろいろ

6.1. 最大値･最小値

```
>> m=[11:20]
>> max(m)
>> min(m)
>> min(17,m)  【いずれか小さい方を返す】
>> max(13,min(17,m))  【13 から 17 の間に収める】
>> plot(m, max(13,min(17,m)), 'o-')
```

例：波形のクリップ(頭打ち)

```
>> plot(t, min(1,sig*2), 'o-')
>> plot(t, max(-0.5,min(1,sig*2)), 'o-')
```

◆課題１４：4.3.の「 sound(sig*2,fs) 」では何が起きていたか？ グラフを書いて確認する(できれば before と after を重ねて描いて比べる)。

6.3. 演算子

算術演算子：
加算 + , 減算 - , 乗算 .* , 除算 ./ (乗算と除算にはドットが必要。ドット無しだと行列演算), べき乗 .^ , 転置 .' (ドット無しだと複素共役転置になるので注意), 剰余 rem(a,b)

関係演算子：
等しい == , 等しくない ~= , より大きい > , 以上 >= , より小さい < , 以下 <=

論理演算子：
論理積(AND) & , 論理和(OR) | , 否定 ~

6.4. よく使う(かもしれない)関数いろいろ

```
zeros / ones / eye
sin / cos / tan / asin / acos / atan(全てラジアン)
log(底を e とする対数) / log10 (底が 10) / exp
abs(絶対値) / round(四捨五入) / ceil(切り上げ) / floor(切り捨て) / real(実部) / imag(虚部) / sign(符号だけを取り出す)
max / min / mean / std / median
cd / pwd
```

6.8. 行列を２次元画像として表示

```
>> image([1:64])
>> image(eye(100,150)*16+randn(100,150)*8+32)
```

7. 信号処理

7.1. FFT

```
>> u=sin([0:0.5:10]);  【なにか適当な信号を用意】
>> fft(u)  【FFT 自体はこれだけで OK】
>> plot(abs(fft(u)),'o-')  【FFT 結果の振幅スペクトル表示(FFT の結果をそのまま表示するだけでは，横軸は Hz 等ではなく「点数」となる → 課題１５参照)】
```

例：ひずんだ正弦波の振幅スペクトルとの比較

```
>> subplot(2,1,1)
>> plot(t,sig,'o-b')  【4.1.で作った sig】
>> sig_clip=min(1,max(-0.8,min(0.8,sig)));
>> hold on
>> plot(t,sig_clip,'o-r')  【クリップ波形】
>> legend('正弦波','クリップされた正弦波');
>> title('波形')
>> xlabel('時刻 [s]'); ylabel('振幅[arb. unit]');
>> subplot(2,1,2)
>> plot(abs(fft(sig)),'o-b')  【元の波形】
>> hold on
>> plot(abs(fft(sig_clip)),'o-r')  【クリップ波形】
>> legend('正弦波','クリップされた正弦波');
>> title('振幅スペクトル')
>> xlabel('点数 [点]'); ylabel('振幅スペクトル');
```

◆課題１５：4.1.で作成した sig（あるいは上記の例で作成した sig_clip）の FFT 演算をおこない，振幅スペクトルを正しい横軸（点数ではなく Hz）で表示する。【ヒント：FFT の周波数解像度は 1/T［Hz］（T は信号の全長［s］），FFT 結果の点数は FFT 前の時間波形の点数 N と同じ】

8. 自作スクリプト・自作関数

8.1. スクリプトの作成と実行

処理を上から順次実行するだけの場合（引数や返り値やローカル変数の使用が必要ない場合）は，単にテキストファイルにコマンドを列挙するだけで良い。

・`edit` を実行すればエディタが開く。
・拡張子`.m` で保存する。ファイル名は半角英数のみ。また，ファイル名の１文字目に数字は使えない。
・実行方法：任意のフォルダーに保存（半角英数のみ）して［F5］を押す。あるいは，コマンドで実行する場合やスクリプトからスクリプトを呼び出す場合は，そのファイルの存在するフォルダーに `cd` コマンドで移動するか `Path` を通してから，単にファイル名（`.m` を除いた前半部分のみ）を入力する

8.2. 関数の作成と実行

返り値が必要な場合やローカル変数を用いる場合（関数化）は以下のようにおこなう。

seigenha.m

```
function [t_axis,waveform] = seigenha(f,T,fs)
% コメント行は % です
      t_axis = [0:T*fs-1]/fs;
      waveform = sin(2*pi*f*t_axis);
```

参考：ファイル名.m と関数名が一致する必要がある。作ったファイルの存在するフォルダーに移動するか Path を通してから実行する（ファイル名の.m を除いた前半部分＝関数名 を入力する）。

実行方法：

```
>> [t,sig] = seigenha(440, 1, 8000);
>> sound(sig, 8000);
```

◆課題２０：周波数 fc［Hz］の搬送波（正弦波）を周波数 fs［Hz］の信号で変調度 m で振幅変調する関数を作成し，波形と振幅スペクトルを確認する。

9. ファイルの読み書き

9.1. CSV 形式（テキストファイル）の読み書き

カンマ区切りファイル（列をカンマで区切ったファイル）を作り，保存したフォルダーに移動してから下記を実行する（もしくはファイル名を `'c:¥hoge¥data.csv'` のようにフルパス指定してもよい）：

```
>> X=csvread('data.csv');
>> plot(X(:,1),X(:,2),'o-')
>> csvwrite('data2.csv',X)
```

参考：任意の区切り文字（カンマ以外）で区切られたファイルを読み書きする場合は `dlmread/dlmwrite` が便利。

```
>> X=dlmread('data.txt',' ');  【スペース区切りの場合】
```

10. 総合課題

◆課題２１：信号 u（7.1.節）にハン窓（ハニング窓）などを掛けて FFT し，矩形窓（窓なし）と結果を比較する。【ヒント：Octave/MATLAB では hann 関数も利用できますが，ハン窓の数式 0.5*(1-cos(2π n/N)) をそのまま使ってみるのもよい練習になるでしょう（いずれにせよ１行で書ける）】

◆課題２２：信号 u（7.1.節）をそのまま FFT した結果と，信号の後ろにゼロ詰めしてから FFT した結果とを比較する。【ヒント：波形の後ろに任意の点数（通常は元信号の整数倍を選ぶことが多い）のゼロ行列を追加すると，信号の全長 T［s］が変わるので，それに応じて周波数解像度 1/T［Hz］も変化します（課題１５も参照）。】

◆課題２３：スイープ音（連続的に正弦波の周波数が変化する）を作成して聞いてみる。また，そのスペクトルも確認する。【ヒント：周波数が一定の正弦波ならば位相［rad］は時間に正比例しますが，周波数が時間的に変化する場合には位相の増え方がどうなるかを考えてください。いずれにせよ単調増加であることには違いありませんが，正比例（一次関数）ではないはず。ついでに指摘しておくならば，sin( ) 関数の中身は位相［rad］ですね。】

11. もう少し発展的な技術

11.2. ウィンドウ（グラフ）を画像として保存

```
>> saveas(figure(1),'hoge.png','png');  【ウインドウを開いたときの「figure(1)」の数字と揃える】
```

本資料について

本資料は神戸市立工業高等専門学校 総合情報センター広報 第 30 号（2018） pp. 10-17 の抜粋版です。章，節，課題番号は元の原稿から変更していません。本資料のフルバージョンの電子ファイル，電子書籍版，および課題の解答例を筆者のウェブサイトで公開しています。個人や研究室などでぜひご活用ください。

筆者のウェブサイト：
`https://ultrasonics.jp/nagatani/`