授業科目名	開講学科
応用情報処理演習 B	建築・社会環境工学科(土木 3 コース)

開講セメスター

5 セメ

単位

1 単位

担当教員名

非常勤講師　蒔苗　耕司　（まかなえ　こうじ）
連絡先：　宮城大学事業構想学部デザイン情報学科
電子メール：makanae@myu.ac.jp　　電話番号：022-377-8368

授業科目の目的・概要及び達成目標等

1．目的：社会資本整備やその維持管理，運用において必要とされる実践的な情報システムの基礎知識とその構築技術について学ぶ．
2．概要：本講義では，数値情報処理，可視化技術，シミュレーション，空間情報システム等を題材とし，個人及びグループ演習により情報システムの構築の実際を学ぶ．
3．達成目標等：具体的な達成目標はつぎの 2 点である：
・社会資本整備に関連する情報システムに関する基礎知識の習得する．
・課題解決のための情報システムの設計・構築技術・マネジメント手法を習得する．

他の授業科目との関連及び履修上の注意

履修要望科目：「応用情報処理演習A」の履修は必須である．また同時期に開講される「システムズ・アナリシス」を併せて受講することが望まれる．

授業計画

1．4/9　　ガイダンス－土木情報処理に求められるもの
2．4/16　数値計算システムの基礎(1)－Octave の使用方法と行列計算
3．4/23　数値計算システムの基礎(2)－関数描画とグラフィックス
4．5/7　　画像処理の基礎 (1)－濃度変換関数
5．5/14　画像処理の基礎 (2)－空間フィルタリングと画像合成
6．5/28　画像処理演習－ファイル入出力と地形情報の処理
7．6/4　　数値計算システムによるプログラミング－制御構造と擬似乱数
8．6/11　モンテカルロシミュレーション
9．6/18　遺伝的アルゴリズムと人工知能
10．6/25　空間情報システムと経路探索
11．7/2　3次元コンピュータグラフィックスの基礎
12．7/9　バーチャルリアリティ
13．7/23　情報システムの設計と構築(1)
14．7/30　情報システムの設計と構築(2)
15．8/1　プレゼンテーション　　※6/25 は休講の場合あり

成績評価の方法及び基準

毎回の課題及びプレゼンテーションにより評価する．毎回提示される課題で 60%，最終課題 40%（プレゼンテーション＋個人課題）である．

教科書・参考書（教書名をクリックすると図書館の所蔵・貸出状況を確認できます）

教科書・参考書は特に必要ないが，各自，利用するソフトウェアに則した入門書があれば参考となる．

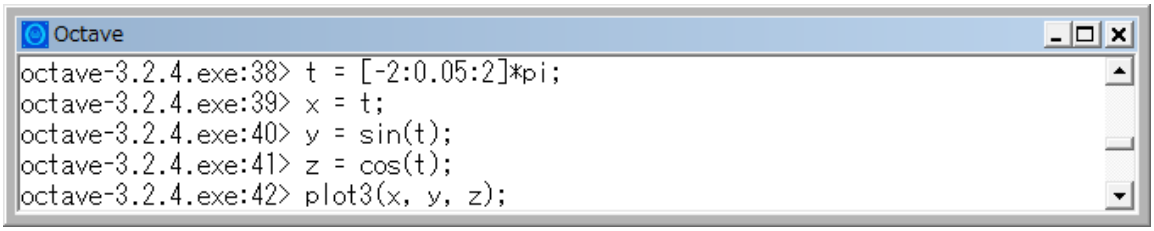
関連ホームページ

http://www.myu.ac.jp/~makanae/

応用情報処理演習 B で用いるシステム

1. 数値解析システム GNU Octave

・数値計算を主たる目的として開発されたオープンソースの高レベル言語である．
・コマンドラインインターフェースを介して，行列計算を基本とした複雑な科学技術計算が可能である．
・MATLAB に類似（一部互換）した言語であり，バッチ指向言語として利用できる．
・公式ホームページは以下を参照．
　http://www.gnu.org/software/octave/
・インストールプログラムは下記より入手可能である．
　http://octave.sourceforge.net/



2. 数式処理ソフトウェア Maxima

・GNU GPL に基づくオープンソースの数式処理システム（コンピュータ代数システム）である．
・1960 年代に MIT で開発され，米国エネルギー省（DOE）で配布されていた MACSYMA を，テキサス大学の Schelter 氏が Common Lisp 環境で移植したものである．
・コマンド処理，バッチ処理によるプログラミングが可能である．
・公式ホームページは以下を参照．
　http://maxima.sourceforge.net/
・インストールプログラムは下記より入手可能である．
　http://sourceforge.net/projects/maxima/files/

```
(%i18) x^4+2*x^3*y + 2*x*y^3 -y^4;

                  4        3      3      4
(%o18)          - y  + 2 x y  + 2 x  y + x

(%i19) factor(%o18);

                  2    2    2           2
(%o19)         - (y  + x ) (y  - 2 x y - x )
```

3. プログラミング環境 Microsoft Visual Basic

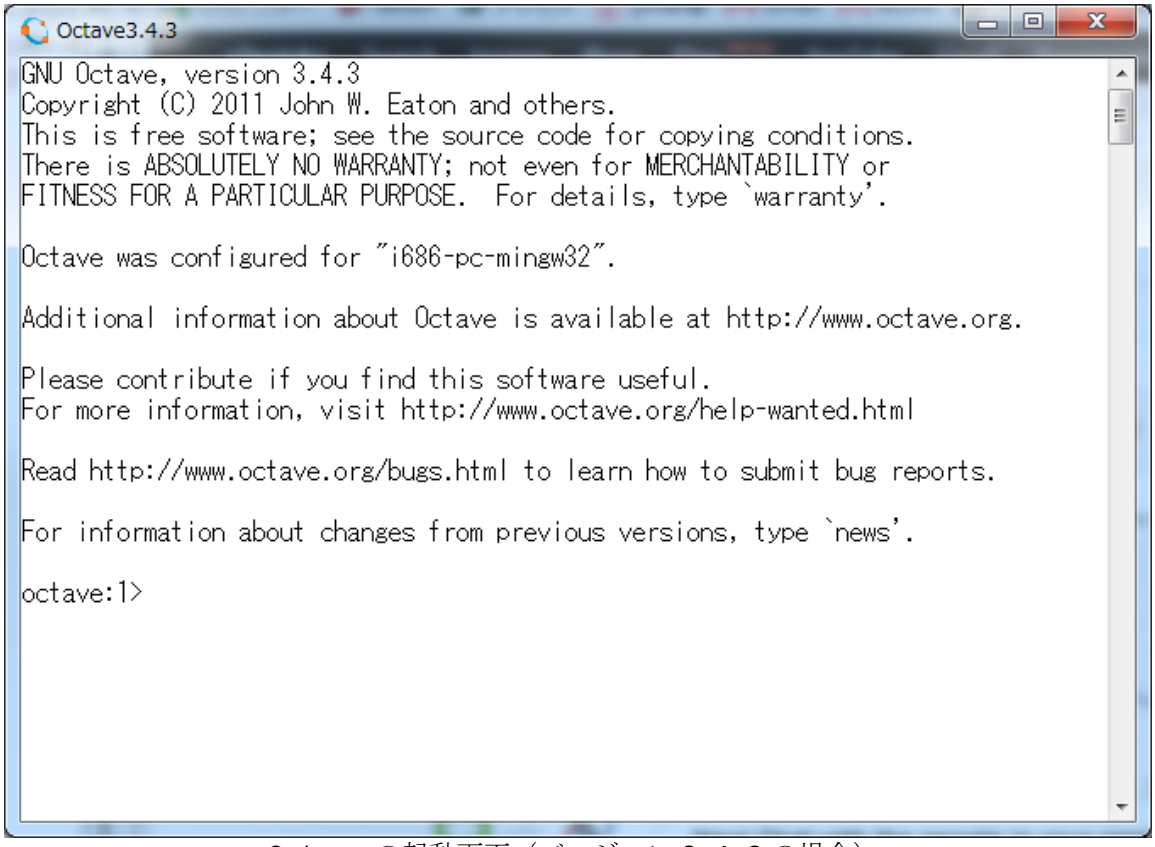
・Microsoft が開発した Basic をベースとしたオブジェクト指向の開発環境である．
・高級言語によるグラフィックスインターフェースを用いたプログラミングが可能である．
・公式ホームページ，インストールプログラムは下記より入手可能である．
　http://www.microsoft.com/japan/msdn/vstudio/express/

※GNU（グヌー）とは：UNIX 互換のソフトウェア環境を全てフリーウェアで実装することを目標とするプロジェクトである．（GNU's Not UNIX の略称）

数値計算システム Octave の基礎

1. Octave の起動

・スタート→すべてのプログラム→Octave3.4.3_gcc4.5.2→Octave を起動する.



Octave の起動画面（バージョン 3.4.3 の場合）

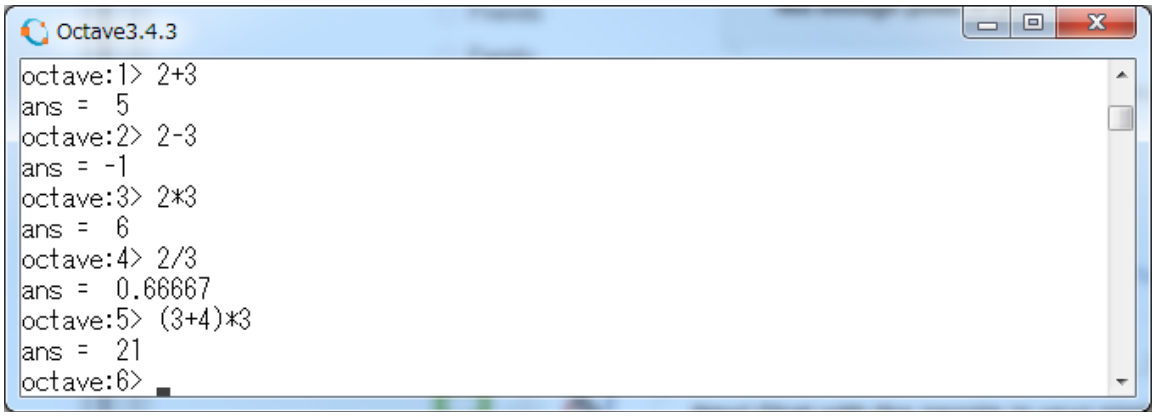
・終了のコマンドは　**exit** あるいは　**quit** である.

2. かんたんな計算

・Octave の基本的な演算子は以下の通りである.

＋　：加算
－　：減算
＊　：乗算
／　：除算
^　：べき乗

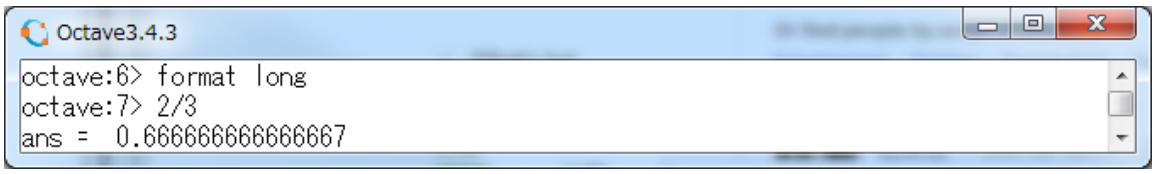
※括弧(　)を用いた演算も可能である.



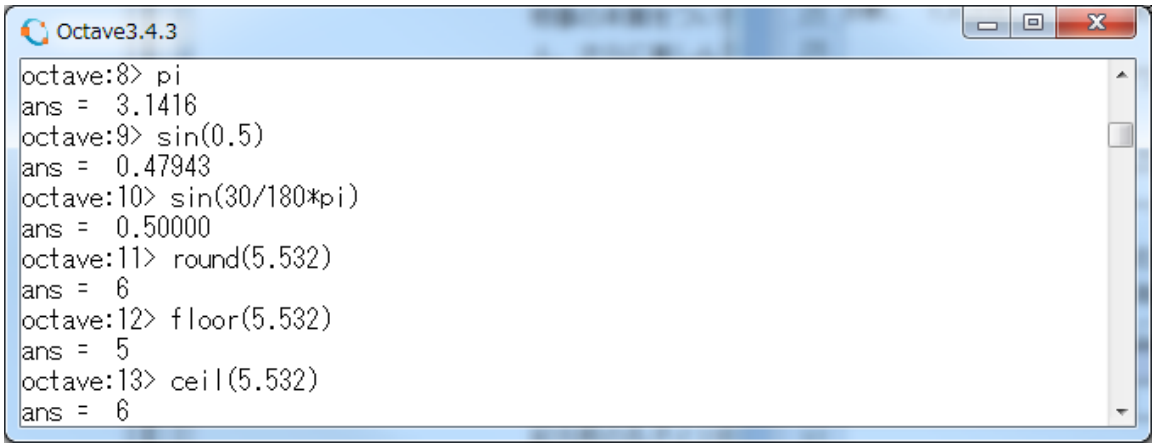
実行画面の例(1)

・Octave の計算精度はデフォルトで単精度（short）である．⇒小数点以下５桁を表示する．
・倍精度（小数点以下 15 桁）にする場合には，以下により設定を変更する．

```
format long     （倍精度にする）
format short    （単精度にする）
```



3. 算術関数



(1)ユーティリティ関数
ceil (x)：x よりも小さくない最小の整数を返す．（小数点切り上げ）
floor (x)：x よりも大きくない最大の整数を返す。（小数点切り捨て）
round (x)：x に最も近い整数を返す。（四捨五入）
exp (x)：x の指数関数を計算する．
log (x) ： x の各要素について，自然対数を計算する．
log10 (x) ： x の常用対数（10 を底とする対数）を計算する．
log2 (x) ： x について，2 を底とする対数を計算する．
mod (x, y) ： x÷y の剰余（割り算の余り）を計算する．
pow2 (x) ： x の平方（2 乗）を計算する．
sqrt (x) ： x の正の平方根（ルート）を計算する．
abs (x) ： x の絶対値を計算する．

(2)三角関数
※引数はラジアンで指定．角度（deg.）の場合は x/180*pi とする必要がある．
sin (x) ： x の正弦（サイン）を計算する．
cos (x) ： x の余弦（コサイン）を計算する．
tan (z) ： x の各要素について，正接（タンジェント）を計算する．
sec (x) ： x の正割（セカント）を計算する．
csc (x) ： x の余割（コセカント）を計算する．
cot (x) ： x の余接（コタンジェント）を計算する．
asin (x) ： x の逆正弦（アークサイン）を計算する．
acos (x) ： x の逆余弦（アークコサイン）を計算する．
atan (x) ： x の逆正接（アークタンジェント）を計算する．
sinh (x) ： x の双曲線正弦（ハイパボリックサイン）を計算する。
cosh (x) ： x の双曲線余弦（ハイパボリックコサイン）を計算する．
tanh (x) ： x の双曲線正接（ハイパボリックタンジェント）を計算する．

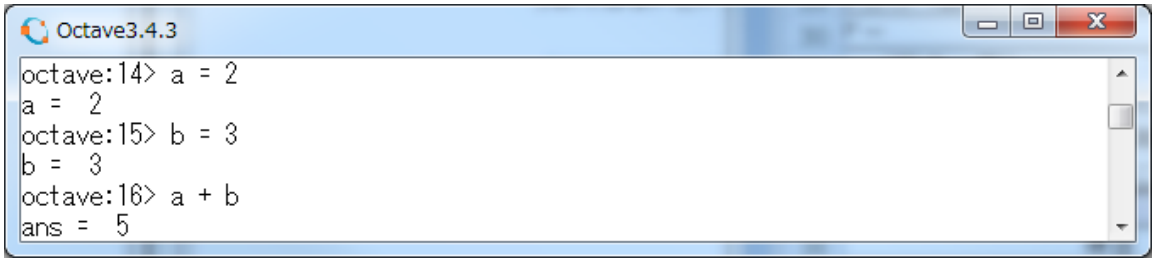
・その他，関数については，Octave マニュアルを参照のこと．
http://www.obihiro.ac.jp/~suzukim/masuda/octave/html/octave_93.html#SEC152

4. 変数

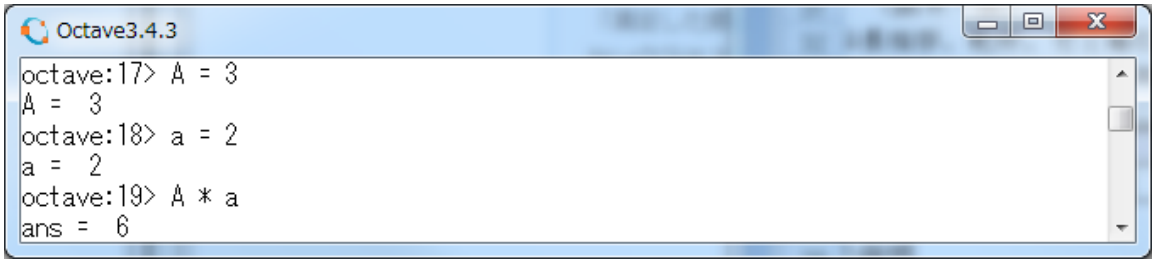
・変数名には文字あるいは文字列を用い，数値あるいは行列を記憶させることができる．

・変数は実数型であり，定義せずに用いることができる．

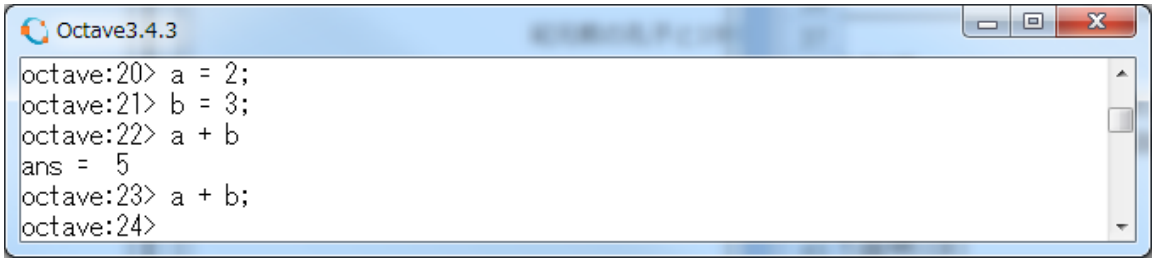
・変数の初期化には　=　を用いる．例：　a = 3



・変数名の大文字と小文字は区別される



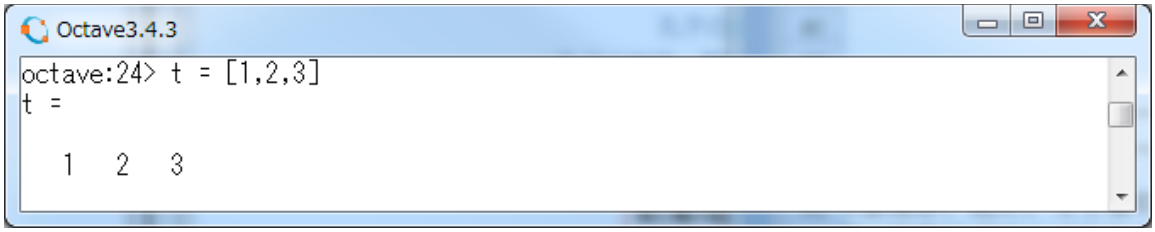
（補足）行末のセミコロン(;)の意味：結果をコンソールに表示しない

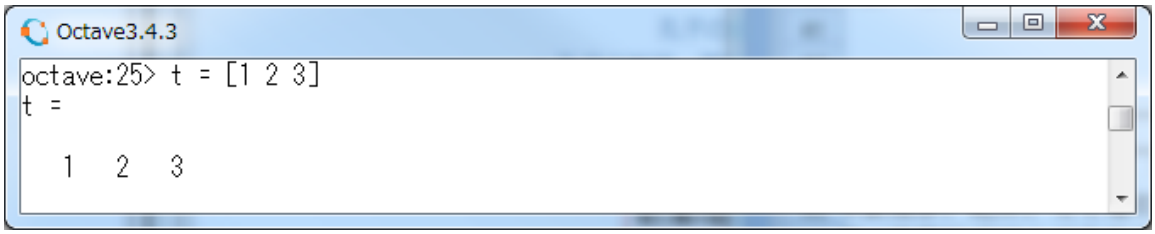


5. 行列

(1) 行ベクトル$X=(x_1,x_2,x_3,\cdots)$の定義

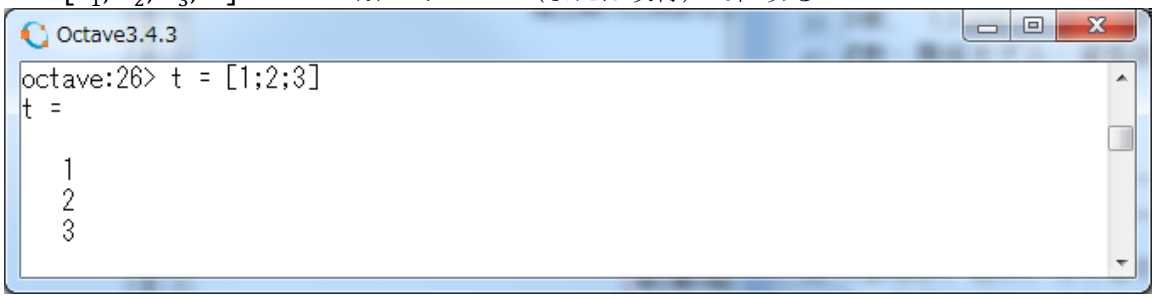
X = [x_1, x_2, x_3 …] または X= [x_1　x_2　x_3 …]　　※スペースあるいはカンマ(,)で区切る.





(2) 列ベクトル$X=\begin{pmatrix}x_1\\x_2\\x_3\\\vdots\end{pmatrix}$の定義

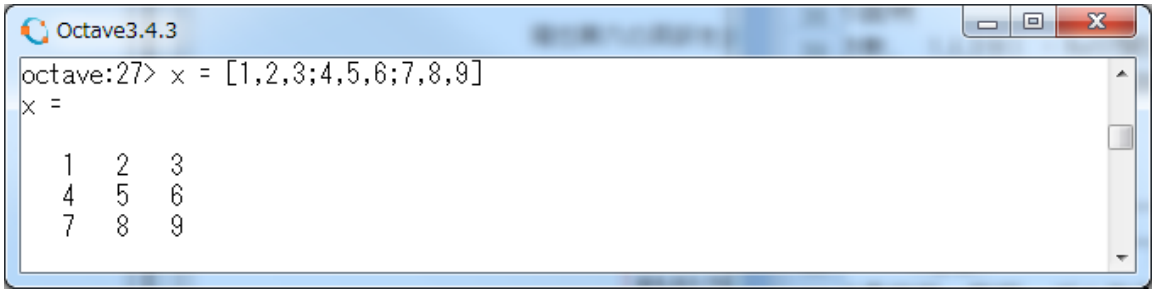
X=[x_1; x_2; x_3,…]　　※セミコロン（または改行）で区切る.



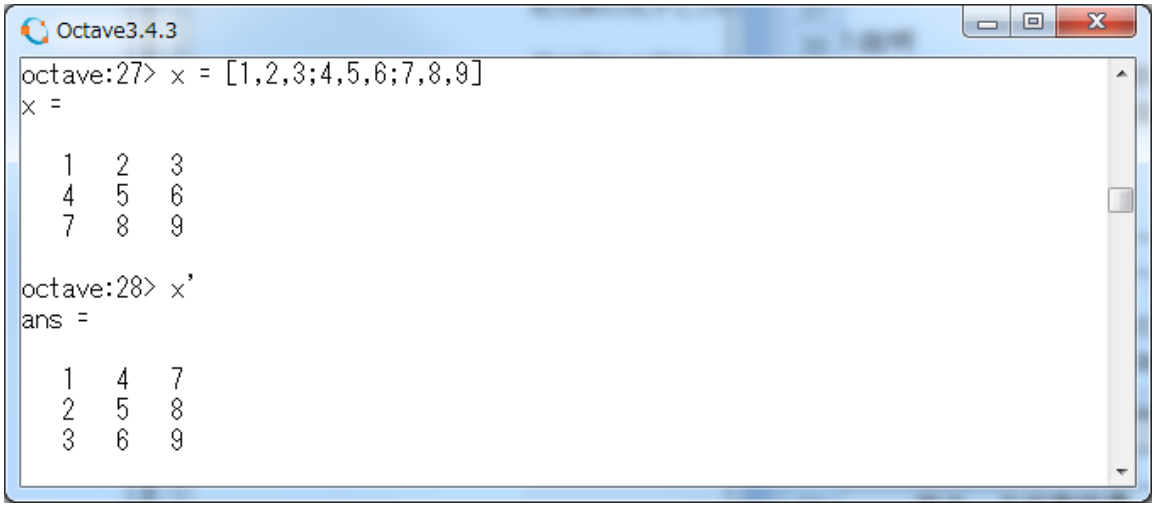
(3)行列 $X=\begin{pmatrix}x_{11} & x_{12} & \cdots & x_{1n}\\x_{21} & x_{22} & \cdots & x_{2n}\\\vdots & \vdots & \ddots & \vdots\\x_{m1} & x_{m2} & \cdots & x_{mn}\end{pmatrix}$ の定義

X=[$x_{11},x_{12},\cdots,x_{1n}$; $x_{21},x_{22},\cdots,x_{2n}$; … ; $x_{m1},x_{m2},\cdots,x_{mn}$]

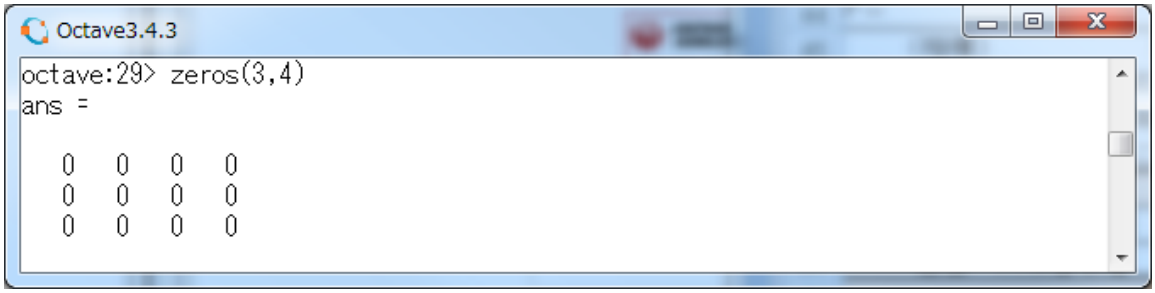
※行はカンマ(,)またはスペースで，列はセミコロン(;)で区切る



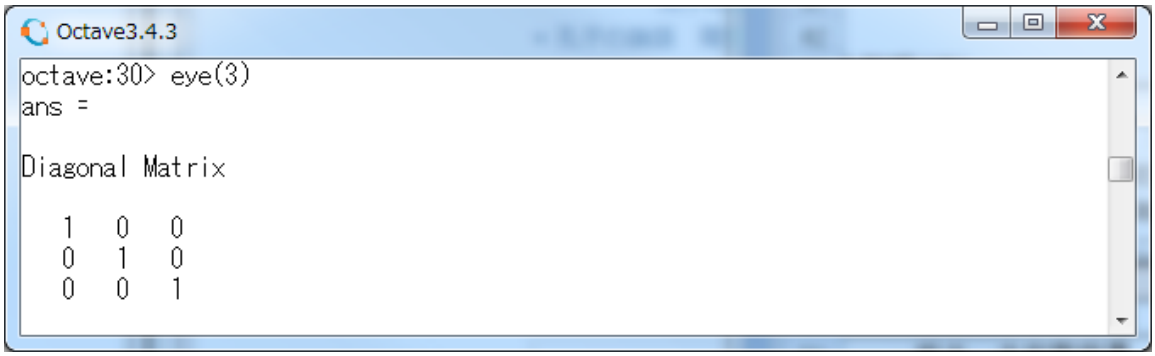
(4) X の転置行列 X′
※アポストロフィ(’) を付する.



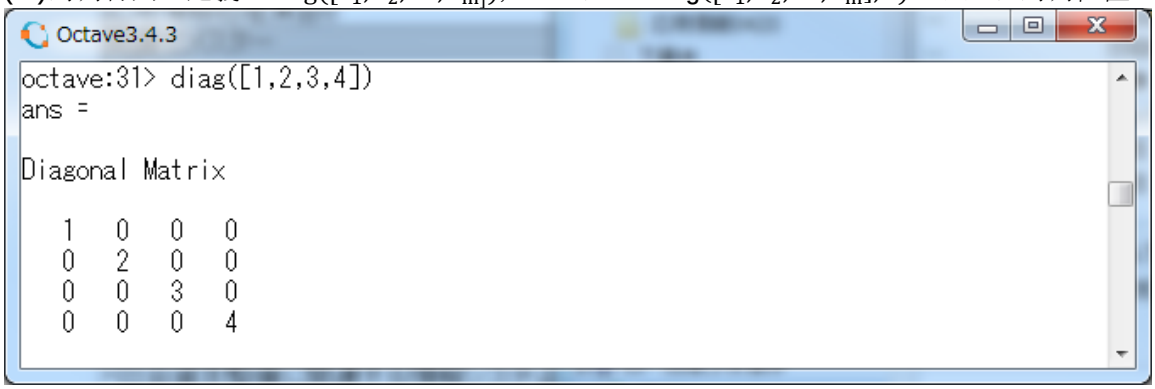
(5)ゼロ行列（m×n）の定義　zeros(m, n)



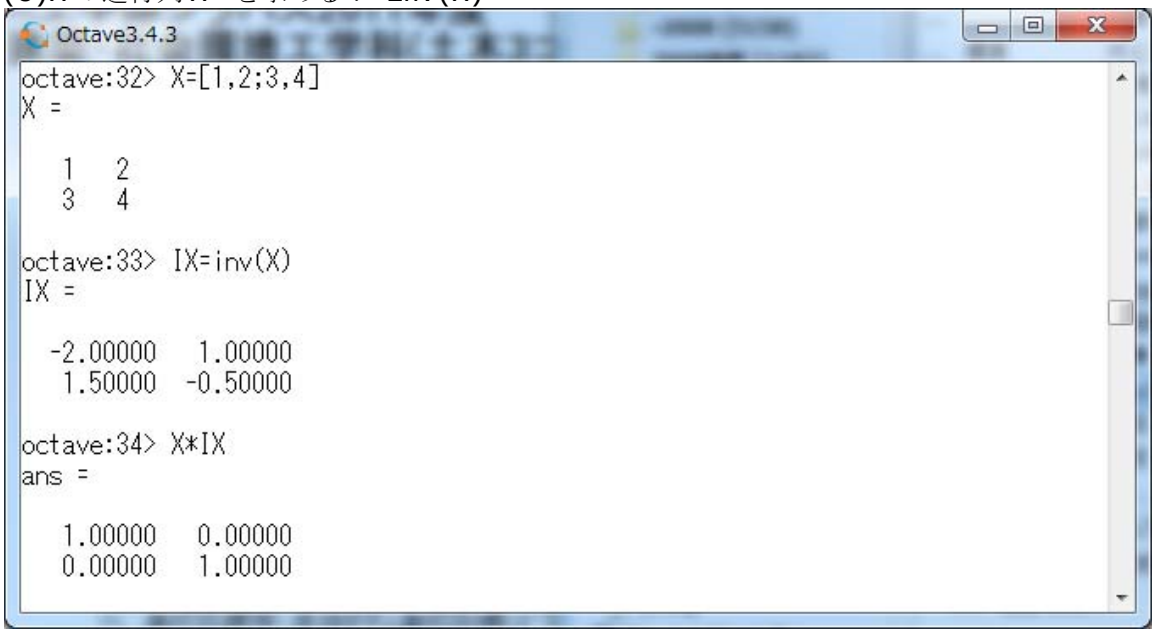
(6)単位行列の定義　eye(n)　　※n は次数



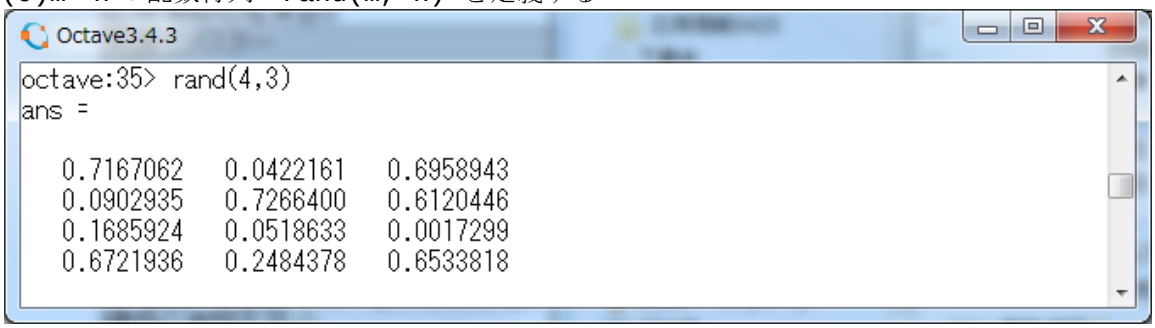
(7)対角行列の定義　diag([$x_1, x_2, \cdots, x_m$])，または　diag([$x_1, x_2, \cdots, x_m$], k)　※n は対角位置



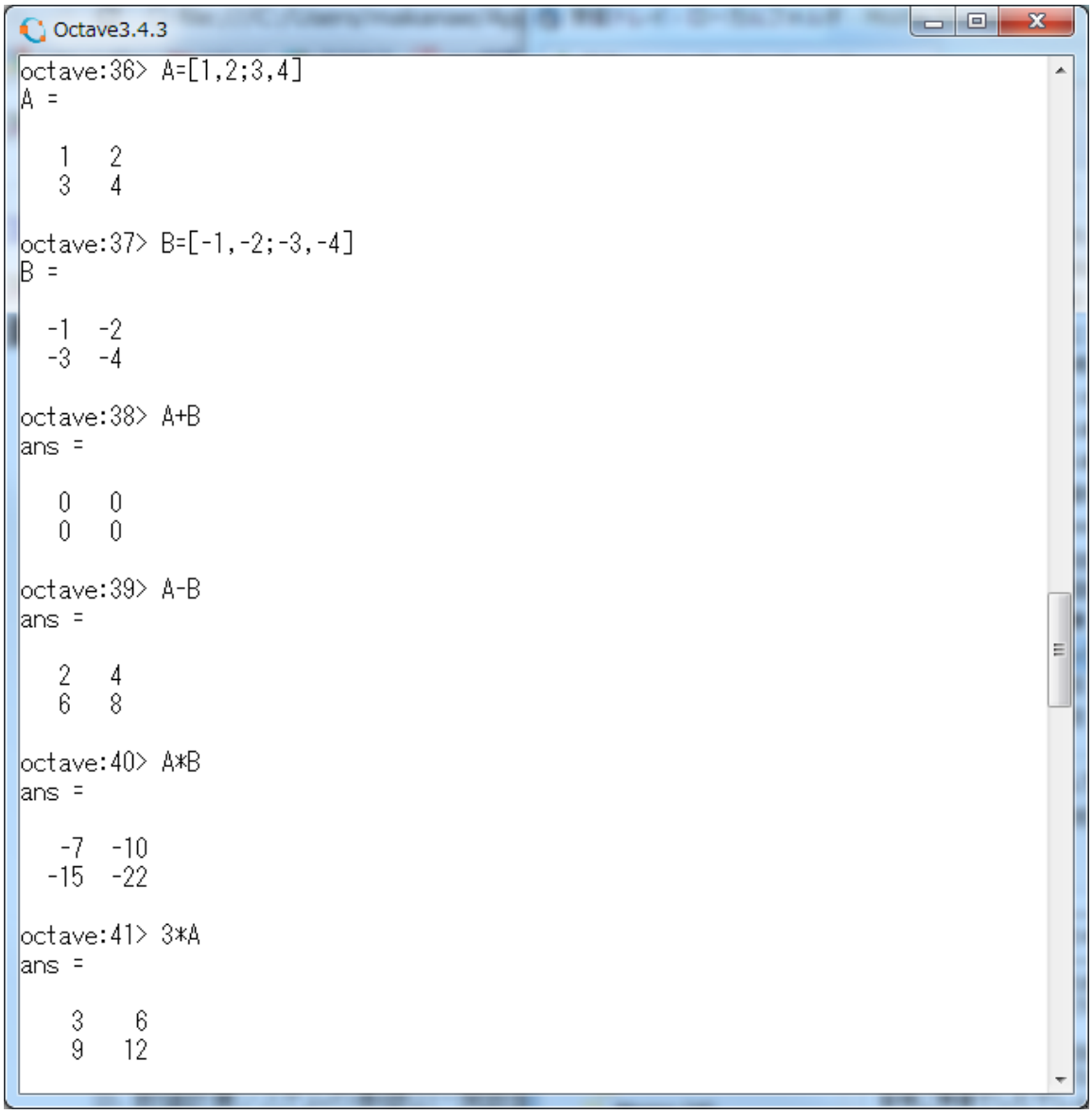
(8)X の逆行列 X^{-1} を求める：　inv(X)



(9)m×n の乱数行列　rand(m, n) を定義する

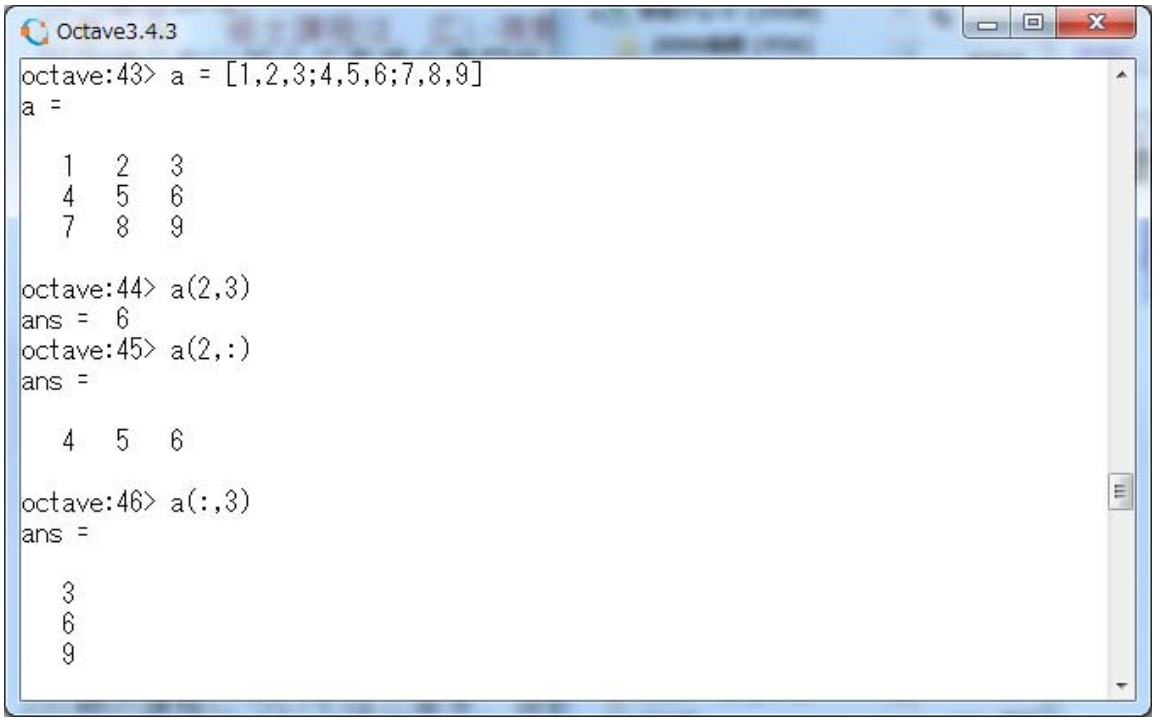


(10)行列の演算例　[行列 A と B の和・差・積・スカラー倍]



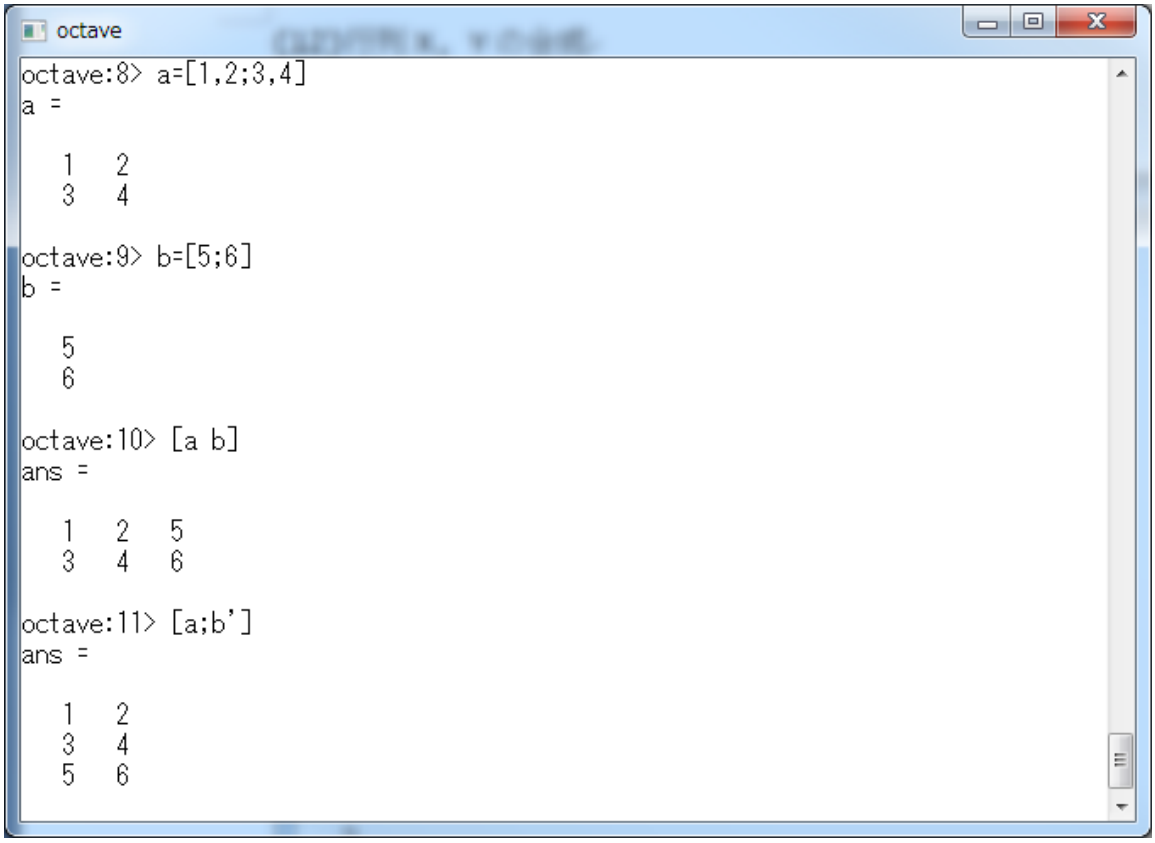
(11)行列要素の取り出し

(m,n)要素の取り出し：X(m,n)
n 列目の要素の取り出し：X(:,n)
m 行目の要素の取り出し：X(m,:)
※コロンは任意の要素を表す



(12)行列 X, Y の合成

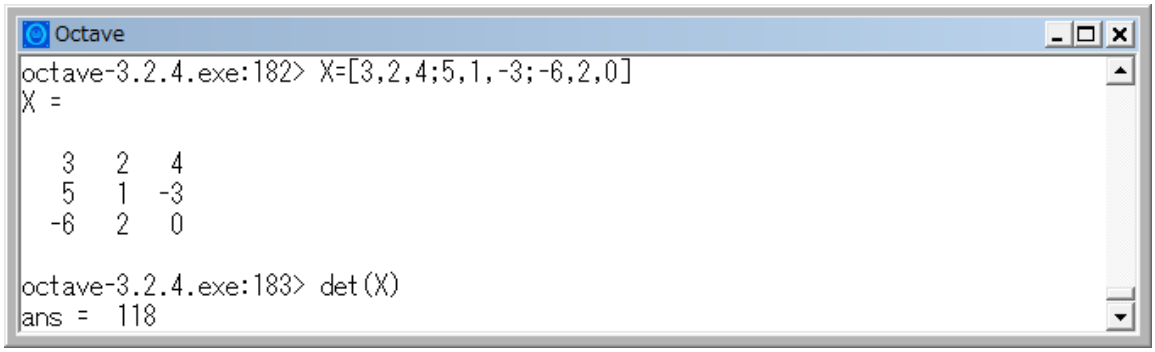
列の合成[X Y]
行の合成[X; Y]



(13)行列式の計算

行列式（determinant）の値は，det(X)により求めることができる．

$$X = \begin{vmatrix} 3 & 2 & 4 \\ 5 & 1 & -3 \\ -6 & 2 & 0 \end{vmatrix}$$



6. その他の基本操作

・直前またはそれ以前のコマンドを表示・編集する場合には，↑キーを押す.

・過去のコマンドを表示するには　history　を用いる.

・コマンドについてのヘルプは，help コマンド名　で利用できる．例：help history

・コマンドの中止は，ctrl＋C　である.

・Octave コンソールのテキストをコピーする場合には，ウィンドウのタイトルバーを右クリックし，編集→範囲指定を選択し，マウスあるいは Shift＋矢印キーで範囲を指定した後，Enter キーを押す．その後，別のソフトウェアの入力画面上で貼り付け（Ctrl+V）を行う.

【演習課題】

(1)行列 A の=$\begin{pmatrix}2 & 5\\ 3 & 1\end{pmatrix}$ を定義しなさい.

(2)行列 B=(2　−1)を定義しなさい.

(3)行列 B の転置行列 B′ を求めなさい.

(4)行列 C=$\begin{pmatrix}3\\ 1\\ 5\end{pmatrix}$を定義しなさい.

(5)行列 A に B′を列として，C′を行として合成した 3×3 の行列を求めなさい.

(6)以下の連立方程式の解を求めなさい

① $\begin{cases}2x-3y=1\\ x+4y=8\end{cases}$

② $\begin{cases}x+y+z=1\\ 2x-y+z=7\\ 3x+2y+z=3\end{cases}$

(7)次の行列式の値を求めなさい

① $\begin{vmatrix}4 & 6\\ -5 & 16\end{vmatrix}$

② $\begin{vmatrix}-3 & 5 & 2\\ 1 & 2 & 3\\ 4 & -5 & 1\end{vmatrix}$

実行したコードを PDF ファイルに出力し，
次週の講義前までに oyojoho@gmail.com　までメールで提出すること

数値計算システム Octave の基礎(2)

－関数定義とグラフィックス－

1. 関数の定義

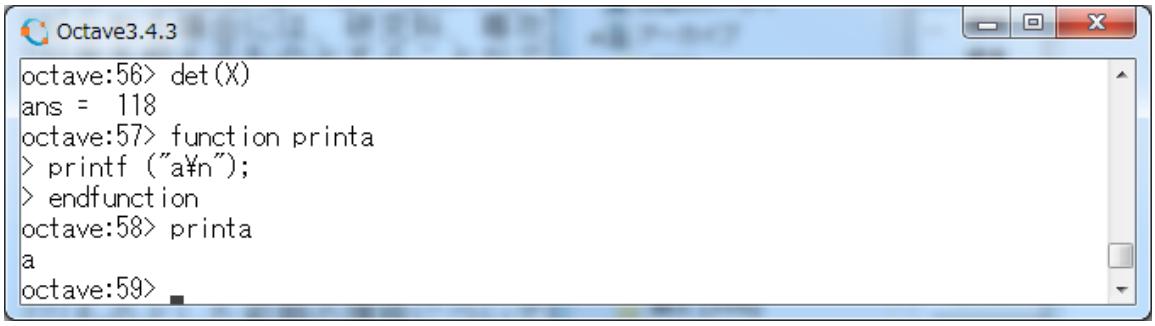
(1)戻り値のない場合

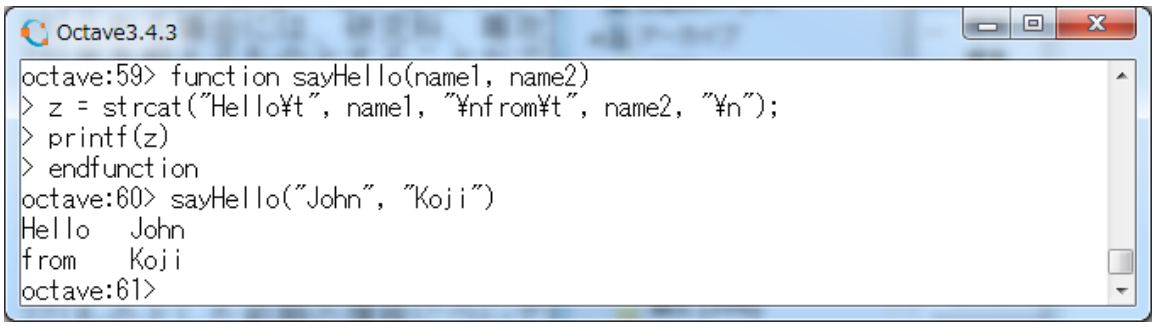
Octave 上でのユーザー定義関数は以下により行う.

```
function name
  body                          ※body には定義する関数を記述する
endfunction
```

あるいは

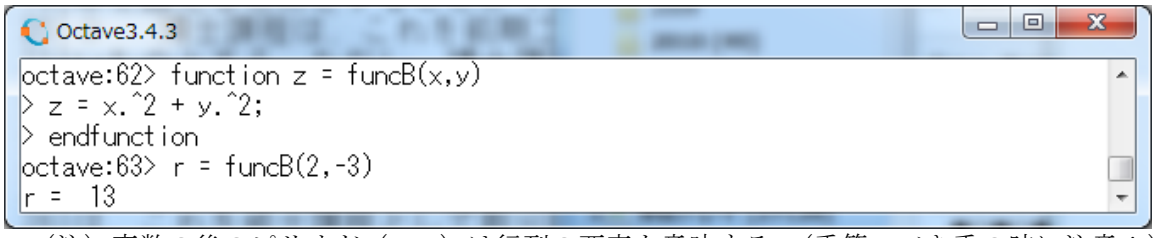
```
function name (arg-list)        ※arg-list は，関数の引数をカンマで区切って記述する
  body
endfunction
```





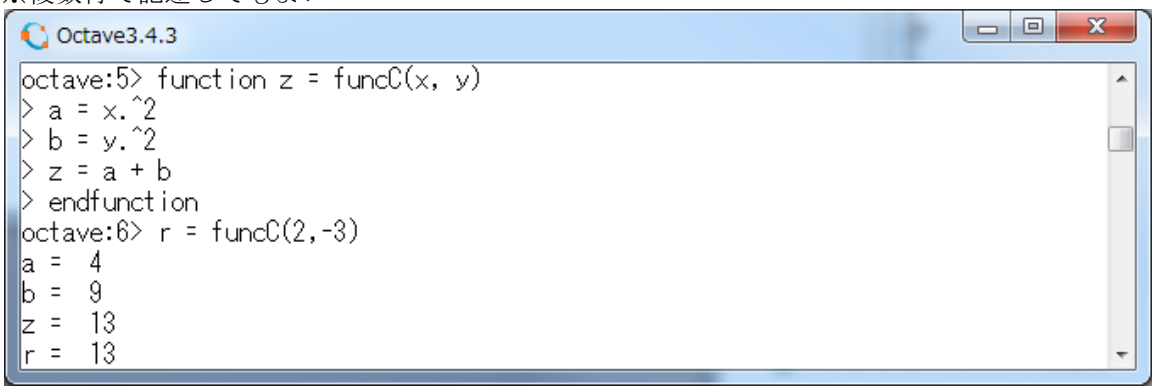
(2)戻り値 (retval)のある場合

```
function ret-val = name (arg-list)
  body
endfunction
```



（注）変数の後のピリオド（ . ）は行列の要素を意味する.（乗算，べき乗の時に注意！）

※複数行で記述してもよい

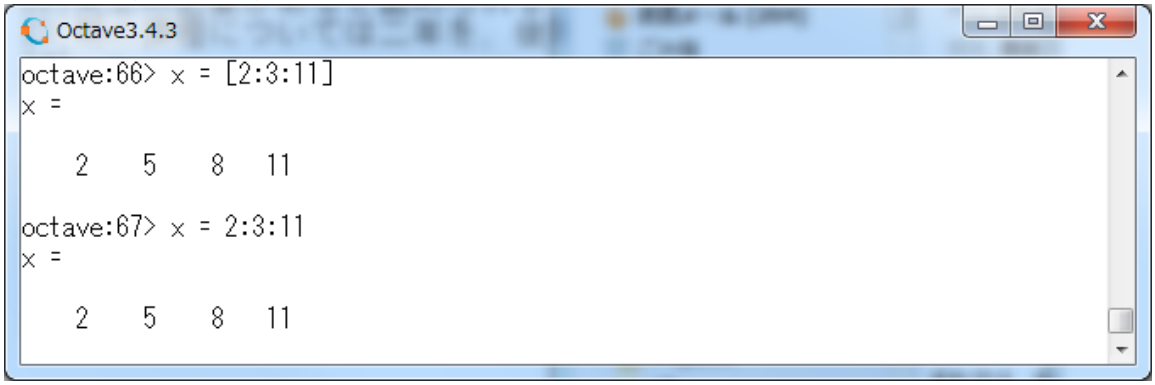


※関数内の変数はその関数内でのみで有効である．（⇔ `global`）

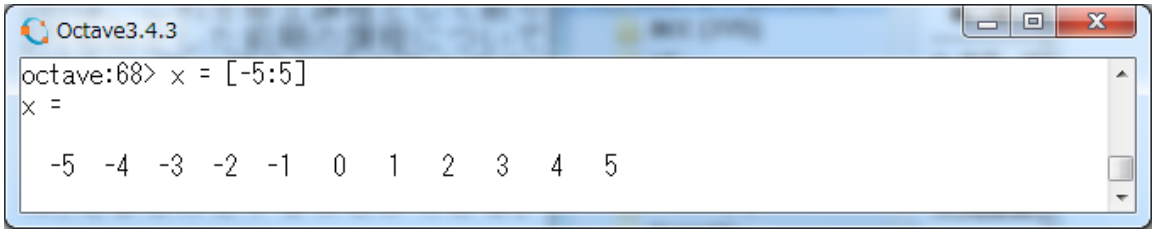
2. 等差数列

・等差数列は，[初項:公差:末項]（あるいは 初項:公差:末項）により生成される．

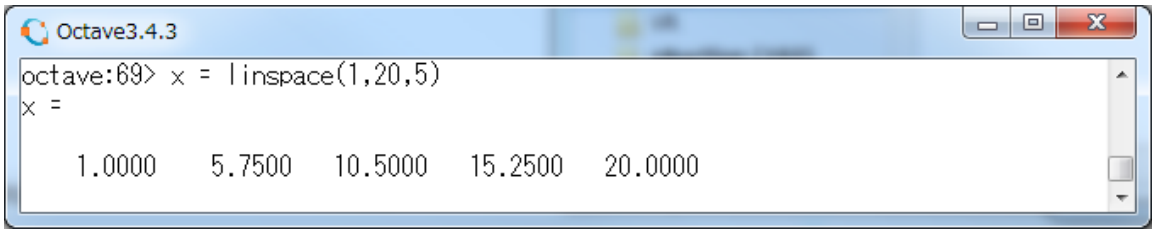
例）初項 2，末項 11，公差 3 の数列を定義する．



・公差を省略した場合には，公差は 1 の数列が生成される．



・項数を指定した数列を定義する場合には，linspace 関数を用いる
　linspace(初項,末項,項数)



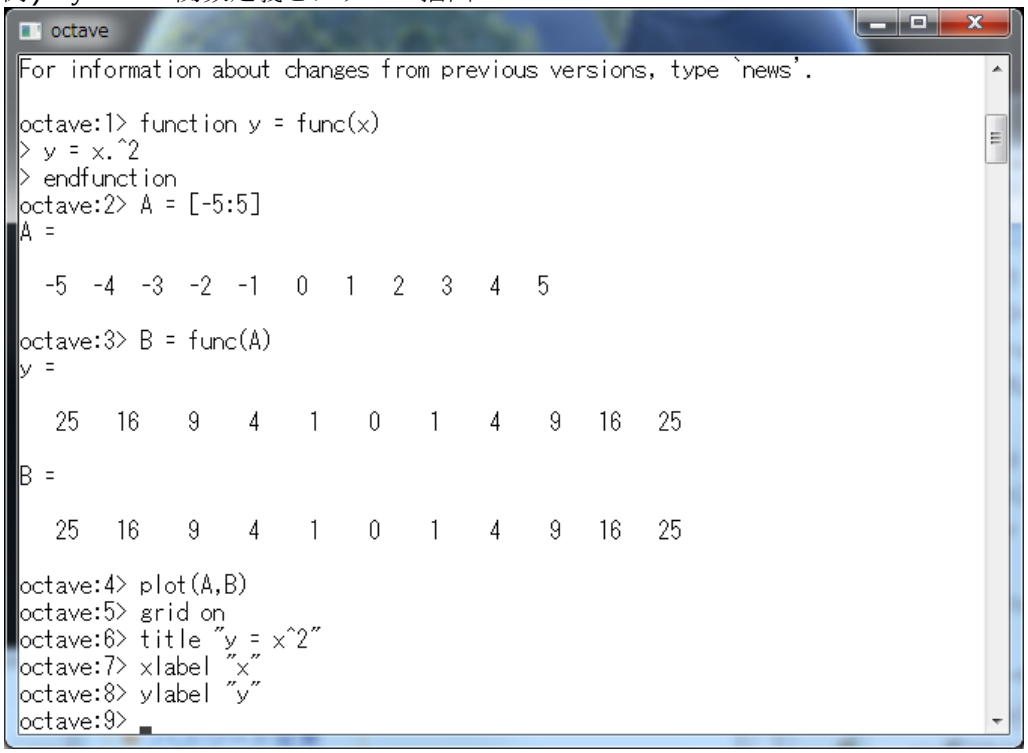
3. グラフィックス

・Octave のグラフ描画は gnuplot により行われる．
・２次元グラフの描画コマンドは plot，３次元グラフの描画コマンドは plot3　である．

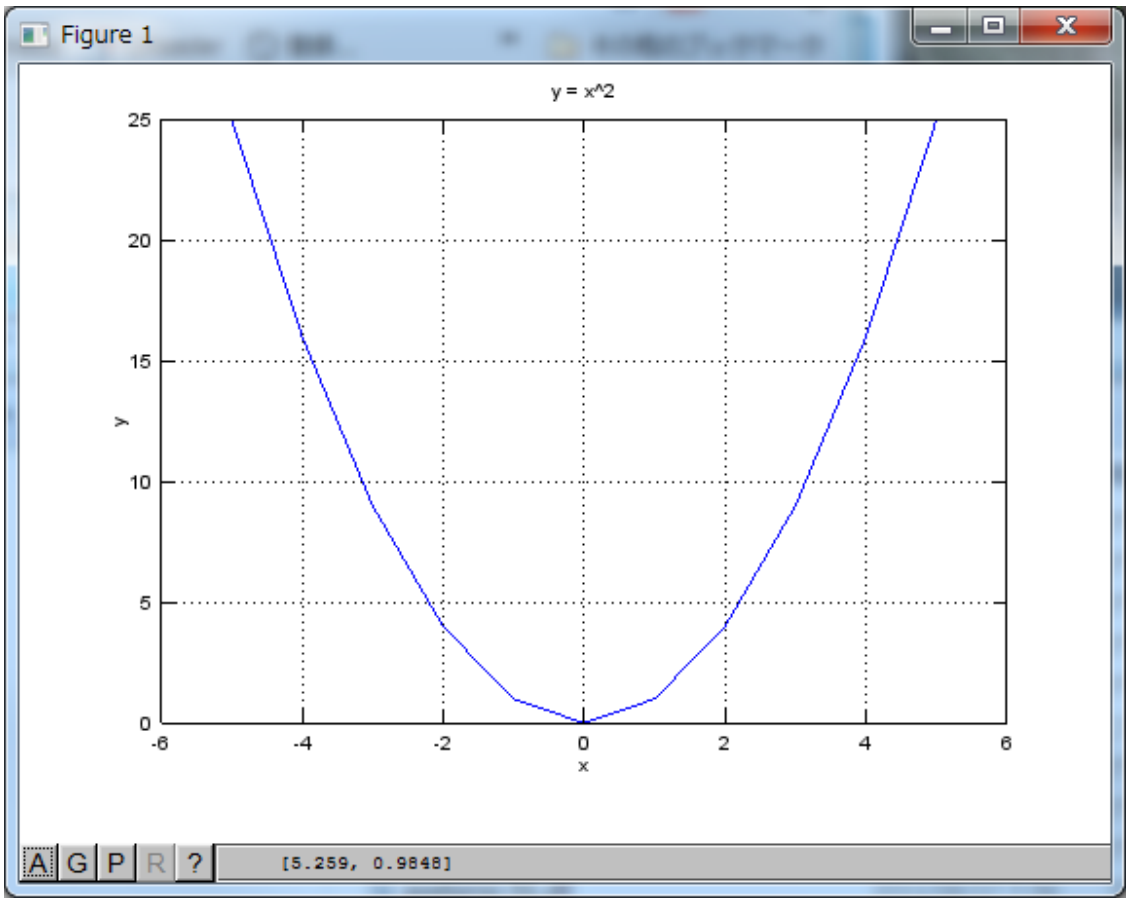
(1) ２次元プロット
・引数が１つの場合　plot(y)　　※x の値は要素のインデックスとなる．
・引数が２つ以上の場合　plot(x, y, fmt …)　※ fmt は省略可能

※fmt の設定（初期値（省略時）は直線プロットスタイルである）
`-'：直線プロットスタイル
`.'：ドットプロットスタイル
`@'：ポイントプロットスタイル
`-@'：直線・ポイントプロットスタイル
`^'：インパルスプロットスタイル
`n'：もし n が 1 から 6 の範囲にあれば，プロット色として解釈する。
　　(1:赤，2:緑，3:青，4：マゼンタ，5:シアン，6：茶色
`nm'：もし nm が 2 桁の整数の場合，n はプロット色，m がポイントスタイル
　　(1:*，2:+：3: o，4: x，5:house 6:there exists
`c'：もし c が"r"，"g"，"b"，"m"，　"c"，"w"のうちの 1 つならば，プロット色として解釈．
`;title;'："title"はキーについてのラベル

例)　$y = x^2$の関数定義とグラフの描画



※ grid on　はグラフ背景への格子の表示
※ title, xlabel, ylabel はそれぞれタイトル，x 軸ラベル，y 軸ラベルの文字列を設定する．



グラフの出力（印刷・他ソフトウェアへの貼り付け等）について

Octave3.6.1 の場合：Alt+PrintScreen でコピーして貼り付ける.

Octave3.4.3 の場合：メニューから Save で任意の拡張子（.pdf, .png. jpg 等）を付して保存する.

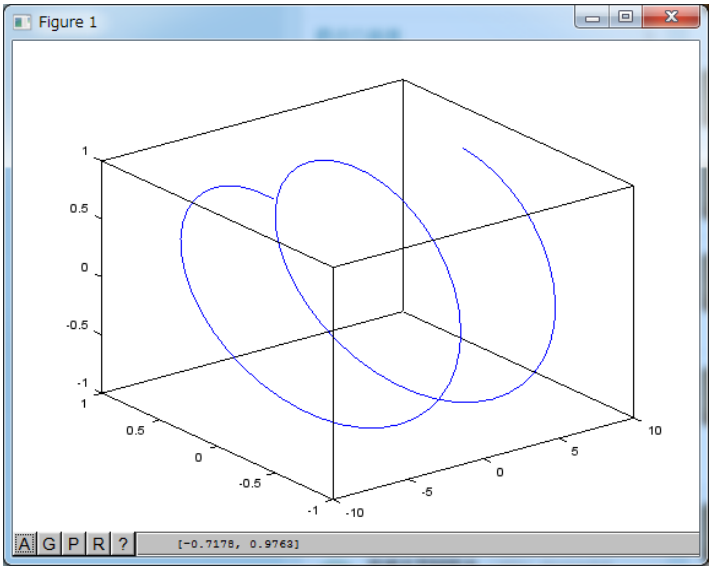
(2) 3 次元プロット

3 次元曲線の例

$$\begin{cases} x = t \\ y = \sin(t) \\ z = \cos(t) \end{cases}$$

$-2\pi \leq t \leq 2\pi$のときのグラフを描く.

```
octave:1> t = [-2:0.05:2]*pi;
octave:2> x = t;
octave:3> y = sin(t);
octave:4> z = cos(t);
octave:5> plot3(x,y,z);
```



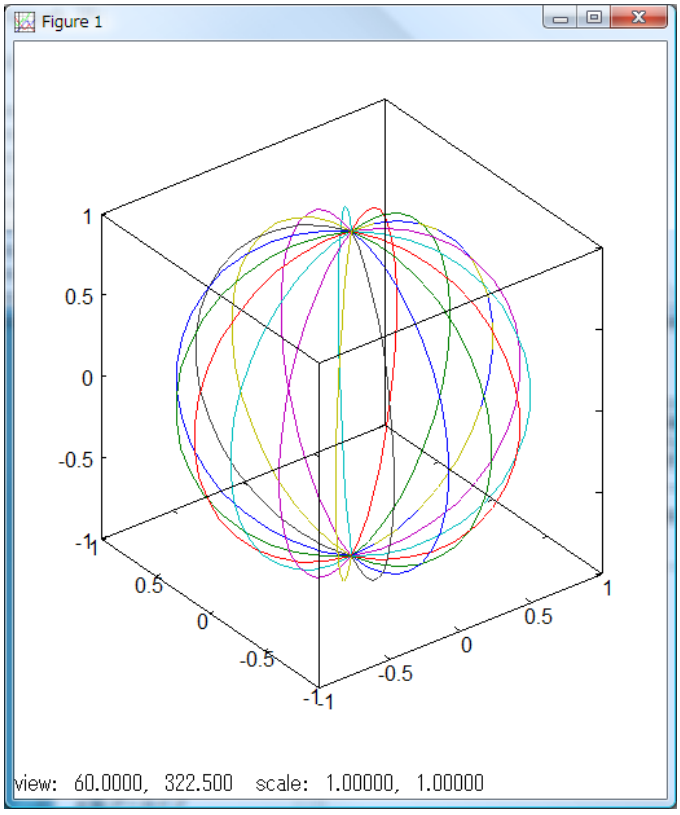
[演習課題]
(1)function を用いて任意の 3 次関数を定義し，それを plot を用いて描画しなさい．
(2)function を用いて任意の関数を定義し，それを plot を用いて描画しなさい．
(3)任意の 3 次元曲線を定義し，plot3 を用いて描画しなさい．
(4)plot3 を用いて球面を描画しなさい．(sphere あるいは ellipsoid を用いないこと)
(5)任意の 3 次元曲面を定義し，plot3 を用いて描画しなさい．

(1)～(5)について実行したコードとプロット画面を MS Word あるいはワードパッドに貼りつけ（PDF にしてもよい），次週の講義前までに oyojoho@gmail.com までメールで提出する．

(3)のヒント
・球のパラメトリック表現

$$\begin{cases} x = R\cos(u)\cos(v) \\ y = R\cos(u)\sin(v) \\ z = R\sin(u) \end{cases} \quad \text{ただし，} -\frac{\pi}{2} \le u \le \frac{\pi}{2} \quad , \quad 0 \le v \le 2\pi$$

・ある行列 x と同じ行数・列数をもつ全要素値 1 の行列をつくる　ones(size(x))
・axis square　（座標系を正方形，あるいは立方体にする）



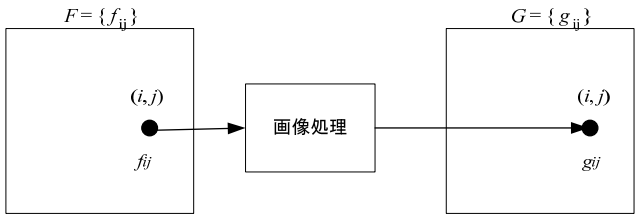
画像処理の基礎（Octave を用いた画像解析(1)）

1. 画像処理のモデル

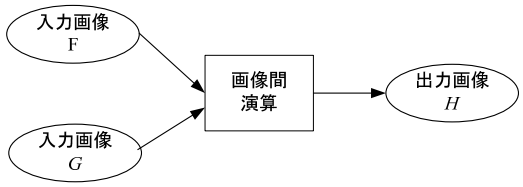
画像処理は，ディジタル画像（入力画像）に対して何らかの処理を加え，新たな画像（出力画像）をつくり出す作業である．

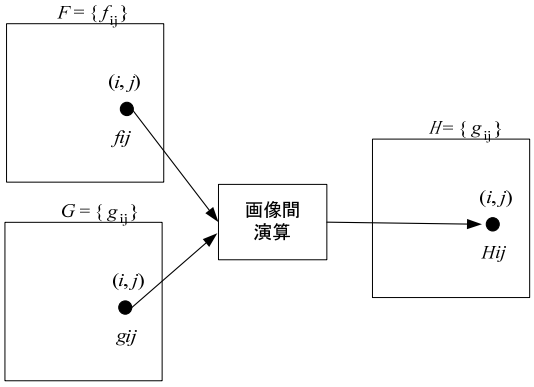
M 行 *N* 列の画素からなるディジタル画像 *F* は，
$F = \{ f_{ij} \}$
f_{ij} =第 *i* 行 *j* 列の画素の濃度値
$i = 1, 2, 3, \cdots, M \quad j = 1, 2, 3, \cdots, N$

(1)　1 入力 1 出力の画像処理モデル



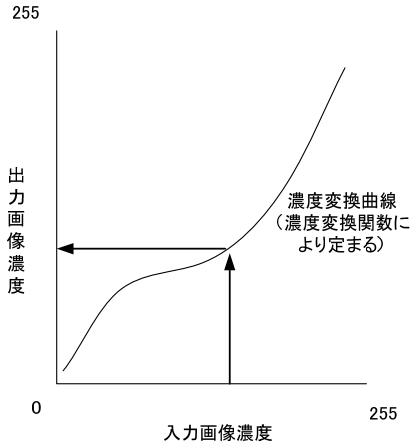
(2) 2 入力 1 出力の画像処理モデル



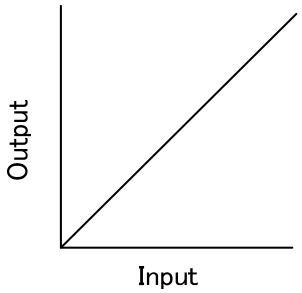


２．濃度変換

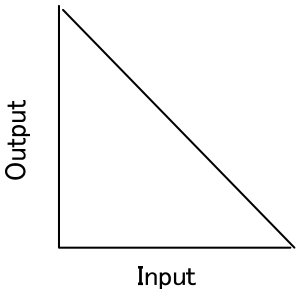
入力画像の各画素の濃度値に対する出力画像の濃度値を求める濃度変換関数を定め，それに基づき処理を行う．



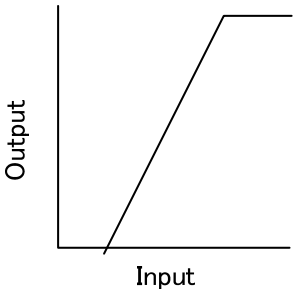
(1)　リニア



等値出力

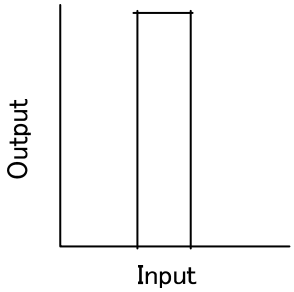


色調反転

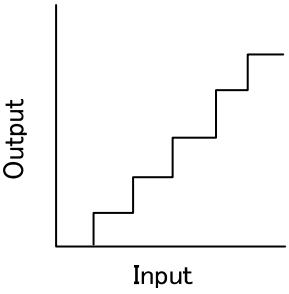


コントラスト調整

(2)　レベルスライス

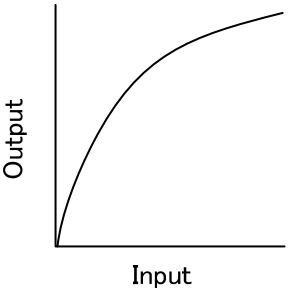


(3)階調化



(4)ガンマ補正

$$y = 255\left(\frac{x}{255}\right)^{\frac{1}{\gamma}}$$



3. Octave を用いた濃度変換

(1)画像ファイルの読み込み

```
[img, map, alpha] = imread( filename )
```

img：画像データの配列
（カラー：m×n×3，グレー・白黒：m×n の配列）
map：カラーマップ：カラー画像の場合は不使用
alpha：アルファチャンネルの配列

例：

```
[img, map, alpha] = imread("z:/images/photo1.jpg");
```

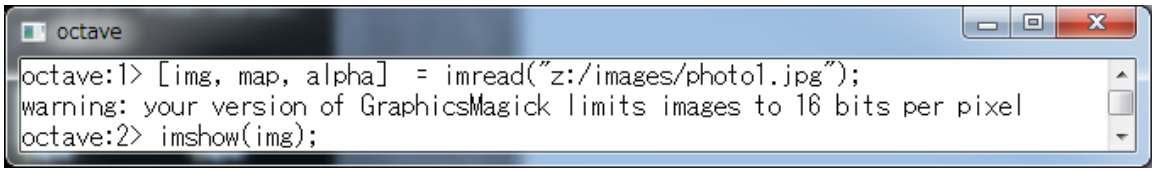
※img = imread("z:/images/photo1.jpg"); でも可

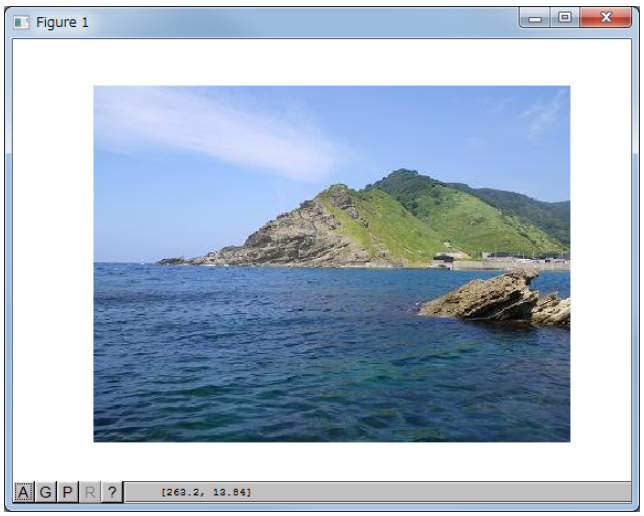
(2)画像の表示

```
imshow( img )
```

※ image(img)でも可

例：
※あらかじめ画像ファイル picture1.jpg を z ドライブの images フォルダに置いておく．
使用する画像の画素数は画面内に収まる適当なサイズとすること



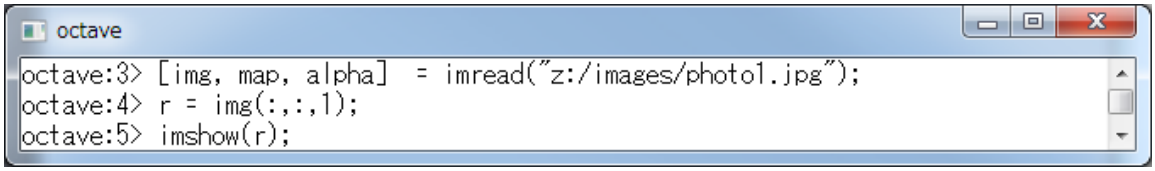


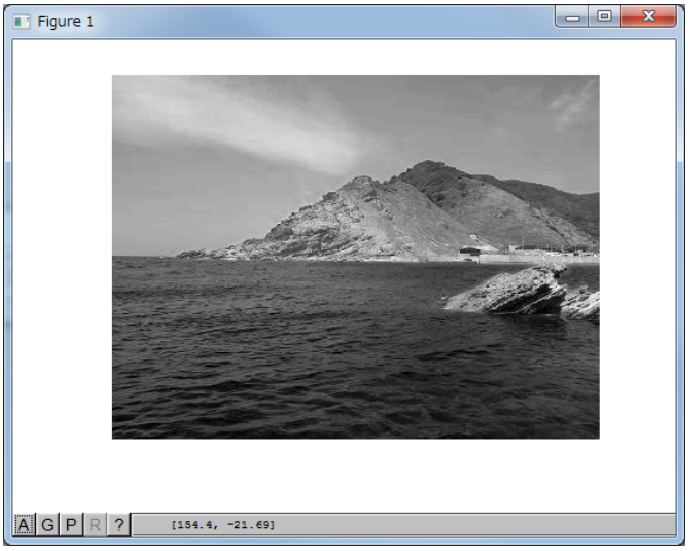
(3)カラー画像からの色成分の取り出し

カラー画像の場合，Octave 上で画像ファイルは 3 次元の配列に格納される．

R（赤）成分：(:, : ,1)　G（緑）成分：(:, :, 2)　B（青）成分：(:,:,3)

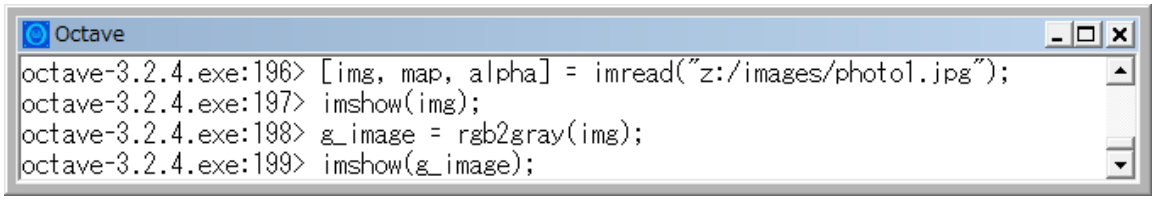
例：R 成分のみを取り出す





(4)カラー画像のグレースケール画像への変換

```
rgb2gray(img)
  [img, map, alpha] = imread("z:/images/photo1.jpg");
```

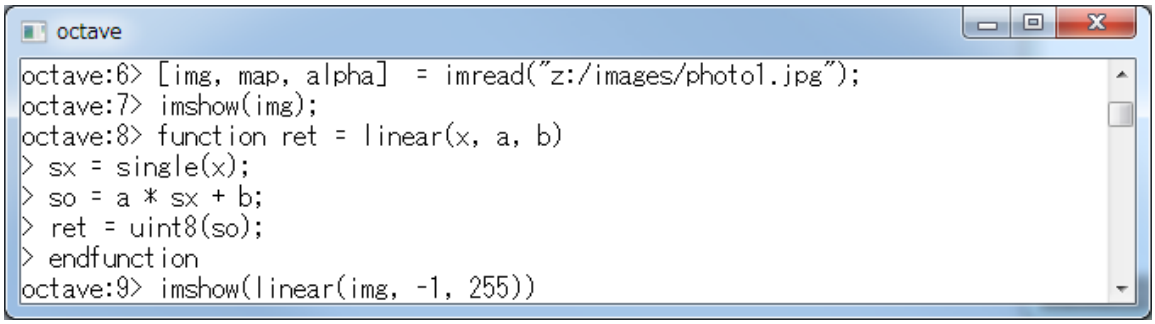


※Octave 上で pkg rebuild -auto image　を実行した後，
Octave を再起動することにより使用できる（一度のみで良い）

演習問題
(1)画像情報を格納した img から G 成分，B 成分を取り出し，それぞれ g, b に格納しなさい．
(2)r, g, b の画像を縦に並べて出力しなさい．
(3)R 成分をそのままに，G 成分と B 成分を交換した画像 img2 を作成しなさい．

(5) 濃度変換関数

y = a x + b 型の一次関数（リニア）による濃度変換関数を作る.



※画像の配列は uint8 型（符号無し 1 バイト整数）で定義されることから，画像配列を含んだ式では 0～255 の整数の範囲を超えて計算することができない．そのため浮動小数点数 (single または double)に変換して計算を行い，計算結果を再度，uint8 型に戻す処理をしている.

型変換関数の例：

整数への変換： int8(x), int16(x), int32(x), int64(x)
符号無し整数への変換：unit8(x), uint16(x), uint32(x), uint64(x)
浮動小数点数： single(x), double(x)
複素数： complex(x), complex(re, im)

参照：

(6) 画像の保存

```
imwrite(image, filename)
```

image：保存する画像
filename: 保存する画像のファイル名 （形式は.jpg, .png, .gif 等）
（例：imwrite(img, "z:/images/test.jpg") ）

演習課題

濃度変換関数のうち，下記の画像を作成する関数を作成し，それにより処理した画像の例を示しなさい.（関数のグラフも示すこと）

(1)ガンマ補正関数（グラフ）と処理画像（γの値は任意に指定できる関数とする）
(2)階調化関数と処理画像
（画像はグレースケール画像でもよい，階調数は４，または任意に指定できる関数とする）
(3)自らが考えた関数（グラフ）とその処理画像（その効果についても記載すること）

次週の講義まで上記をまとめてメールにて提出すること

※小数点以下切り下げする関数： floor(x)

(スクリプト・関数)M ファイルの作成

1. M ファイルの種類

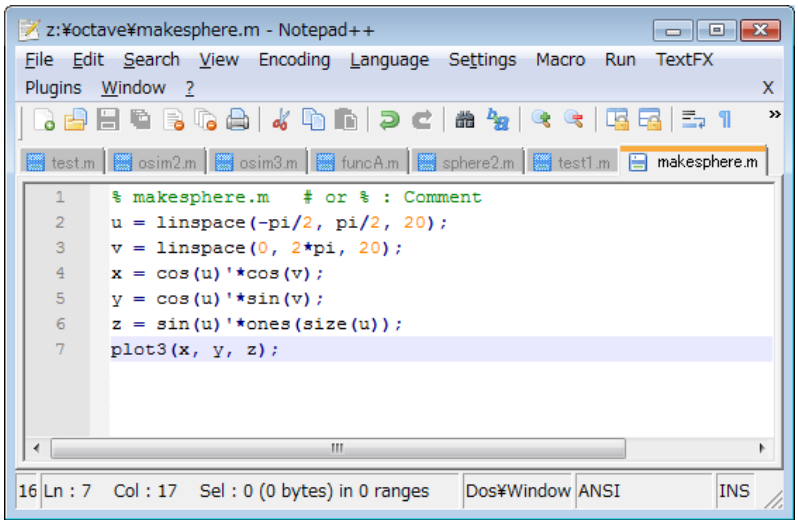
・関数 M ファイル：関数を記憶できる（引数と戻り値をもつことができる).
・スクリプト M ファイル：コマンドをスクリプトとして記憶できる.
・M ファイルの拡張子は `.m` である.

2. M ファイルの作成

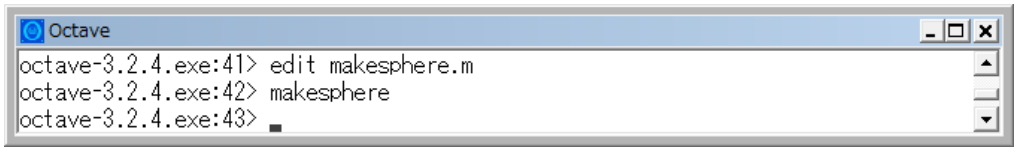
・メモ帳などのエディタで作成する.
（ファイルの種類は「すべてのファイル(*.*)」を選択し，xxxxx.m として保存する.）

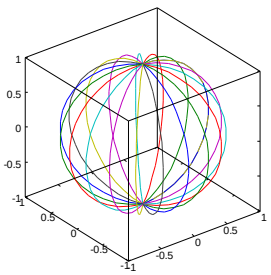
※octave 内でのディレクトリ(フォルダ)の扱い
現在のディレクトリを表示：`pwd`
ディレクトリの移動：`cd` 移動先　　※ 例：`cd z:/octave/script/`
ディレクトリの作成：`mkdir` ディレクトリ名　　※ 例：`mkdir octave` など

3. スクリプト M ファイルの例(引数無し)



・実行する場合には，Octave のコマンドラインからスクリプト名を入力する（.m）は不要.





4. 関数 M ファイルの例

Z:¥octave¥funcsphere.m - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

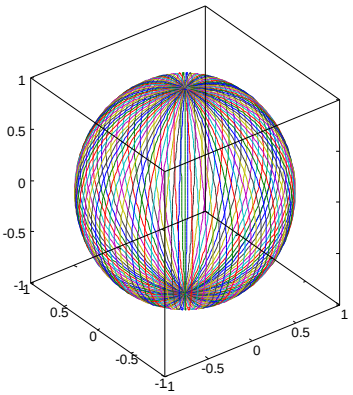
funcsphere.m | funcsphre.m | imgprocess1.m | fgamma.m | geomapmesh.m | geomapimshow.m | geomaploader.m

```
3  function ret = funcsphere (n)
4      error(nargchk(1,1, nargin));
5      u = linspace(-pi/2, pi/2, n);
6      v = linspace(0, 2*pi, n);
7      x = cos(u)'*cos(v);
8      y = cos(u)'*sin(v);
9      z = sin(u)'*ones(size(u));
10     axis square;
11     plot3(x, y, z);
12     ret = 0;   #if ret-val required
13 endfunction
14
```

267 chars 293 bytes 14 lines | Ln : 12 Col : 36 Sel : 0 (0 bytes) in 0 ranges | Dos¥Window | ANSI | INS

Octave

```
octave-3.2.4.exe:58> edit funcsphere.m
octave-3.2.4.exe:59> funcsphere(100)
ans = 0
octave-3.2.4.exe:60>
```



【補足】関数mファイルに関する補足事項

・メモ帳で作成する場合には、ファイルの種類を"すべてのファイル"にすることを忘れずに！(関数名`.m.txt` になってしまうので要注意)

・`geomapread` を実行した場合に `undefined...` のエラーが出る場合

`source` ***`filename.m`*** を実行してみる.

(***`filename.m`*** は作成した関数名)

画像処理の基礎(Octave を用いた画像解析(2))

1. 空間フィルタによる画像処理

(1)空間フィルタリング

・画像のある 3×3pixel の領域を選び，それぞれの画素値を $V_{x,y}$ (x, y は領域の中心を(0,0)としたときの画素座標)とする.

(-1,-1) $V_{-1,-1}$	(0,-1) $V_{0,-1}$	(1,-1) $V_{1,-1}$
(-1, 0) $V_{-1,-0}$	(0,0) $V_{0,0}$	(1, 0) $V_{1,0}$
(-1,1) $V_{-1,1}$	(0,1) $V_{0,1}$	(1, 1) $V_{1,1}$

(a)処理前の領域

・ここで座標 (0,0)の画素について，領域内の近傍の画素値を用いて新しい値を求める演算処理を考える.

・近傍の画素との演算を行うにあたり，空間フィルタを以下のように設定する.

(-1,-1) $f_{-1,-1}$	(0,-1) $f_{0,-1}$	(1,-1) $f_{1,-1}$
(-1, 0) $f_{-1,-0}$	(0,0) $f_{0,0}$	(1, 0) $f_{1,0}$
(-1,1) $f_{-1,1}$	(0,1) $f_{0,1}$	(1, 1) $f_{1,1}$

(b)フィルタ行列

・(a)の各画素の濃度値と空間フィルタを基に，下式による演算（積和）を行い，(a)の中心(0,0)に対応した値 V'0,0 を得る.

$$
\begin{aligned}
V'_{0,0} = {} & V_{-1,-1} \cdot f_{-1,-1} + V_{-1,0} \cdot f_{-1,0} + V_{-1,1} \cdot f_{-1,1} + \\
& V_{0,-1} \cdot f_{0,-1} + V_{0,0} \cdot f_{0,0} + V_{0,1} \cdot f_{0,1} + \\
& V_{1,-1} \cdot f_{1,-1} + V_{1,0} \cdot f_{1,0} + V_{1,1} \cdot f_{1,1} \\
= {} & \sum_{m=-1}^{1} \sum_{n=-1}^{1} V_{m,n} f_{m,n}
\end{aligned}
$$

・この処理を画像上の周辺部を除く全画素に行い，処理後の画像を得る.（カラー画像の場合には RGB 各チャンネルについて行う.）

・設定するフィルタ値，フィルタの大きさ（3×3，5×5，7×7 など）により，様々な効果を得ることができる.

(2)3×3 の空間フィルタの例

（元の画像）

例 1) シャープ化

0	-1	0
-1	5	-1
0	-1	0

元画像のピクセル成分が以下の通りであったとすれば,

12	17	18
11	22	33
9	25	35

その中央のピクセルの新しい成分は,

0×12＋(-1)×17＋0×18＋(-1)×11＋5×22＋(-1)×33＋0×9＋(-1)＋25＋0×35＝24

となる.

例 2) ぼかし（デフォーカス）の例

0.08	0.12	0.08
0.12	0.2	0.12
0.08	0.12	0.08

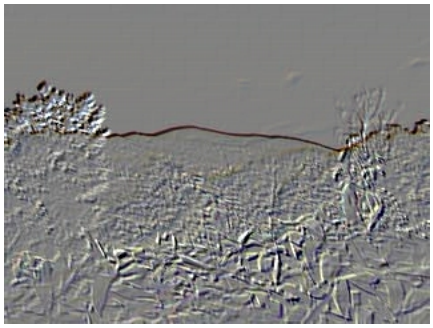
例 3）輪郭（エッジ）抽出

1	1	1
1	-8	1
1	1	1

例 4）エンボス

-1	0	0
0	0	0
0	0	1

さらに計算値に 128（グレー）を加える.



(3)Octave による空間フィルタリング

（注）起動時に画像処理関連パッケージをロードする． `pkg load image`

```
octave:1> pkg load image
octave:2>
```

・Octave 内でフィルタを配列として設定する.
例) `flt = [-1, -1, -1; -1, 9, -1; -1, -1, -1]`
・フィルタリング関数
`imfilter(filter, img)`
`fillter`：設定した `filter` 変数
`img`：画像名

例)

```
octave:5> img = imread("z:/images/photo1.jpg");
octave:6> imshow(img);
octave:7> axis square;
octave:8> flt = [-1,-1,-1;-1,9,-1;-1,-1,-1]
flt =

  -1  -1  -1
  -1   9  -1
  -1  -1  -1

octave:9> img2 = imfilter(img, flt);
octave:10> imshow(img2);
octave:11> axis square
```

（処理前）

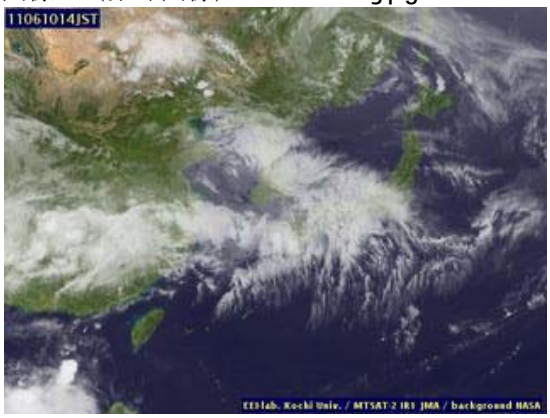
（フィルタリング後）

2. 2入力1出力の処理−2画像の合成−

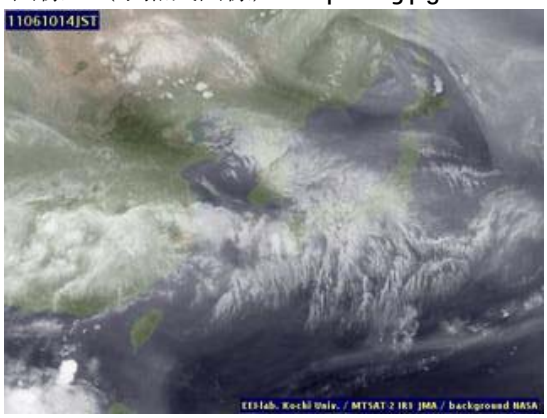
同じ大きさの２画像間の合成演算や差分演算は算術演算子(+, - など)を用いて容易に行うことができる.

気象衛星画像

画像 A（赤外画像）： infra.jpg



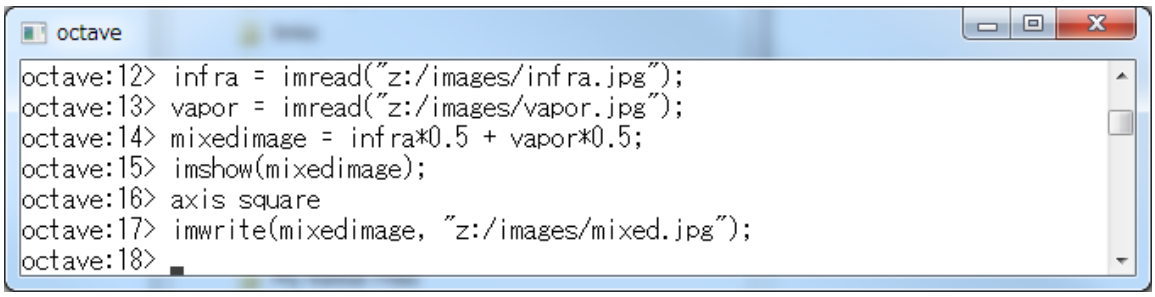
画像 B（水蒸気画像）：vapor.jpg



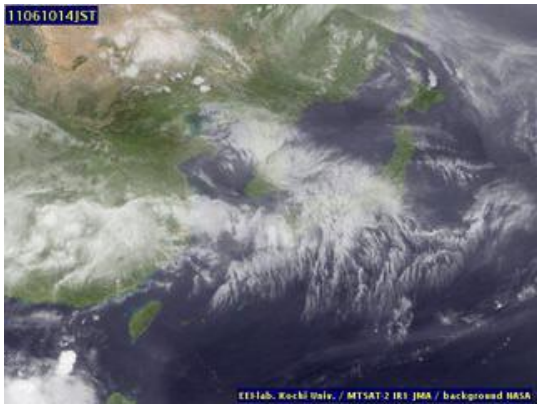
（高知大学気象情報頁より取得：http://weather.is.kochi-u.ac.jp）

※サンプル画像は http://www.makalab.org/photo/から取得可能

・画像合成の生成例（画像 A と画像 B を 0.5 の割合で合成）



出力画像：

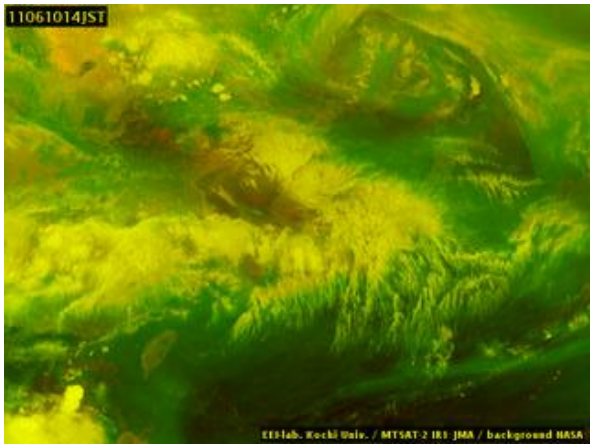


演習課題

(1)P.30～31 で例として挙げた空間フィルタをもとにした一連の画像処理を行い，それらのコードと画像を示しなさい．
（フィルタの設定値は任意に設定してよい）

(2)自らの考えで新しい空間フィルタを設定し，それによる画像を示しなさい．設定した空間フィルタとそれにより画像処理の効果を示すこと．
（複数のフィルタを重複して用いるものでもよい）

(3)P.33 の赤外画像を R 成分（赤），水蒸気画像を G 成分（緑）としてカラーで表示した画像を求めなさい．コードと画像を示すこと．ただし，赤外画像・水蒸気画像については RGB いずれか 1 チャンネルのみを用いるか，rgb2gray で 1 チャンネルに変換すること．）



（画像の出力例）

(4)2 枚以上の画像間演算を用いた効果的な画像処理の例を示しなさい．
コードと画像を示すこと．

作成したコードと画像をまとめて，PDF ファイルとして，
次回講義前までに oyojoho@gmail.com に提出すること．

※エンボスの場合には，計算上，負の値が生じる可能性がある．そのため img 自体を一度，double(img)あるいは single(img)により実数として imfilter に引渡し，負の値を含むことができるよう配慮する必要がある．また計算後の値についても，uint8( )により符号無しの 1 バイト整数に戻す必要がある．

※空の画像（RGB が全て 0）の画像は
uint8(zeros(縦画素数，横画素数，3))で作成できる．

※あるカラー画像(img)と同じ大きさの空の画像は，
uint8(zeros(size(image)))
で作成できる．

補足：

【注意】
自作の実行ファイル.m ファイルを動かす場合には，
その.m ファイルが存在するディレクトリ（フォルダ）に移動する必要がある．

【Octave 上でのディレクトリの移動】

任意のディレクトリに移動する　　cd 移動先ディレクトリ
例）z ドライブの最上位（ルート）に移動　cd z:/
z ドライブの下位の任意のディレクトリ dir1 に移動　　cd z:dir1

ひとつ上の階層に移動する　　cd ..

現在の位置を示す　　pwd
ファイルの一覧を表示する　　ls　または　dir

【画像処理関係 image パッケージのロード】

演習室の PC の場合：
http://www.makalab.org/ojb/　から
以下のファイルを Octave の作業用のフォルダにダウンロードする
imfilter.m
padarray.m

ダウンロードしたフォルダに移動すれば実行可能である．

自宅の PC の場合
画像処理関連 image パッケージをロードする．

```
pkg load image
```

自動的に読み込むように設定する場合

```
pkg rebuild -auto image
```

上記の設定により，imfilter や rgb2gray などのコマンドが使用できる．

【前回の課題に関する補足】
前回の課題でガンマ補正関数を作成し，実行した場合に，
"error: gamma: invalid conversation of FloatNDArray to FloatMatrix"
などのエラーが生じる場合があります．
これは行列自体をべき乗しようとして問題が生じているので，
変数にピリオドを付して行列の要素に対して計算をするようにしてください．
（例えば配布資料 P.15 および P.18 のべき乗のある関数を参照）

ファイル入出力

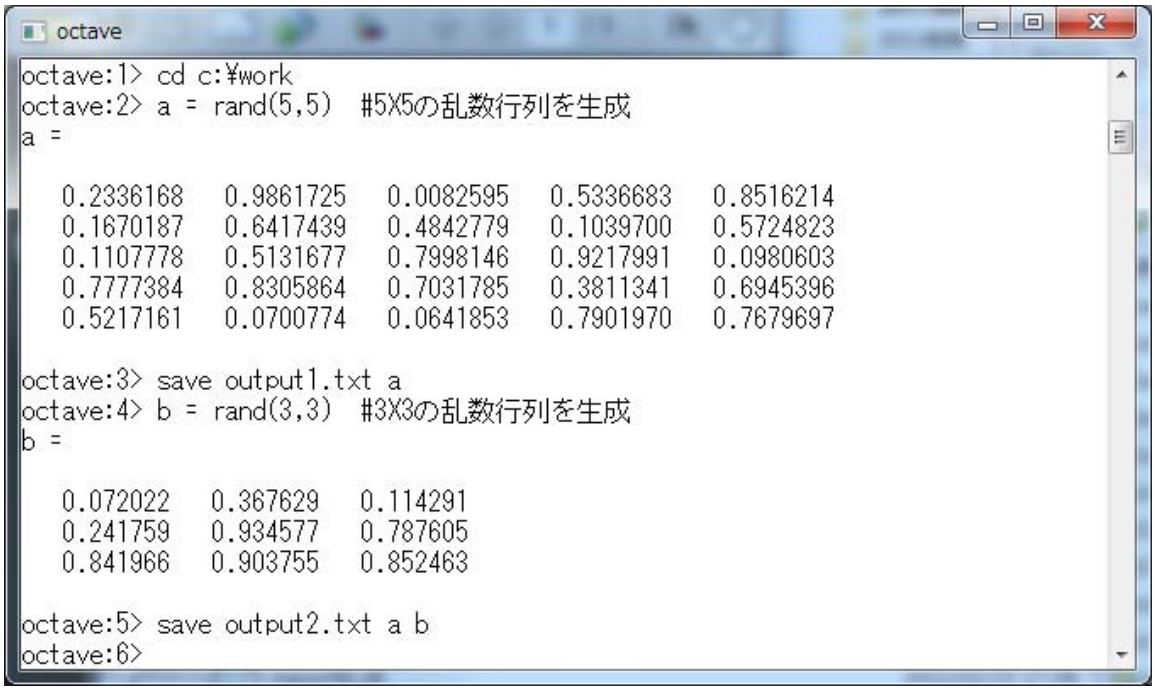
1. テキストファイルの入出力

(1)ファイルへの書き出し

save *options file v1 v2* ….

options（オプション，省略可能）
-text　テキストデータフォーマットでデータを保存する（デフォルト）
-binary　バイナリデータフォーマットでデータを保存する
filename：ファイル名（クォーテーションでくくる必要はない）
v1　v2　…　：出力する変数名（省略可；省略した場合は全ての変数，
ワイルドカード：?，文字列に合致：*も利用可能）

例 1）テキストファイルへの書き出し



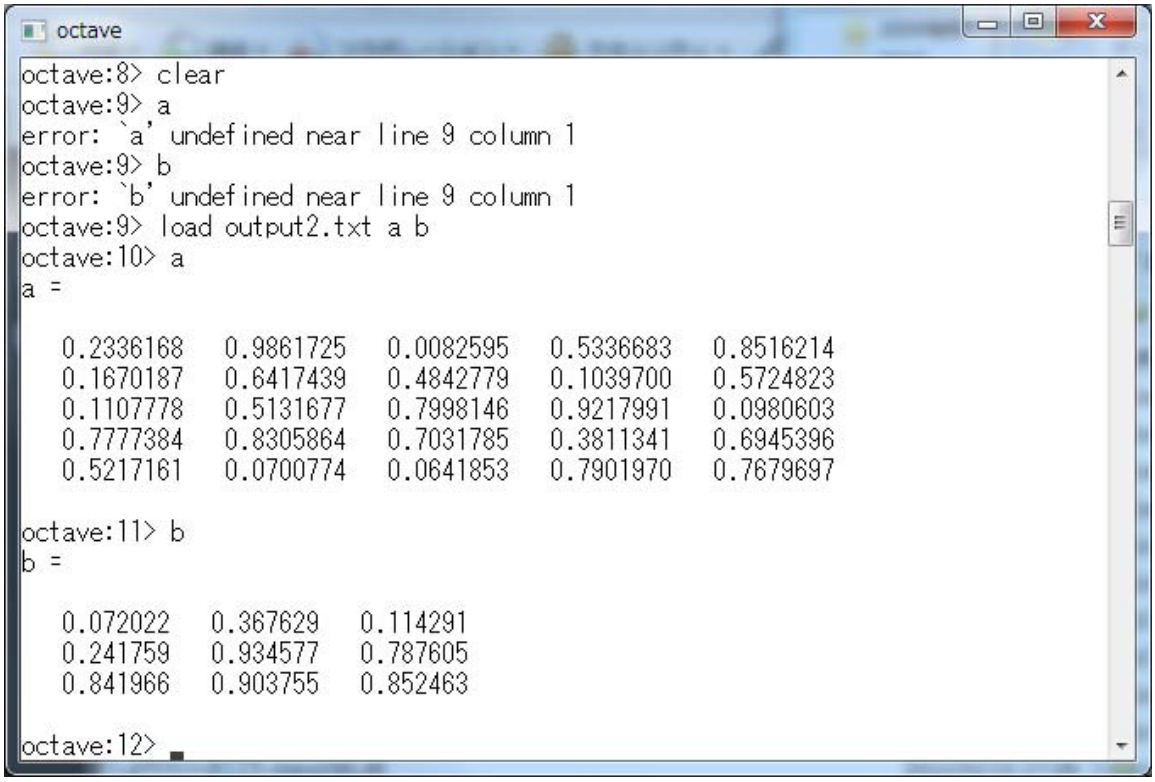
※保存したファイルをメモ帳などで開き，確認してみること．

(2)ファイルの読み込み

load　*options file v1 v2*, ….

options（オプション，省略可能）
-text　テキストデータフォーマット（デフォルト）
-binary　バイナリデータフォーマット
filename：ファイル名（クォーテーションでくくる必要はない）
v1 v2, …：変数名（省略可，指定した場合には合致したもの）

例 2）ファイル読み込みの例



※clear ： 全ての変数を消去する

(2) CSV ファイルの入出力

MS-Excel 等とのデータ交換を行う場合には，CSV（comma-separated values）形式を用いるのが容易である.

CSV での書き出し

```
dlmwrite('filename.csv', 変数名, ',')
```

CSV の読み込み

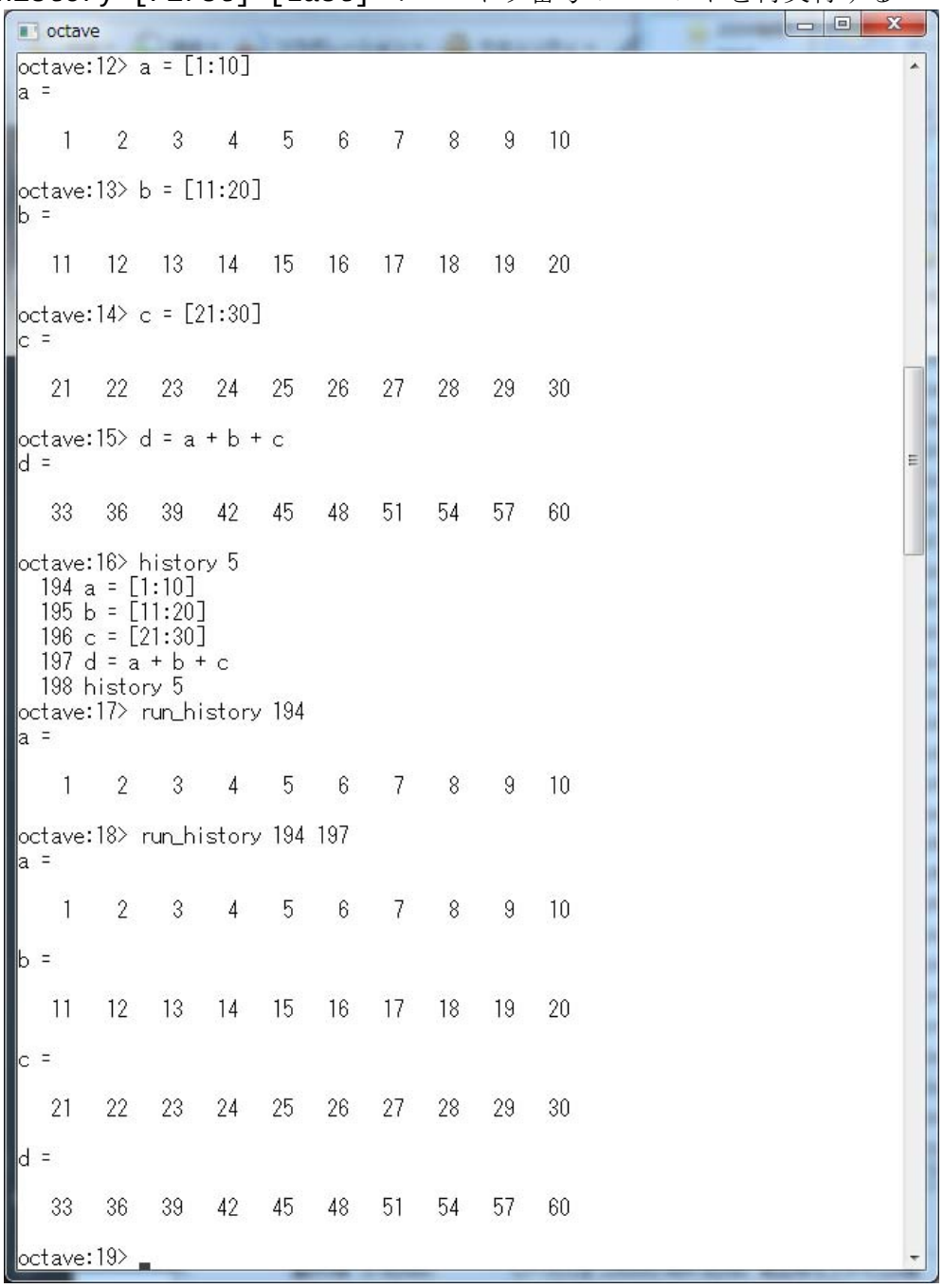
```
変数＝dlmread('filename.csv', ',')
```

※ファイル名 filename.csv はクォーテーション('または")でくくる.
',' は区切り記号がカンマ(,)であることを表す.
（スペース区切りの場合は' 'とする.）

(3) ヒストリの保存

実行したコマンドの履歴は `history` コマンドで参照できる.

`history` :コマンド履歴を表示する
`history N` : 最新の N 個のコマンドを表示する. 例 : `history 10` など
`history –w` ファイル名 : `history` の書き出し 例 : `history –w abc.txt`
`run_history [First] [Last]` : ヒストリ番号のコマンドを再実行する



Octave による地形情報の処理

1. 数値地図(標高)の読み込みと表示

関数 map = geomapread('filename')

指定したファイル名 filename の地形データを読み込み,
標高値のみにより構成される配列を生成する

```
function map = geomapread(fn)
      myfile = fopen(fn,"rt");  #指定されたファイルを読み込み専用で開く
      line = fgetl(myfile);     #myfile から１行目を取り出す
      name = line(1:4);      #メッシュコード
      scale = str2num(line(7:11));    #スケール
      year = str2num(line(19:23));    #数値化年紀
      height_n = str2num(line(27:29));  #東西方向の点数
      width_n = str2num(line(24:26));   #南北方向の点数
      map = [];
      for h = [1:height_n]
            line = fgetl(myfile);
            l = [];
            for w = [10:5:width_n*5+5]
                  level = str2num(line(w:w+4))/10;
                  if (level== -999.9)
                        level = 0;
                  endif
                  l = [l level];
            endfor
            map = [map;l];
      endfor
endfunction
```

※上記関数を m ファイル（geomapread.m）として作成し，c:\work の中に保存する.
※地形データ(メッシュコード.SEM)は講義で指定する URL からダウンロードする.
※海には-9999（-999.9m）が入っているが，ここでは 0m として変換している.

2. 数値地図(標高)の表示

実行例：

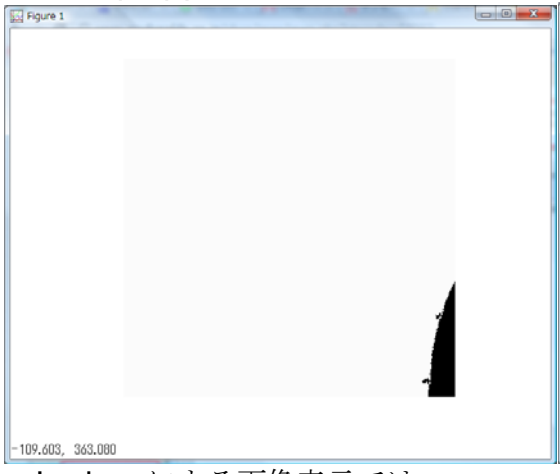
```
octave:1> cd c:¥work
octave:2> map = geomapread('5740.sem');
octave:3> more on
octave:4> int16(map)
```

more on は改ページ制御をする場合に使用する．(⇔more off)

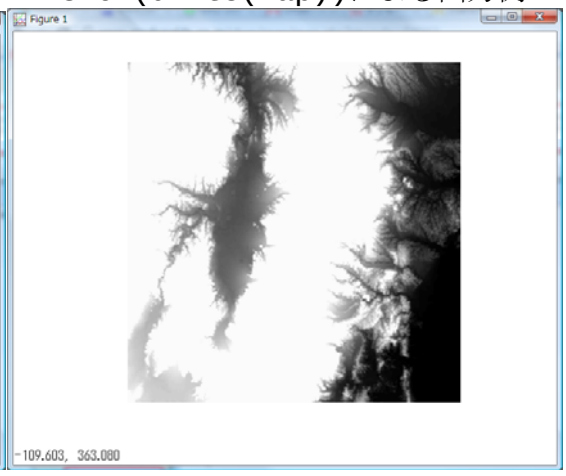
整数値 int16(map)として表示

```
ans =

 Columns 1 through 13:

   208   204   218   252   278   305   330   403   528   556   539   589   616
   211   216   247   269   269   282   337   415   451   481   544   620   650
   213   225   250   310   364   339   357   452   498   530   627   704   744
   218   231   247   303   377   412   437   496   593   665   742   804   843
   221   225   262   307   358   448   548   583   613   741   865   935   941
   229   224   267   306   353   462   563   621   661   736   849   919   905
   232   232   249   280   337   414   472   545   620   705   786   791   809
   241   237   242   280   328   361   407   439   476   605   711   747   833
   261   242   245   290   351   389   424   484   526   556   644   785   921
   278   246   249   277   356   428   472   568   658   689   733   802   899
   287   257   257   256   332   444   512   572   701   849   892   879   895
   313   290   261   252   298   377   447   551   721   875   942   947   960
   345   296   252   267   273   344   428   498   645   783   849   889   943
   340   291   262   273   286   405   516   546   626   709   783   841   875
   337   319   309   278   296   412   533   619   702   740   785   825   866
   383   345   325   292   282   360   489   589   668   714   724   796   909
   416   351   323   308   280   327   433   495   564   650   721   805   868
   414   362   348   349   307   342   408   450   541   657   757   791   768
   404   393   404   403   366   380   388   434   539   617   662   675   669
   412   419   428   432   420   381   334   363   470   525   520   540   592
-- less -- (f)orward, (b)ack, (q)uit
```

imshow(map)による出力例



imshow(uint8(map))による出力例

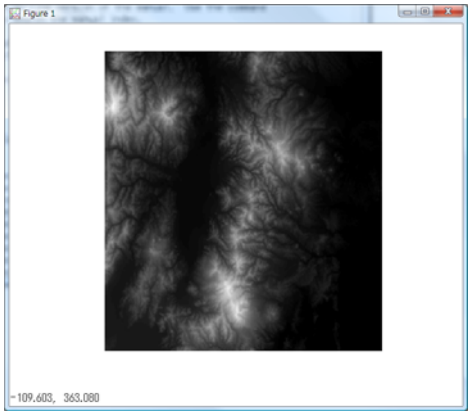


・imshow による画像表示では，
整数の配列（例えば uint8）を与えた場合には 0~255，
実数の配列（例えば double）を与えた場合には 0～1.0 をスケールとしている．
・左の出力例では，map には実数が入っていることから，
1 を超える値を有する画素は 1.0 として扱われ，全て白として表現されている．
・右の出力例では，map には整数が入っていることから，
255 以上の画素は全て 255 として扱われ，全て白として表現されている．

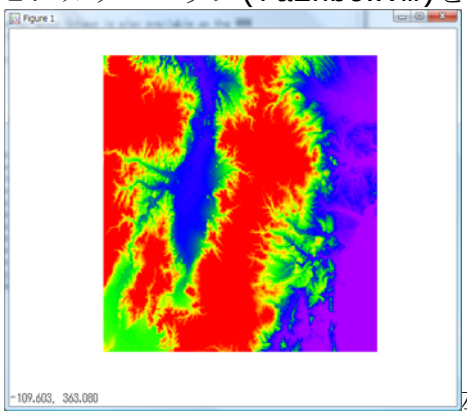
課題：以下についてコードおよび出力画像を示しなさい.

A.地形の 2 次元表現

1．読み込んだ数値地図を，最高地点を白(255，または 1.0)，最低地点（0m）を黒（0，または 0.0）として画像として表現しなさい.

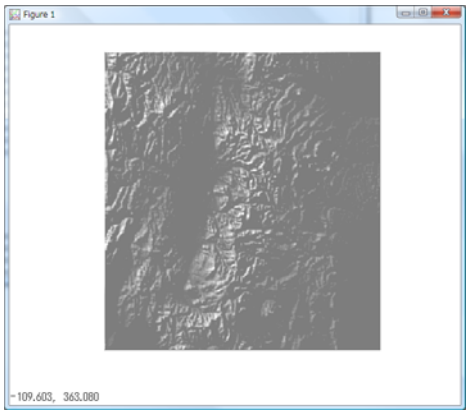


2．カラーマップ(rainbow.m)を用いて，1 を表現しなさい.



左図は rainbow の逆，flipud(rainbow)を使用

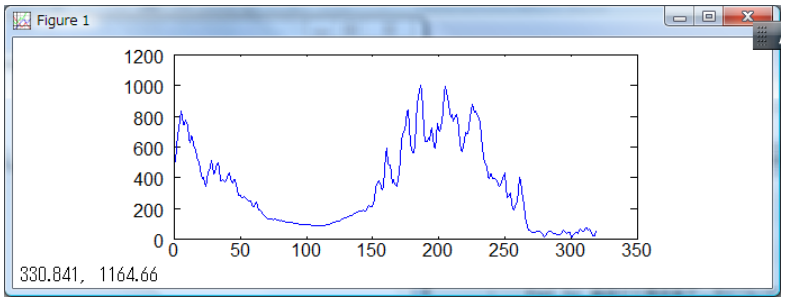
3．下図に示すような擬似レリーフマップを作成しなさい.



B.地形断面図の表現

１．任意の格子上の東西方向の地形断面を表示しなさい.

画面出力の例：

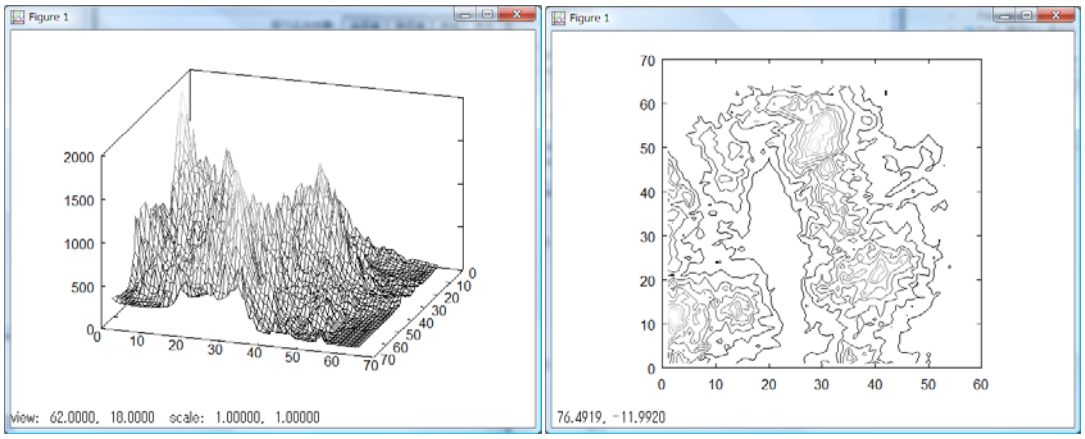


C.地形の３次元表現と等高線表現

１．今回，用いた数値地図には 320×320 の標高データが入っているが，このデータを間引いて，64×64 のデータとしたい．どのように変換すればよいか．（左上端では(5,5)を開始点とするなど必ずしも地図の端辺を含む必要はない.）

２．１．で変換したデータを，mesh コマンドを用いて 3 次元的に表現しなさい.（座標系の違いに注意すること）

３．１．で変換したデータを用いて contour コマンドにより等高線を描画しなさい.



※mesh, contour コマンドの命令の仕様は各自で調べ，効果的に表現できるように工夫すること.

D. 総合問題

１．海水準が 10m 上昇したと仮定した場合の海岸線の位置（0m 以下の地点とする）を示しなさい．表現範囲，表現方法（２次元，３次元）は自由であるが，より工夫を加えることが望ましい

以上を文書ファイル（PDF）に取りまとめ，oyojoho@gmail.com に次回講義前までに提出すること．

※配列の最大値は max(max(配列名))で求めることができる．

※カラーマップは imshow(配列, カラーマップ名)で指定できる．
例：imshow(配列, rainbow)

※３次元グラフの座標系は，axis ij（左上隅に原点をもつ），axis xy（左下隅に原点をもつ）などのコマンドにより指定できる．

※octave は MATLAB 互換ソフトウェアであり，web 上の MATLAB 関連の資料も役立つ場合がある．

※関数ｍファイルの作り方
メモ帳で作成する場合には、ファイルの種類を”すべてのファイル”にするのを忘れずに！ (geomapread.m.txt になってしまうので要注意)

※geomapread を実行した場合に undefined... のエラーが出る場合
⇔source geomapread.m を実行してみる．

※地形データの入手について
入手方法については講義で通知済みですが，データの入手方法がわからない場合は，
makanae@myu.ac.jp
まで連絡をお願いします．

Octave によるプログラミング

1. 分岐構造

(1) if 分岐

```
if (条件式 1)
      実行文 1
elseif （条件式 2）
      実行文 2
・・・
else
      実行文 x
endif
```

※elseif, else は条件によっては省略可能

条件式

a == b	a と b が等しい
a > b	a が b より大きい
a >= b	a は b 以上
a < b	a が b より小さい
a <= b	a は b 以下
a ~= b	a と b が等しくない

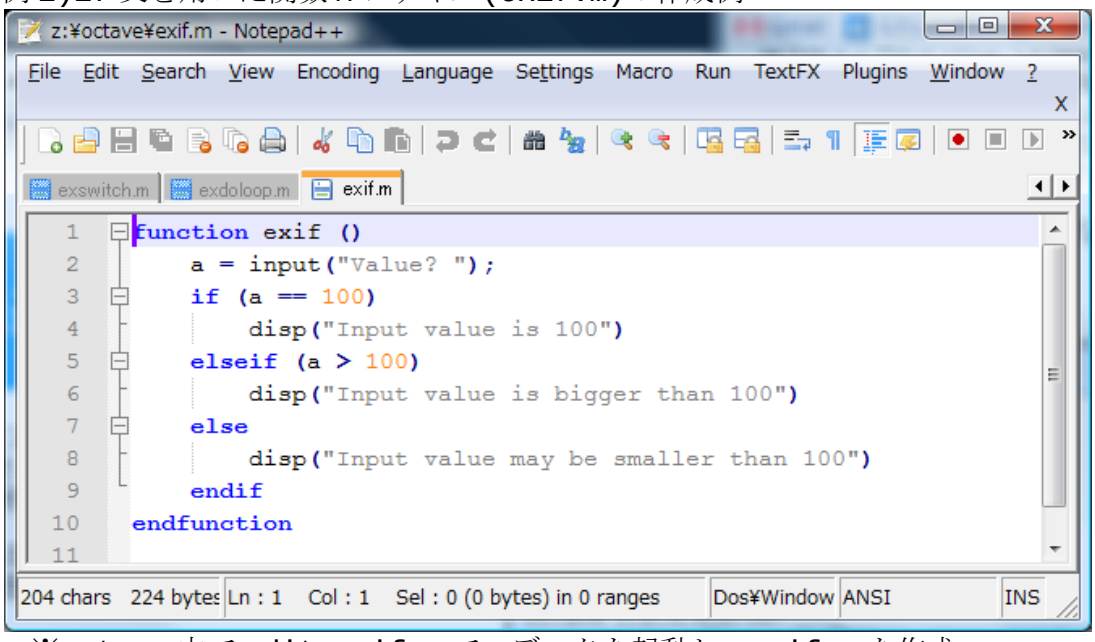
条件式における AND,OR, NOT の記述

AND（かつ）: (条件式 1) && (条件式 2)

OR （または）: (条件式 1) || (条件式 2)

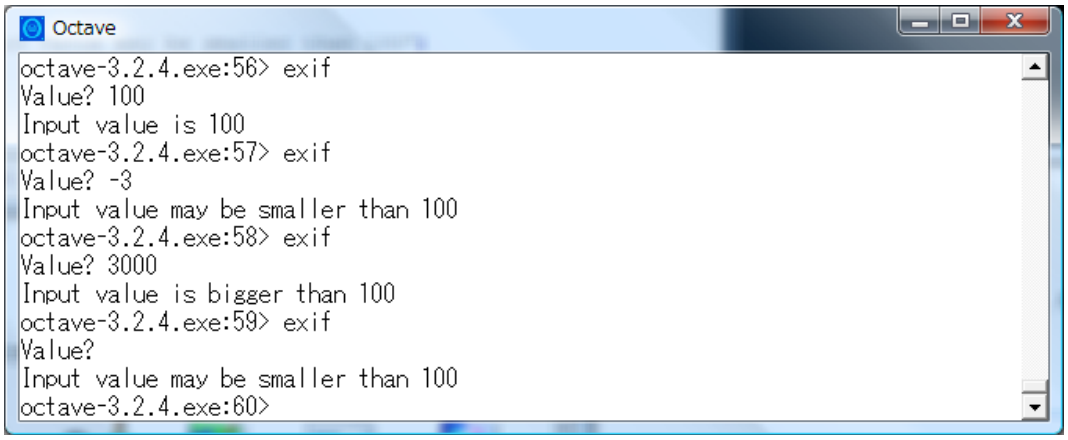
NOT : !（条件式）

例 1)if 文を用いた関数 M ファイル(exif.m)の作成例



※octave 内で edit exif.m でエディタを起動し，exif.m を作成
（edit コマンドでエラーが出る場合には，メモ帳で作成する）

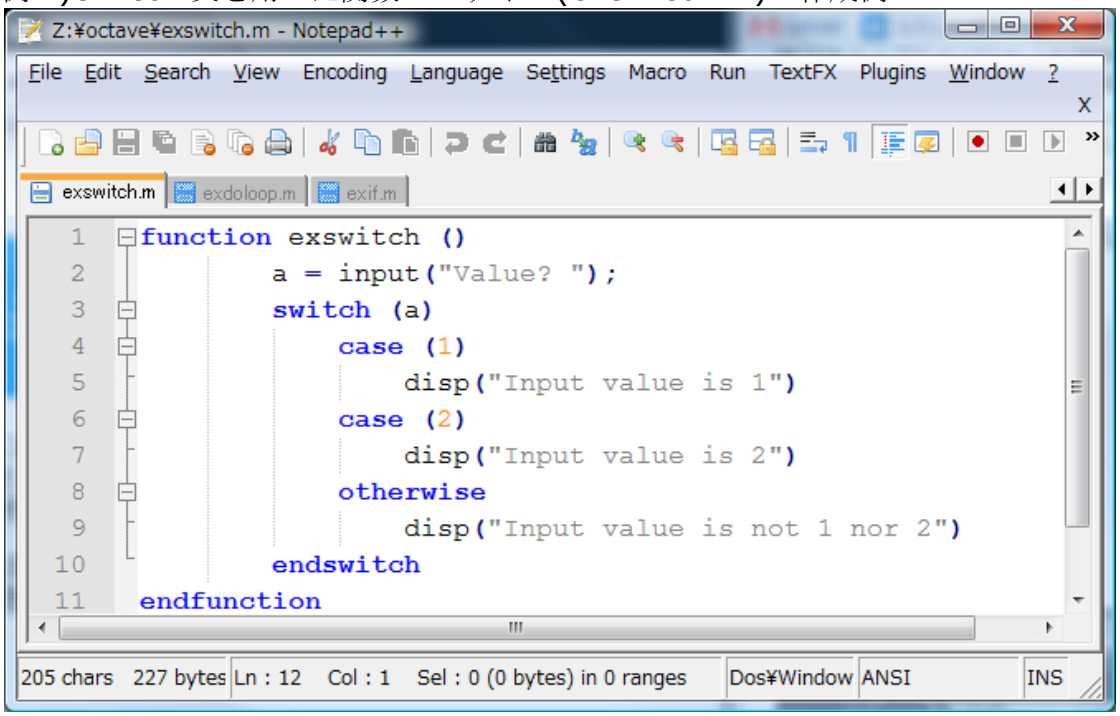
作成した関数 exif.m の実行例



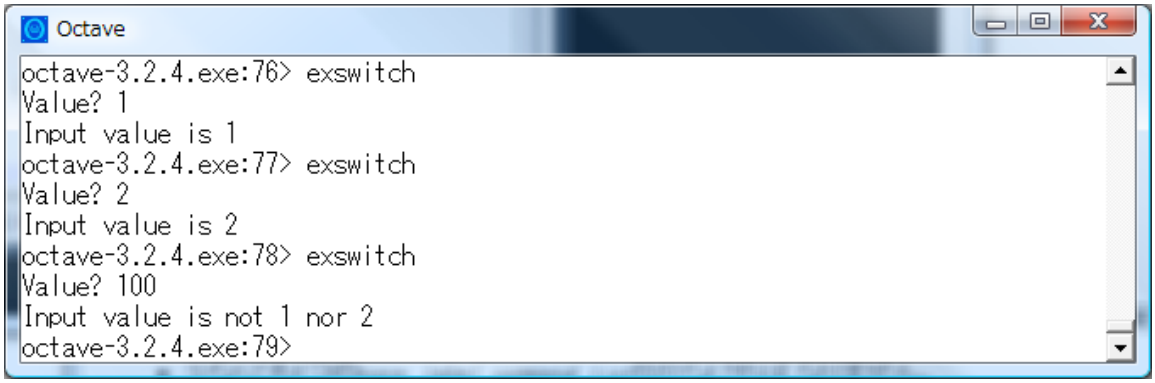
(2) switch 分岐

```
switch 変数名
      case ラベル（式）
            実行文 1
      case ラベル（式）
            実行文 2
        ・・・(略)・・・
      otherwise
            実行文 X
endswitch
```

例 2)switch 文を用いた関数 M ファイル(exswitch.m)の作成例



作成した関数 exswitch.m の実行例

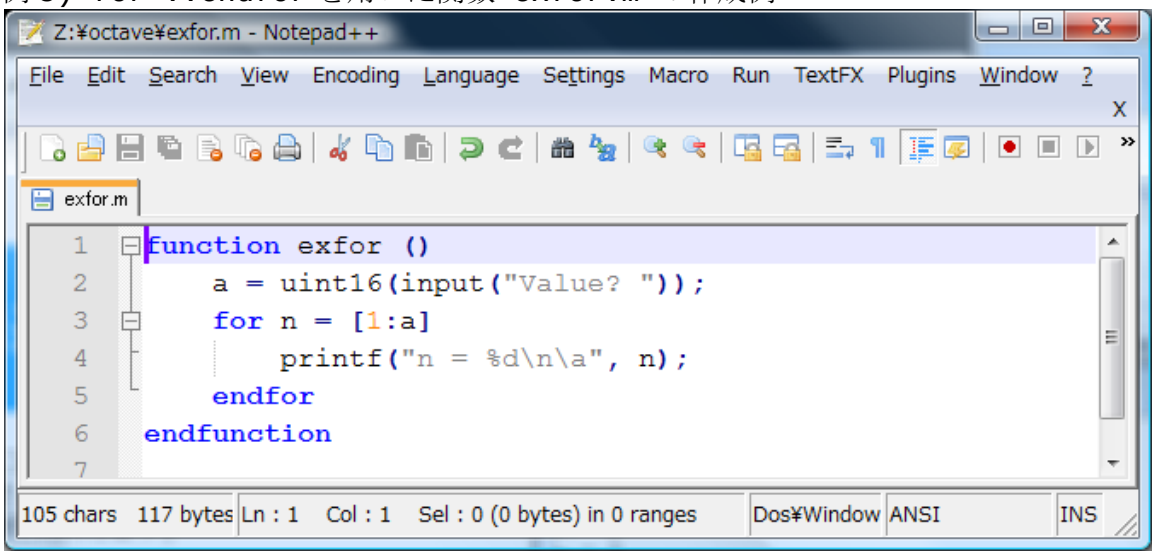


2. 繰り返し構造

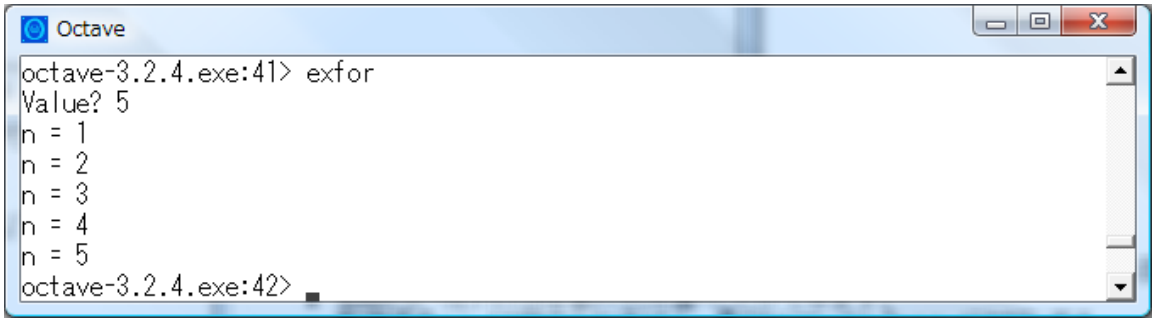
(1) for..endfor による繰り返し

```
for 変数 = [開始:公差:終了]        ※公差を省略した場合は公差１の数列となる
    実行文
endfor
```

例 3) for ..endfor を用いた関数 exfor.m の作成例



作成した関数 exfor.m の実行例

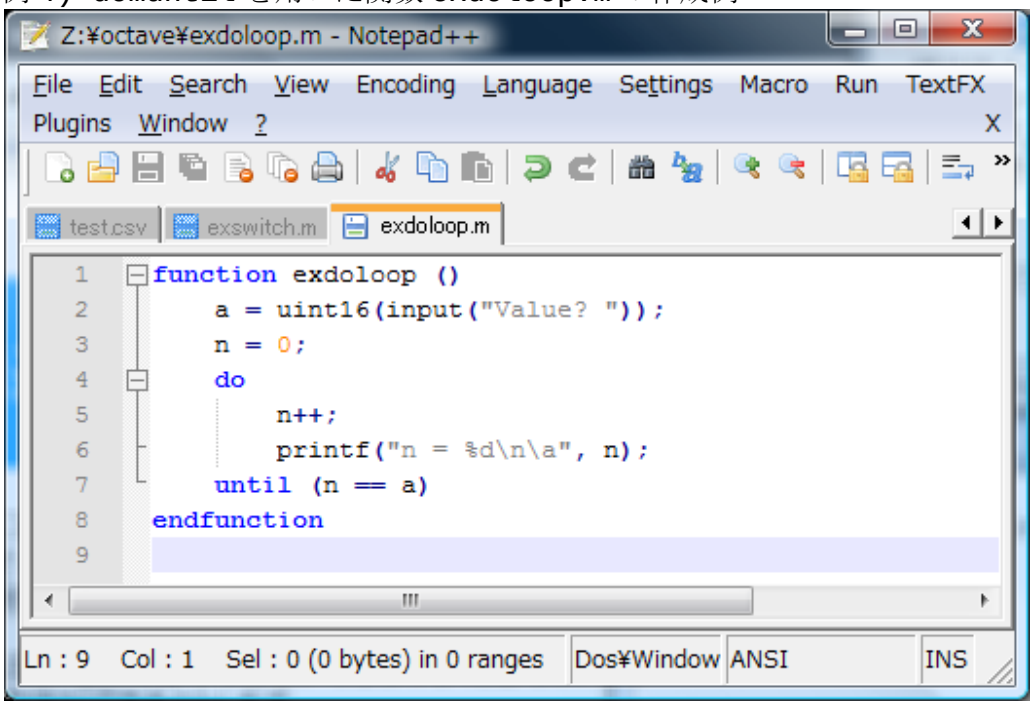


(2) do..until による繰り返し

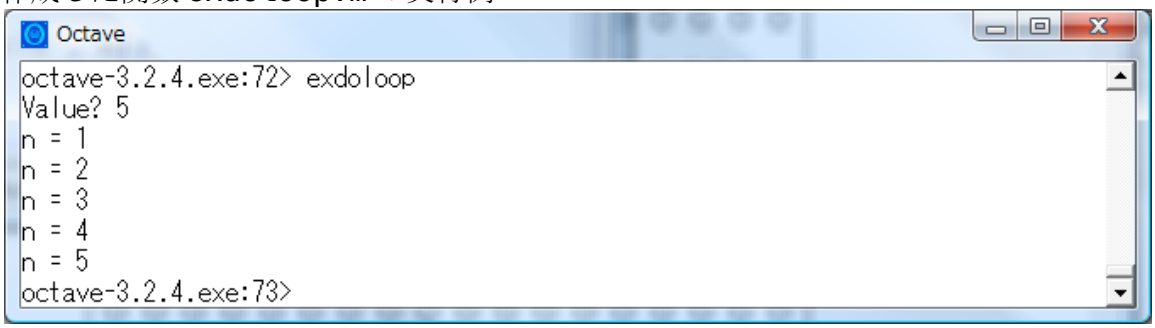
設定した条件式になるまで繰り返す

```
do
      実行文
until (ループ終了の条件式)
```

例 4) do…until を用いた関数 exdoloop.m の作成例



作成した関数 exdoloop.m の実行例

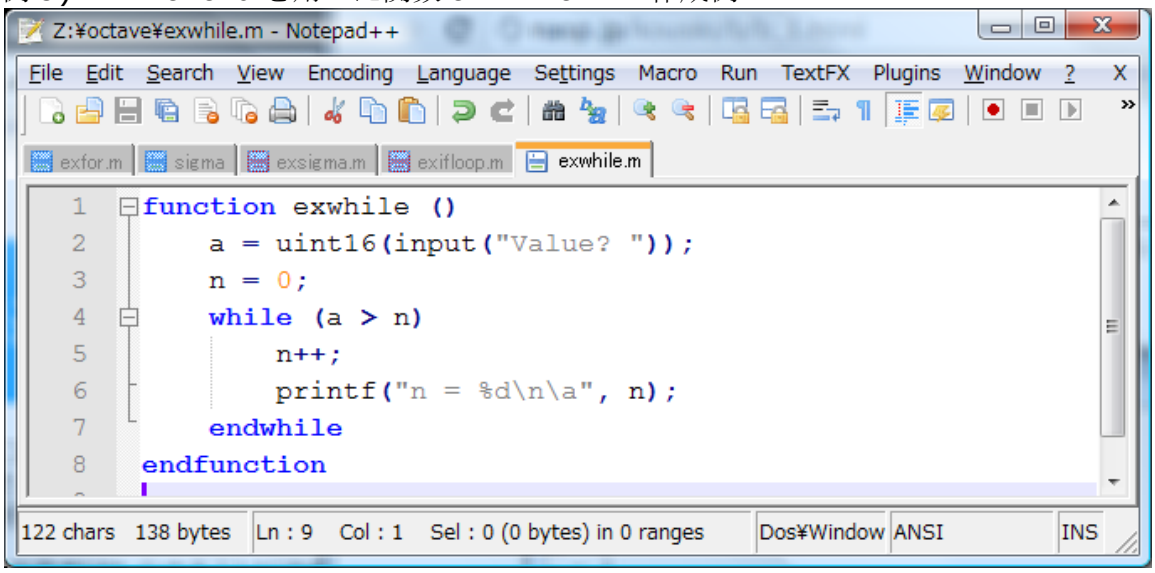


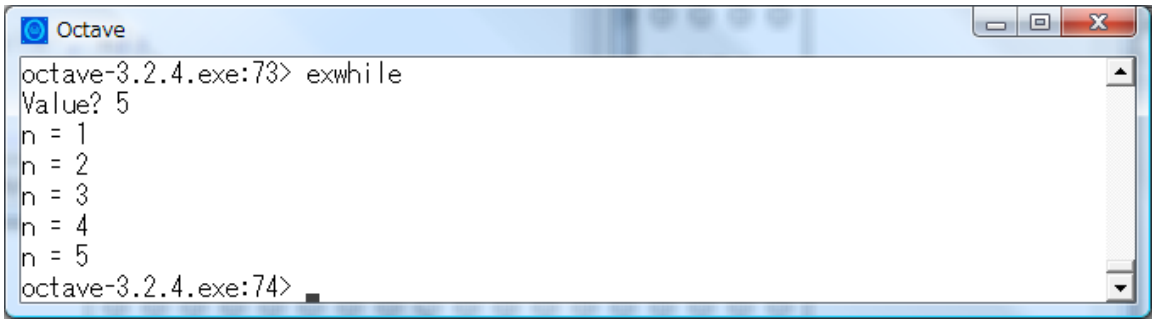
(2) while..endwhile による繰り返し

指定した条件式に合致している間，繰り返す．

```
while (条件式)
      実行文
endwhile
```

例 5) while…end を用いた関数 exwhile.m の作成例





演習課題 1

(1) P.46 例 1 の if 分岐のプログラムについて，終了条件を負の入力値としてループさせるプログラムを作成しなさい．

(2)$\sum_{k=a}^{b} k$ （ただし a,b は正の整数，a ≦ b)を求める関数を作成しなさい．

a,b を引数とし，Σを戻り値とする．

※(1)(2)とも作成した関数ファイルの内容を文書内に貼り付け，提出する．

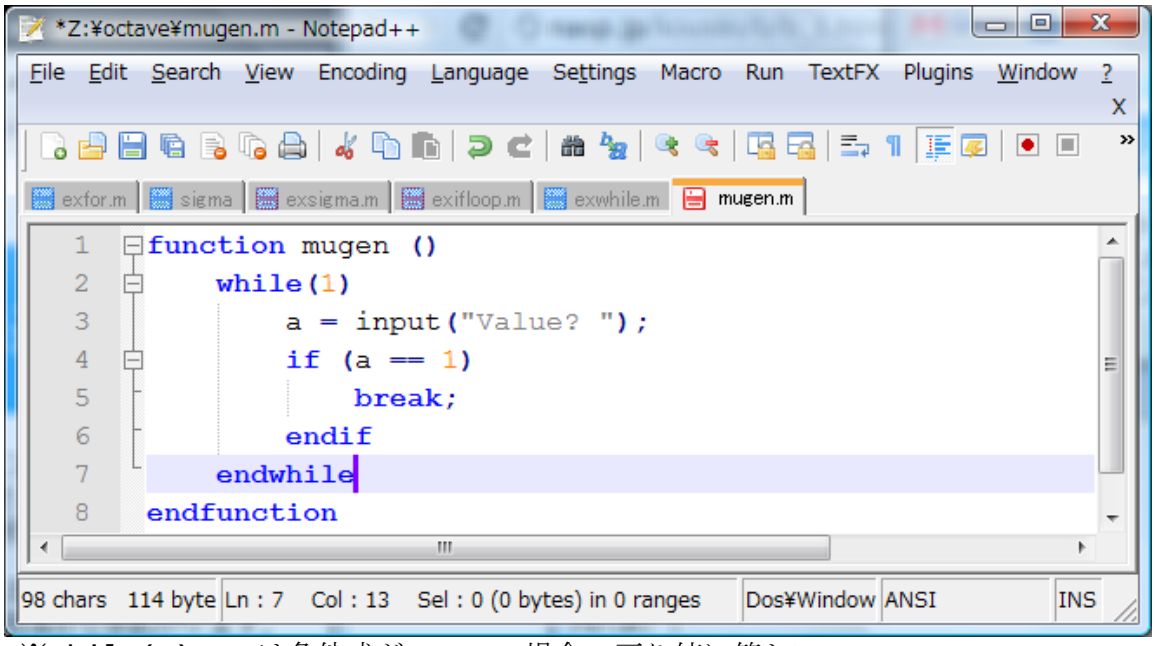
※無限ループの作り方

```
while(1)
   実行文
endfor
```

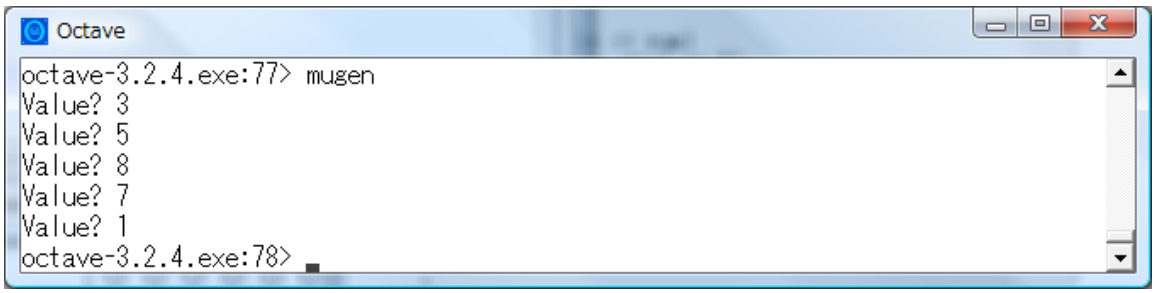
※ループから脱出するためのコマンド

```
break
```

break を使ったプログラム例



※while(1)の 1 は条件式が TRUE の場合の戻り値に等しい．



擬似乱数とモンテカルロシミュレーション(1)

1. モンテカルロ法 (Monte Carl method)とは

モンテカルロ法とは，乱数を用いたシミュレーションにより近似解を求める計算手法を言う．第 2 次世界大戦中に，米国ロスアラモス国立研究所で，John von Neumann（コンピュータの父）により，物質中を動き回る中性子の様子をシミュレートするために考案された．モンテカルロの名はギャンブルの国モナコ公国の首都名に由来する．

2. 擬似乱数

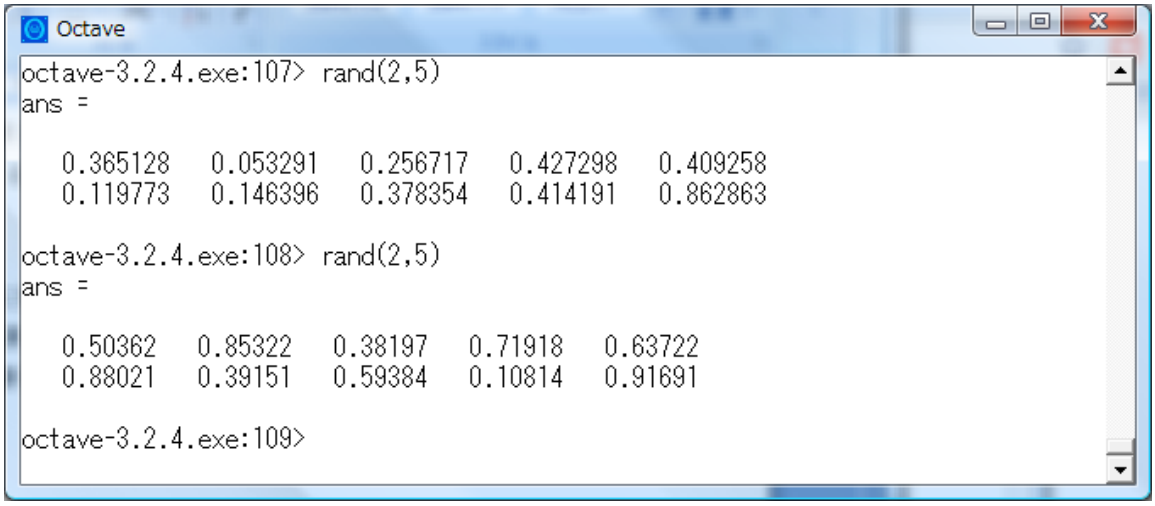
乱数列とは，ランダム（でたらめ）に発生した数の系列であり，無秩序かつ全体として出現する頻度が等しい数列である．乱数列の要素を乱数と呼ぶ．

コンピュータ上で厳密な意味での乱数を求めることは難しく，実際には計算式により求められる『擬似乱数』を用いる．

3. Octave 上での擬似乱数の発生

(1)一様乱数の発生

一様乱数を発生する関数　rand
　rand(x)　：　乱数の入った x の正方行列を返す
　rand(n, m)　：乱数の入った　n 行 m 列の行列を返す



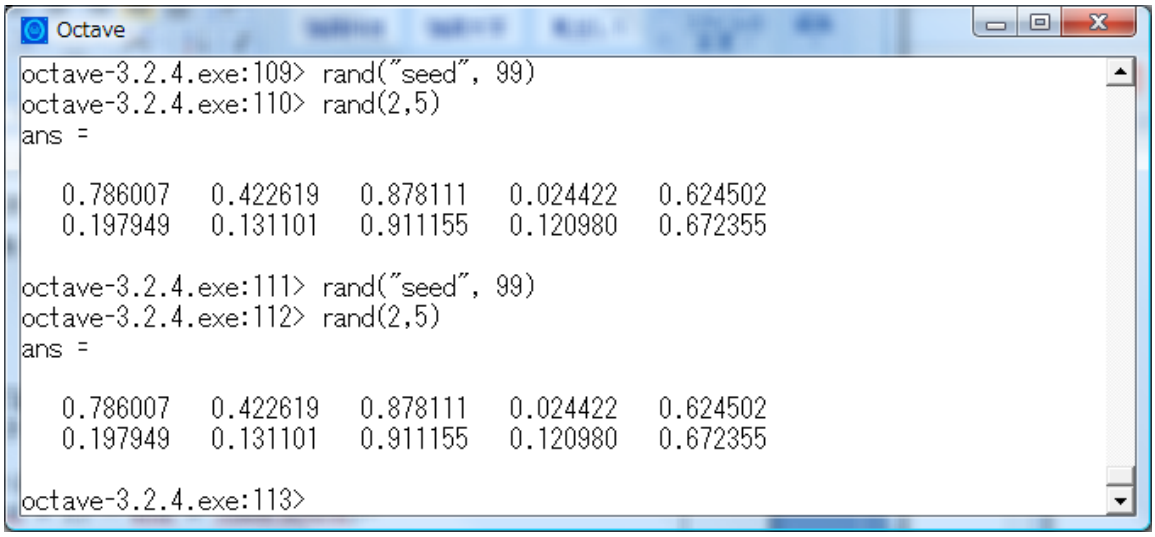
(2) seed 値の設定について

rand 関数は初期シードをシステムクロックより得ることにより，通常は毎回，呼び出される毎に異なる系列の乱数を発生させる．

同じ乱数系列を発生させたい場合には，任意の数値を seed として設定する．

```
rand("seed", x)      ※x はスカラ値
```

※seed の設定例：同じ乱数系列が発生する．

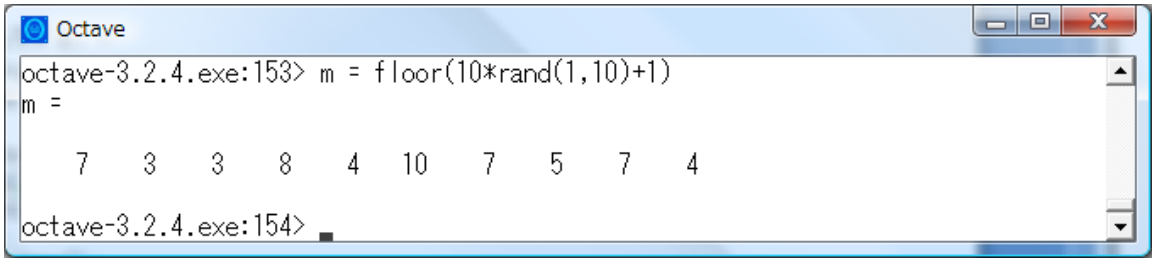


(3)任意の範囲の乱数を発生させる．

```
floor((上限値−下限値+ 1 ) * rand(m,n) + 下限値)
```

※floor() は小数点以下を切り捨てる関数

例 1） １～１０の整数の乱数を要素とする１行１０列の行列の生成

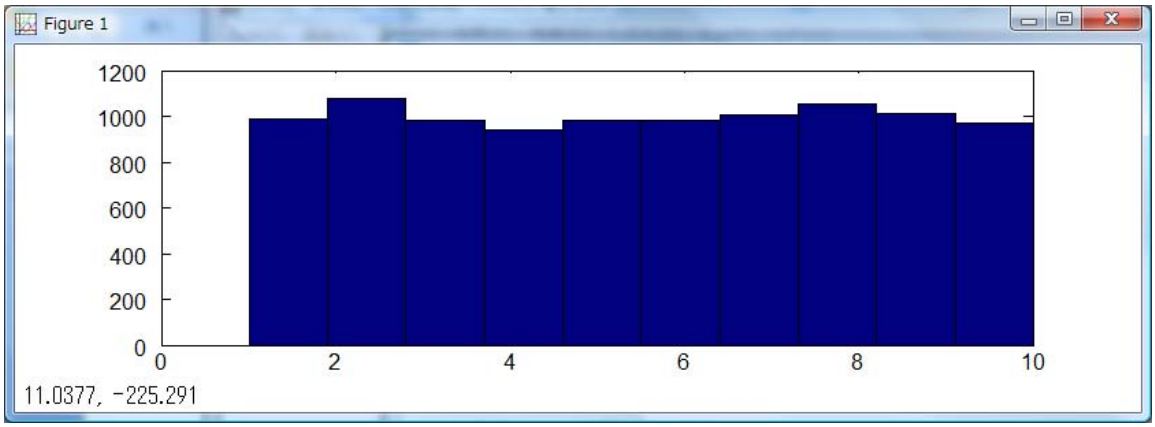


演習課題２
(1)例 1 について，要素数を増加させて rand 関数による乱数の一様性を確認しなさい．（各要素数に対するヒストグラムを作成する）

(2)1 から 6 の目をもつサイコロと同じ範囲の乱数行列（1 行×100 列）を発生させなさい．

※(1)はヒストグラム ，(2)はコードを文書内に貼り付け提出する．
※hist(行列名)でヒストグラムを描画できる．

ヒストグラムの例



例 2)トランプの例

```
function [ ret ] = trump ()
     flag=zeros(1,52);
     n = 0;
     do
          pickedCard = floor(52*rand()+1);
          if (flag(1, pickedCard) == 0)
               n++;
               input("Pick! (Press Enter-key!)");
               flag(1, pickedCard) = n;
               switch(floor((pickedCard-1)/13))
                    case(0)
                         mark = "Spades";
                    case(1)
                         mark = "Clubs";
                    case(2)
                         mark = "Diamonds";
                    case(3)
                         mark = "Hearts";
               endswitch
               number = mod(pickedCard-1, 13) + 1;
               printf("Picked card is %s %d\n", mark, number);
          endif
     until (n == 52)
     disp ("Over");
     input("Pause");
     ret = flag;
endfunction
```

演習課題３
(1)例 2 のトランプのプログラムにジョーカー（１枚）を加えなさい.
(2)一様乱数を用いたゲームまたはシミュレーションを作成しなさい.

※(1)についてはコードを文書内に貼り付ける.
(2)についてはコード，実行画面を文書内に貼り付けるとともに，
作成したプログラムに関する説明を加える．あわせて関数 M ファイルを添付すること.

演習課題の提出について

演習課題１～３をまとめて，PDF ファイルとして，
次週の講義前までに oyojoho@gmail.com に提出すること.

モンテカルロシミュレーション(2)

1. モンテカルロ法による近似解の導出(πの値を求める)

座標(0,0)-(1,1)の正方形の範囲で，乱数により求めた座標をもとに N 個の点を一様にばらまく．それらの点のうち，原点(0,0)から距離が 1 以内のものの数を a とする．対象範囲の正方形の面積と 1/4 円との面積比と，均等にばらまいた点の数と a との比率は等しいと考えられることから，次の式が成立する．

$$1 : \frac{\pi}{4} = N : a$$

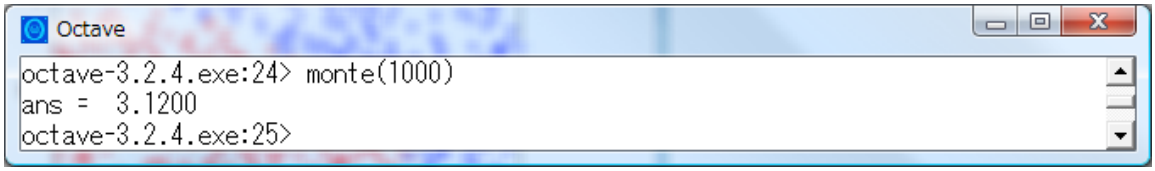
$$\therefore \pi = \frac{4a}{N}$$

この関係を用いて円周率の近似値を求めることができる．

例 1)モンテカルロ法による円周率の近似値の算出　monte.m

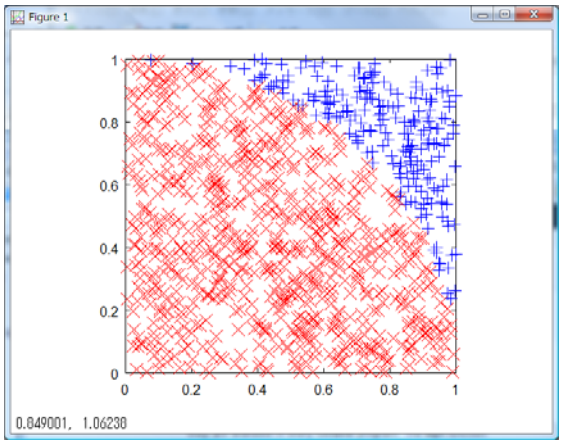
```
function ret = monte(N)
     a = 0;
     clf;              #plot 画面をクリアする
     axis([0,1,0,1], "square");  #X 軸を 0~1, Y 軸を 0~1 の範囲に
     for k = [1:N]
          x = rand();
          y = rand();
          hold on;                #plot 上の描画を維持（消去しない）
          if (sqrt(x*x+y*y) <= 1)
               plot(x, y, 'xr');    #x 印, r(赤)でプロット
               a++;
          else
               plot(x, y, '+b');     #+印, b(青)でプロット
          endif
     endfor
     ret = 4 * a / N;
endfunction
```

実行例）引数はばらまく点の数 N，戻り値は求められたπの値である．



引数は点の数 N，戻り値は求められたπの値である．

実行画面の例)



演習課題 1

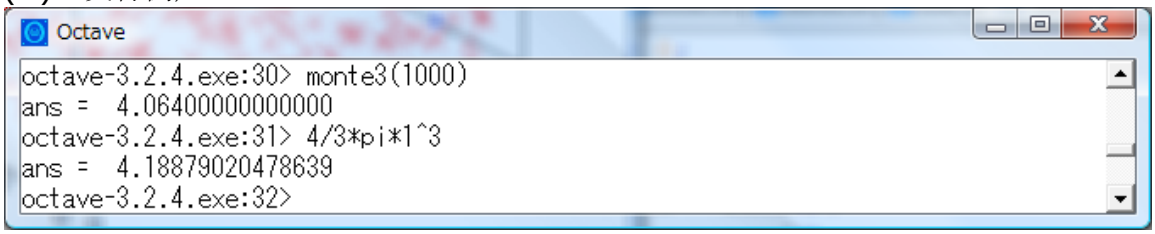
(1)例 1 のプログラムをもとに N を 50 から 3000 まで 100 刻みで増やしたときに求められるπの値の変化を示すグラフを作成するプログラムを作成しなさい.
(※例 1 の if 文内にある plot 文は削除すること)

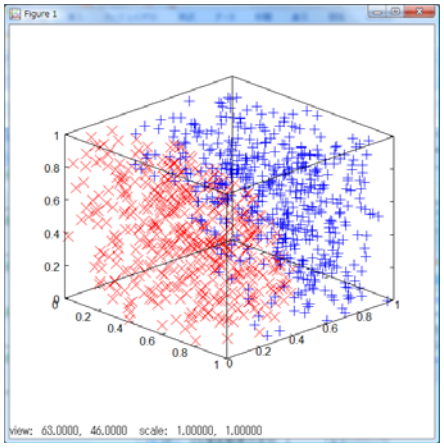
(2)モンテカルロ法を用いて半径 1 の球の体積を求めるプログラムを作成しなさい.

※a を浮動小数点付数として出力する場合 : printf("%f",a)

※一時的に処理を止める(plot に一時的に処理を戻すときに有用) usleep(1)
(plot で描画を行う場合は drawnow でも良い)

(2)の実行例)



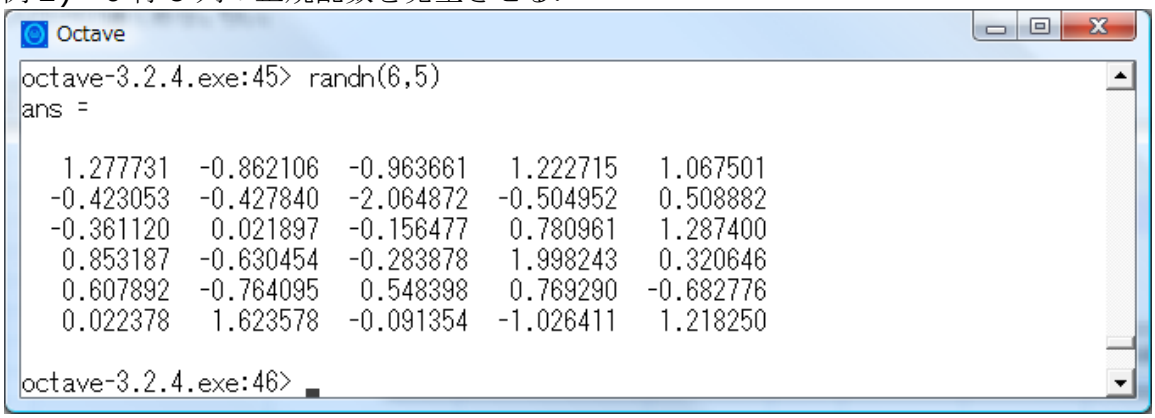


2. 正規乱数とポアソン乱数

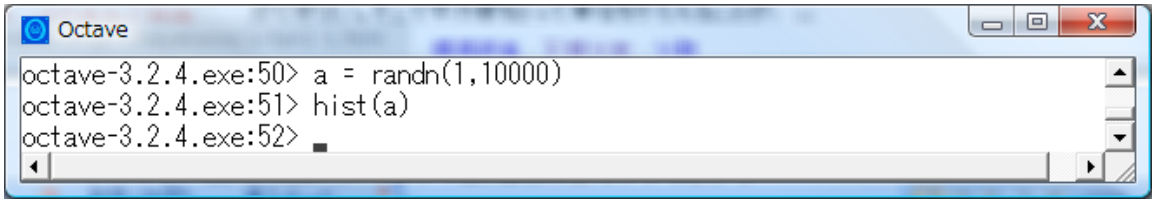
(1)正規乱数

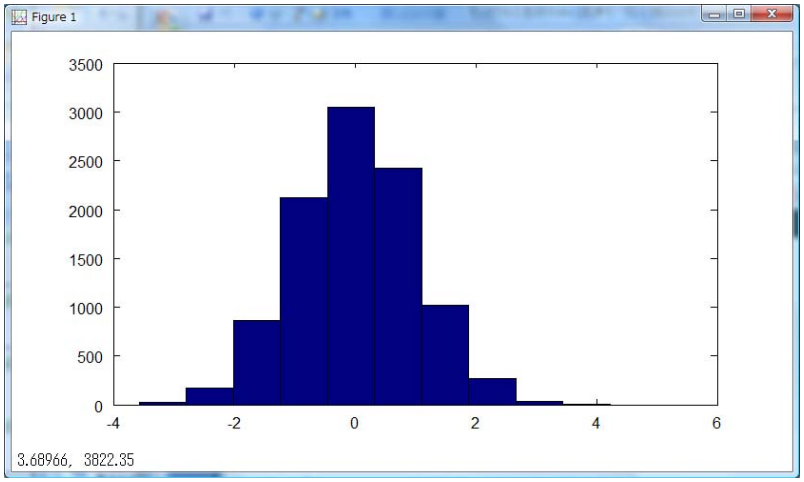
正規乱数(平均 μ =0, 分散 σ^2=1)を発生する関数　rand
　randn(x) ：　乱数の入った x の正方行列を返す
　randn(n, m)　：乱数の入った n 行 m 列の行列を返す
正規乱数における seed 値の設定
　randn(“seed”, x)　　※x はスカラ値

例 1)　6 行 5 列の正規乱数を発生させる.



例 2）10000 個の正規乱数を発生させ，ヒストグラムを作成する



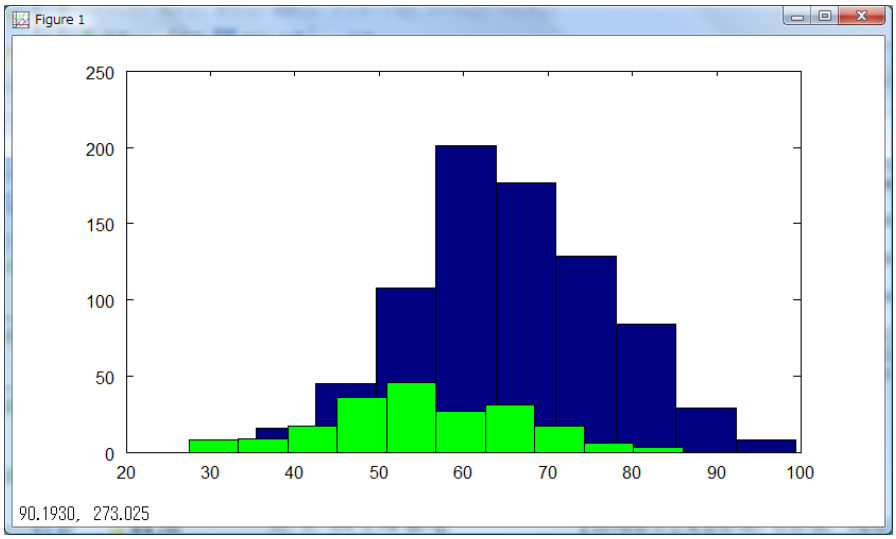


演習課題２

交通量が少ない状態での自動車の自由速度の分布は正規分布に近似することが知られている．以下の条件をもとに擬似的に発生させた乗用車，トラックの速度分布を示すヒストグラムを重ね合わせて表現しなさい．（コードとグラフを提出する）
①乗用車の平均速度は 65km/h，トラックの平均速度は 55km/h とする．
②速度の変動係数（標準偏差/平均速度）は乗用車・トラックとも 0.18 とする．
③発生させる自動車の全車両台数は 1000 台とし，トラックの混入率は 20%とする．

※ヒストグラムの色を変える

```
hist(変数名, 'FaceColor', 'b')      # 'b'は青，'g'は緑，'r'は赤
```



(2)ポアソン確率密度関数

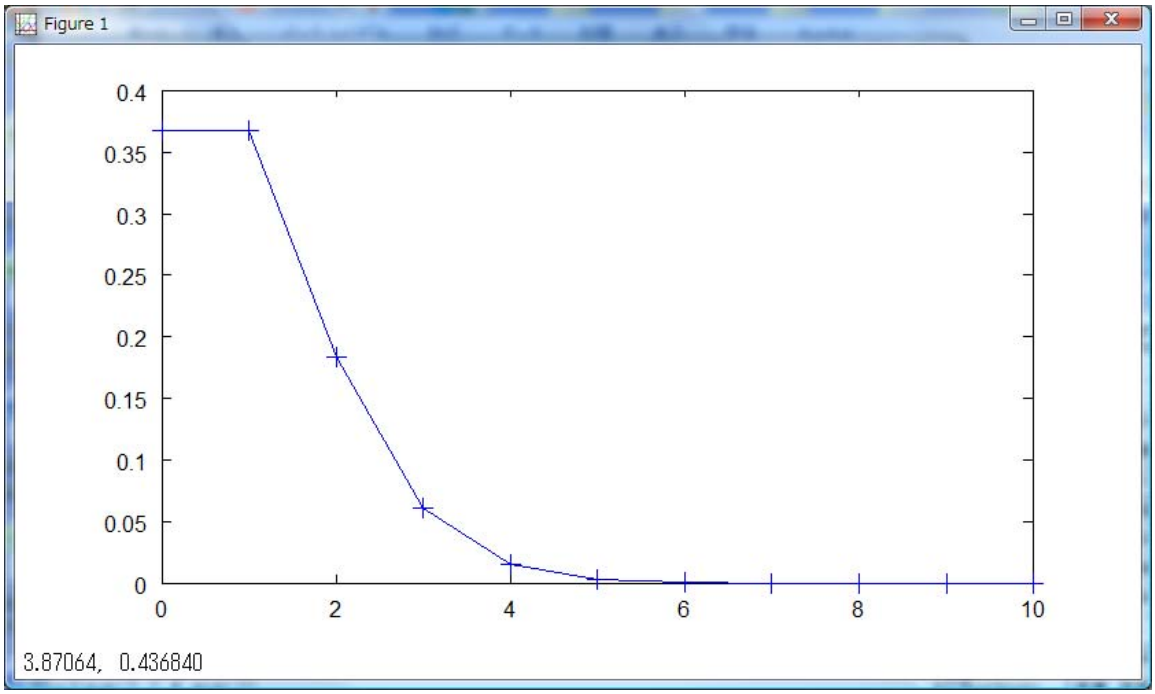
ポアソン確率密度関数

```
Y = poisspdf(X, lambda)
```

※まれに起こる事象はポアソン分布で表わすことができ，ポアソン確率密度関数は
$P(X = x) = \frac{\lambda^x}{x!}e^{-\lambda}$ で表される．λはポアソン分布の平均である．

例１）λ=1 のときの確率密度関数を求める．

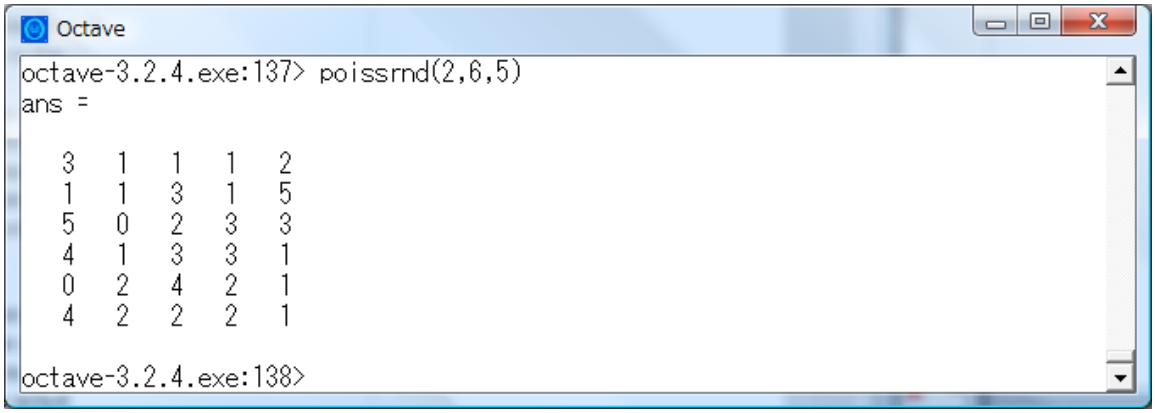
```
function [ ret ] = poisson1 ()
      lambda = 1
      x = [0:10];
      y = poisspdf(x, lambda);
      plot(x, y, '-@');
endfunction
```



(3)ポアソン乱数

ポアソン分布に基づく乱数（ポアソン乱数）を発生させる関数
poissrnd(lambda) ：ポアソン乱数を返す
poissrnd(lambda, x) ：乱数の入った x の正方行列を返す
poissrnd(lamada, n, m) ：乱数の入った n 行 m 列の行列を返す

例１） λ=2 のときの 6 行 5 列のポアソン乱数を発生させる.



演習課題３
(1)例 1)をもとに，λ＝1,2, ・・・, 10（整数）の場合の確率分布関数を重ねてグラフ上に示しなさい.

(2)交通量が少ない場合の車両の到着（発生）はポアソン分布で近似できることが知られている．ある料金所に 1 分間に平均 2 台の車両が到着するときに，60 分間の 1 分毎の車両の到着台数をシミュレートしなさい．到着台数を示すグラフ及びコードを示しなさい.

(3)(2)の料金所において 1 分間に 2 台の処理が可能であるとする．(2)と同じ 60 分間の 1 分毎の待ち台数の数を(2)と同じグラフ上に示すプログラムを作成しなさい．作成したグラフとコードを提出すること.

演習１，２，３をとりまとめ，PDF ファイルとして oyojoho@gmail.com に次回講義前までに提出すること.

※6 月 25 日（月）は休講です．次回講義は 7 月 2 日（月）になります.

※poisspdf, poissrnd が動作しない（見つからない）場合
http://www.makalab.org/ojb/から
poisspdf.m, poissrnd.m
をダウンロードして実行用のディレクトリ（フォルダ）に保存しておく.

経路探索アルゴリズム

1. 再帰的アルゴリズム

プログラム処理において，ある手続きの中でその手続き自らを呼び出す手続きを有する構造（再帰; recursive）をもつアルゴリズムを再帰アルゴリズムという．

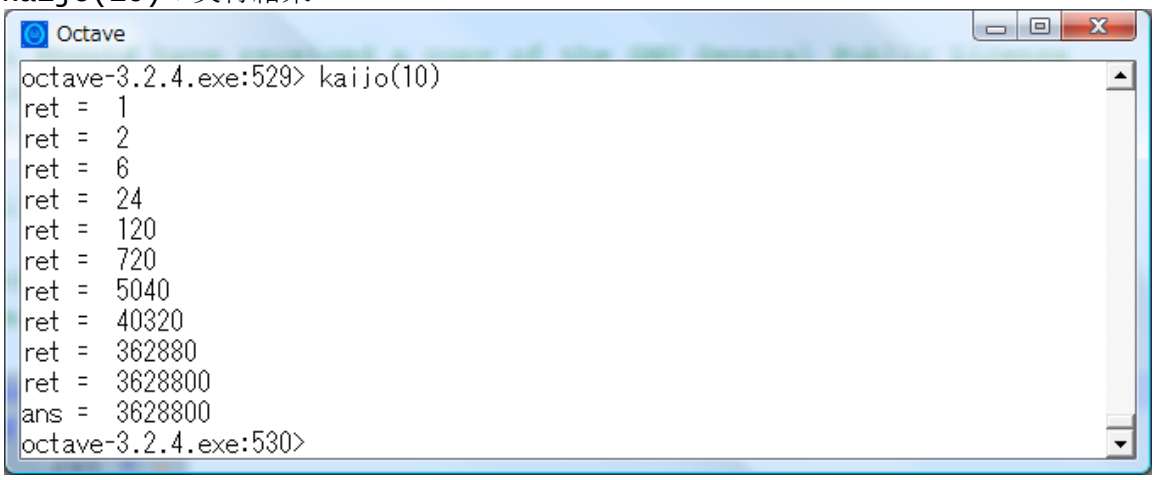
(1)再帰アルゴリズムを用いて階乗を求める

$$\begin{cases} n! = n \cdot (n-1)! & n > 0 \\ 0! = 1 \end{cases}$$

関数 kaijo.m

```
function [ ret ] = kaijo (m)
     if (m == 1)
          ret = 1
     else
          ret = m * kaijo(m-1)
     endif
endfunction
```

kaijo(10)の実行結果



2. 迷路を解く

任意の大きさの格子を設定し，1（通行可），0（通行不可）により迷路を設定する．

1	1	1	0	0
0	0	1	0	1
1	1	1	1	1
0	0	0	0	1
1	1	1	1	1

この迷路を再帰アルゴリズムを用いて解く．

a)迷路の設定と routefinder.m を呼び出すプログラム maze.m

```
function [ ret ] = maze ()
     more off;
     global mazegrid;
     global goalflag;
        mazegrid = [1 1 1 0 0;
                    0 0 1 0 1;
                    1 1 1 1 1;
                    0 0 0 0 1;
                    1 1 1 1 1]
     goalflag = false;
     input("Start Routefinder (press Enter)")
     routefinder(1,1,5,1)   # routefinder(r,c,r,c)
     mazegrid
     disp("Over");
     more on;
endfunction
```

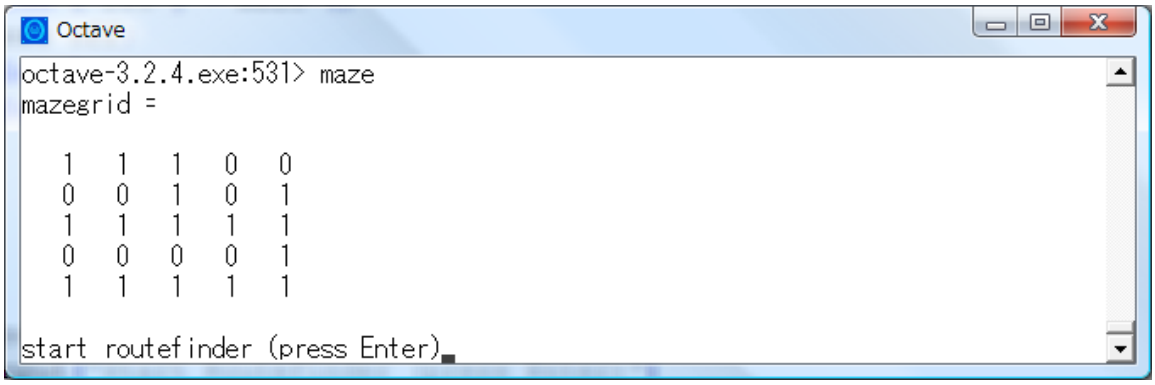
b)再帰的アルゴリズムにより迷路を探索する routefinder.m

```
function [ ret ] = routefinder(m, n, gm, gn)
     global mazegrid;
     global goalflag;
     if (mazegrid(m,n) == 1)
          mazegrid(m,n) = 2;  #"checked point is marked by "2"
          if (m == gm && n == gn)
               disp("goal");
               goalflag = true;
          else
               if (m > 1 && goalflag == false)
                    routefinder(m-1, n, gm, gn)
               endif
               if (n > 1 && goalflag == false)
                    routefinder(m, n-1, gm, gn)
               endif
               if ( m < size(mazegrid,1)&& goalflag == false)
                    routefinder(m+1, n, gm, gn)
```

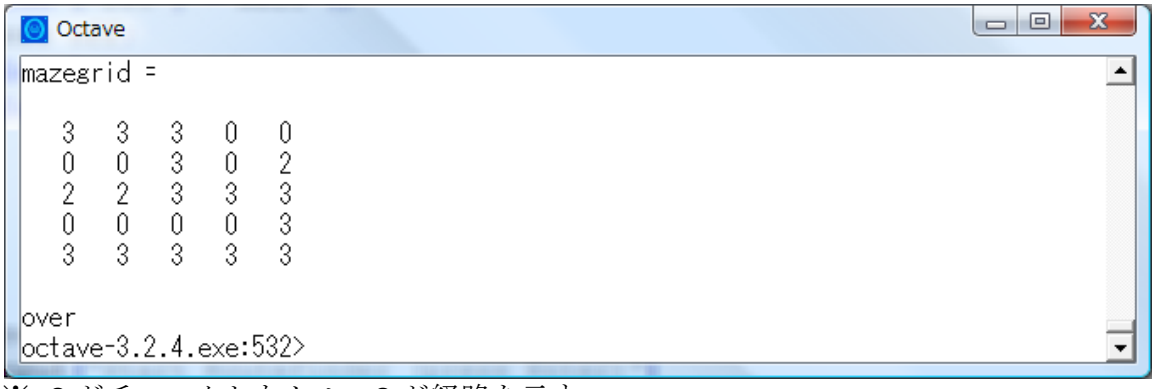
```
                endif
                if (n < size(mazegrid,2)&& goalflag == false)
                     routefinder(m, n+1, gm, gn)
                endif
           endif
           if (goalflag == true)
                mazegrid(m,n) = 3;  # show a route by "3"
                disp("Goal!")
           endif
     endif
     mazegrid
     input("pause  (escape: ctrl+C) ");
endfunction
```

※global 変数について
　複数の関数間で変数を共有したい場合に global 変数を用いる.
　　global 変数名
※more on/ off 出力ページを区切るか否かを設定する
※プログラムの停止は　ctrl+C

実行画面（開始時）



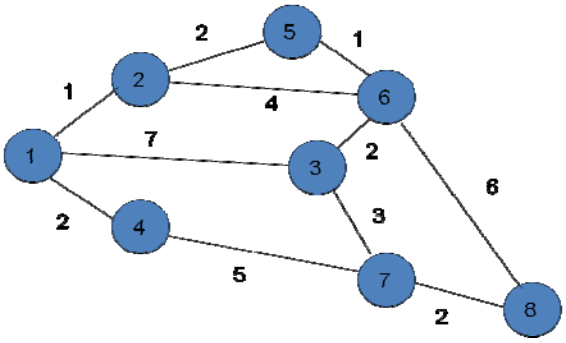
実行画面（終了時）



※ 2 がチェックしたセル，3 が経路を示す.

3. 最短経路を求める（ダイクストラ法）

各辺に重みが定義されたネットワーク（グラフ）上において，ある起点から各点への最短距離を求める方法である．



隣接行列による表現

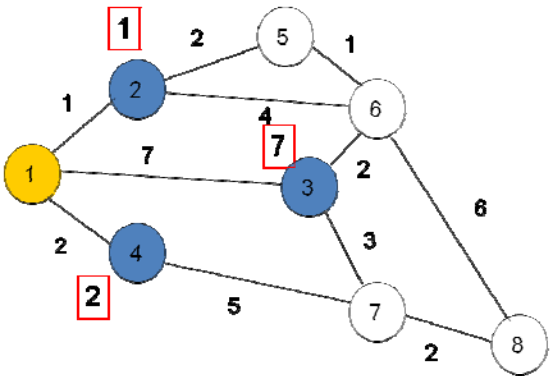
	1	2	3	4	5	6	7	8
1	0	1	7	2	∞	∞	∞	∞
2	1	0	∞	∞	2	4	∞	∞
3	7	∞	0	∞	∞	2	3	∞
4	2	∞	∞	0	∞	∞	5	∞
5	∞	2	∞	∞	0	1	∞	∞
6	∞	4	2	∞	1	0	∞	6
7	∞	∞	3	5	∞	∞	0	2
8	∞	∞	∞	∞	∞	6	2	0

∞は大きな値

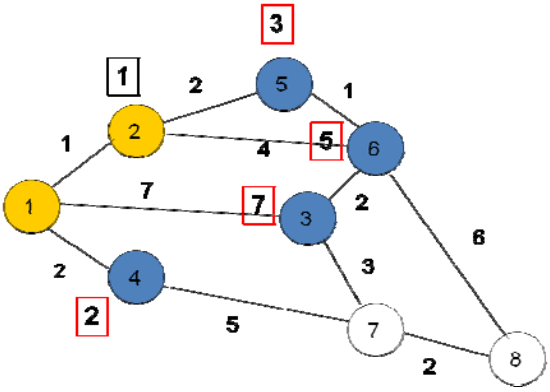
手順１：起点を確定点として設定する（この場合には①）
手順２：起点から，それぞれの確定点を経由して未確定点に達する距離を求め，その中で最短のものを選ぶ．その時の未確定点を確定点とする．
手順３：起点から各点までの距離のデータを最新の情報で更新する．
手順４：手順２に戻る

上記の手順を繰り返すことにより，起点から各点への最短距離，最短経路を求めることができる．

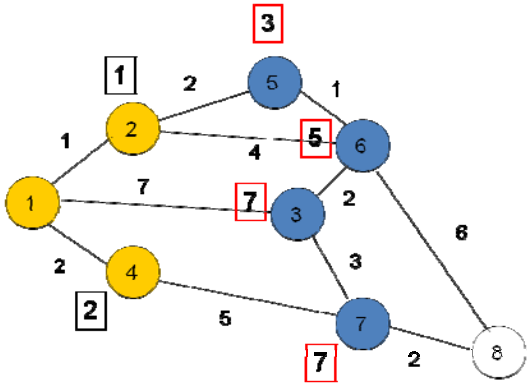
(2)起点①からの距離を求める



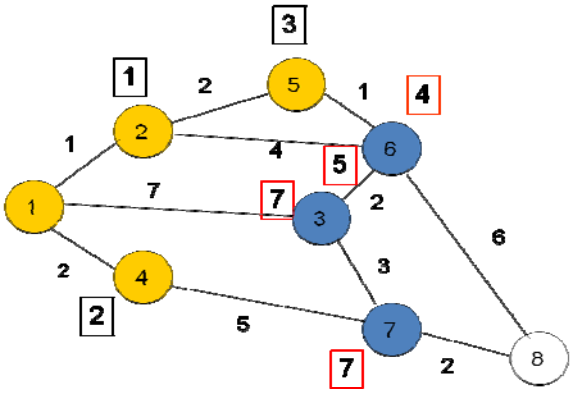
(3)最短のものとして②を確定
確定点から未確定点への距離を求める



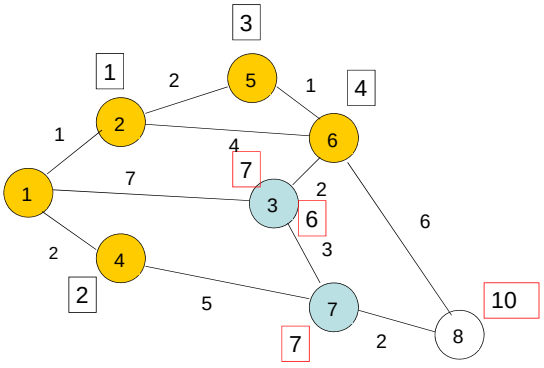
(4)最短のものとして④を確定
確定点から未確定点への距離を求める



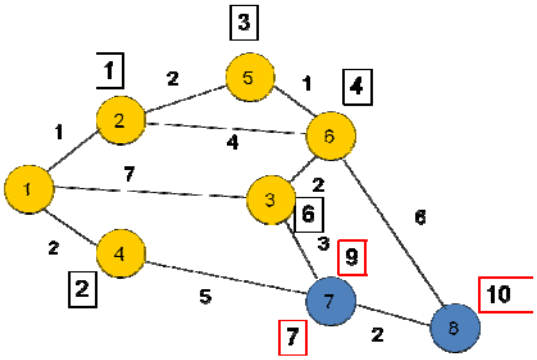
(5)最短のものとして⑤を確定
確定点から未確定点への距離を求める



(6)最短のものとして⑥を確定
確定点から未確定点への距離を求める

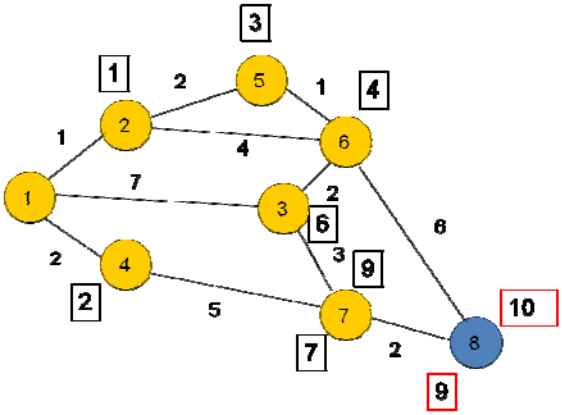


(7)最短のものとして③を確定
確定点から未確定点への距離を求める

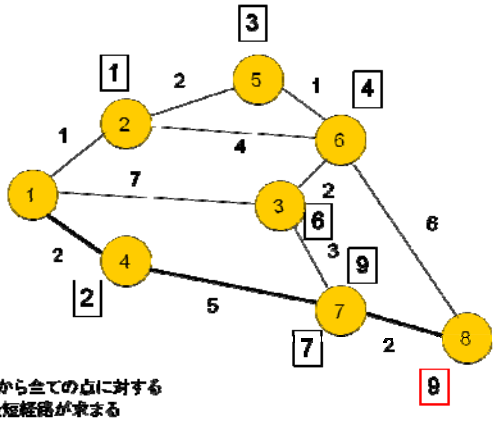


①～⑥の距離は 6 で更新される.

(8)最短のものとして⑦を確定
確定点から未確定点への距離を求める



(9)最短のものとして⑧を確定
①～⑧の距離は 9 で更新



a)隣接行列の設定と経路探索の呼び出し　calldijk.m

```
#set Adjacency Matrix
A = [0 1 7 2 99 99 99 99
     1 0 99 99 2 4 99 99
     7 99 0 99 99 2 3 99
     2 99 99 0 99 99 5 99
     99 2 99 99 0 1 99 99
     99 4 2 99 1 0 99 6
     99 99 3 5 99 99 0 2
     99 99 99 99 99 6 2 0]

#call dijk(Matrix, Origin, Destination)
dijk(A, 1, 1)
dijk(A, 1, 2)
dijk(A, 1, 3)
dijk(A, 1, 4)
dijk(A, 1, 5)
dijk(A, 1, 6)
dijk(A, 1, 7)
dijk(A, 1, 8)
```

b)経路探索 dijk.m

dijk(行列名，起点番号，終点番号)

戻り値は [距離 終点番号 経由点番号・・・ 起点番号]

```
function [ ret ] = dijk (A, origin_p, destination_p)
     Max = 9999;          #Maximum (as Infinity)
     N = size(A, 1);      #Number of Points
     flag = zeros(1,N);  #flag of the confirmed point
     dist = zeros(1,N);  #distance from the start point
     index = zeros(1,N);  #Index of route connection
     for k = [1:N]
          dist(1, k)=Max;
     endfor
     dist(origin_p) = 0;
     for j = [1:N]
          min = Max;
          for k = [1:N]
               if (flag(1,k)==false && dist(1,k) < min)
                    p = k;
                    min = dist(1,k);
               endif
          endfor
          if (min == Max)
               disp("Network is not completed");
               return;
          else
               flag(1,p)=k;
          endif
          for k = [1:N]
               if (dist(1,p)+A(p, k) < dist(1,k))
                    dist(1,k) = dist(1,p) + A(p, k);
                    index(k)=p;
               endif
```

```
            endfor
      endfor
      ret = dist(1, destination_p);

      p = destination_p;
      ret = [ret p];
      while( p ~= origin_p)
            p = index(p);
            ret = [ret  p];
      endwhile
endfunction
```

演習課題

1．フィボナッチ数列を求める再帰アルゴリズムによる関数を作成しなさい．
また，その関数を用いて n=1,2,3,・・・,25 までの値を出力するスクリプトを作成しなさい．

2．任意の迷路および起終点を設定し，設定した起終点での経路を求めなさい．
（独自のルール・工夫を設けても良い）

3．東北地方（周辺都道府県を含む）の 10 都市以上を対象としてネットワークを設定し，実距離（道路あるいは鉄道）および時間距離による経路を探索するプログラムを作成しなさい．
(設定したネットワーク図（MS-Word 等の図でよい)，プログラム，経路探索の実行画面を提出すること．都市名は数字に置き換えてよい（ネットワーク図中に示すこと))

上記を PDF ファイルにまとめて次回講義前までに oyojoho@gmail.com に提出すること．

※フィボナッチ(Fibonacci)数列は下式により表される．

$$\begin{cases} f_n = f_{n-1} + f_{n-2} & n \geq 3 \text{ のとき} \\ f_1 = f_2 = 1 & n = 1,2 \text{ のとき} \end{cases}$$

フィボナッチの兎の問題

・1 つがいの兎は，産まれて 2 か月後から毎月 1 つがいの兎を産む．
・1 つがいの兎は 1 年の間に何つがいの兎になるか？

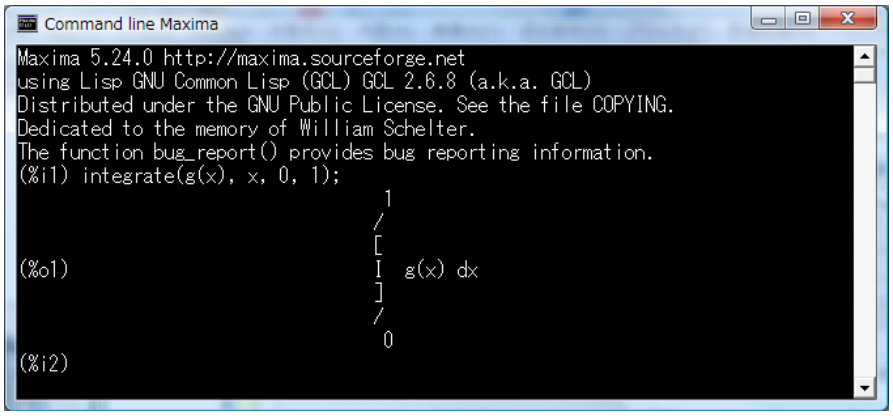
数式処理システム Maxima の基礎

1．数式処理システム Maxima とは（再掲）

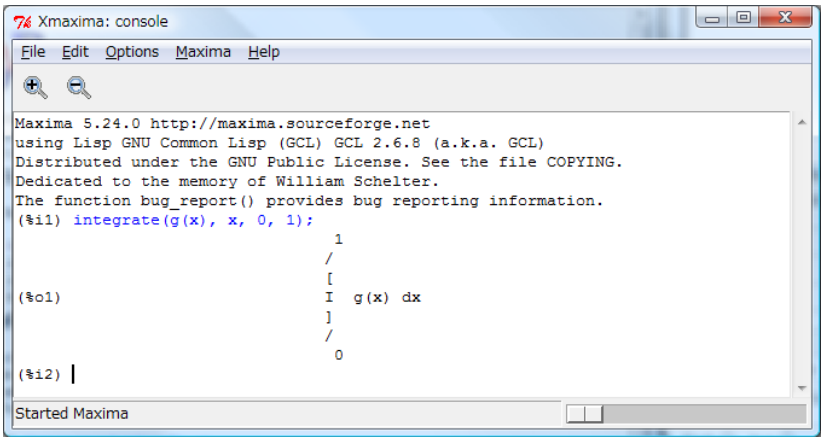
・GNU GPL に基づくオープンソースの数式処理システム（コンピュータ代数システム）である．
・1960 年代に MIT で開発され，米国エネルギー省（DOE）で配布されていた MACSYMA を，テキサス大学の Schelter 氏が Common Lisp 環境で移植したものである．
・コマンド処理，バッチ処理によるプログラミングが可能である．
・公式ホームページは以下を参照．
http://maxima.sourceforge.net/
・インストールプログラムは下記より入手可能である．
http://sourceforge.net/projects/maxima/files/
・Maxima の参考となる資料
http://www.bekkoame.ne.jp/~ponpoko/Math/maxima/MaximaMAIN.html
http://www.bekkoame.ne.jp/~ponpoko/KNOPPIX/MaximaBook.pdf 等

2．Windows 上での Maxima の起動

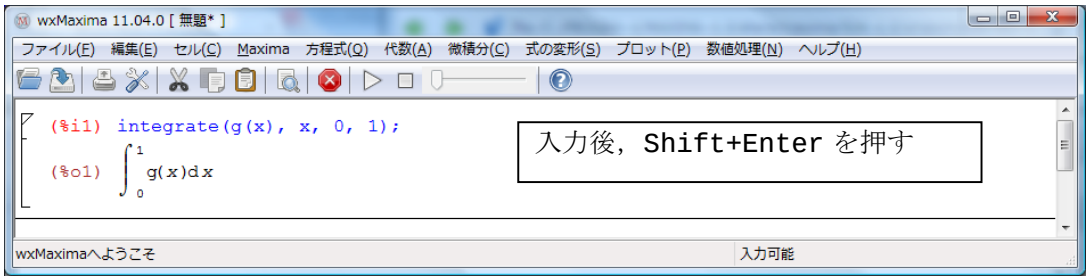
・コマンドライン：
すべてのプログラム -> Maxima-5.x.x -> Command line Maxima を選択・起動



・グラフィカルインターフェース（低度） xMaxima
すべてのプログラム -> Maxima-5.x.x -> xMaxima を選択・起動



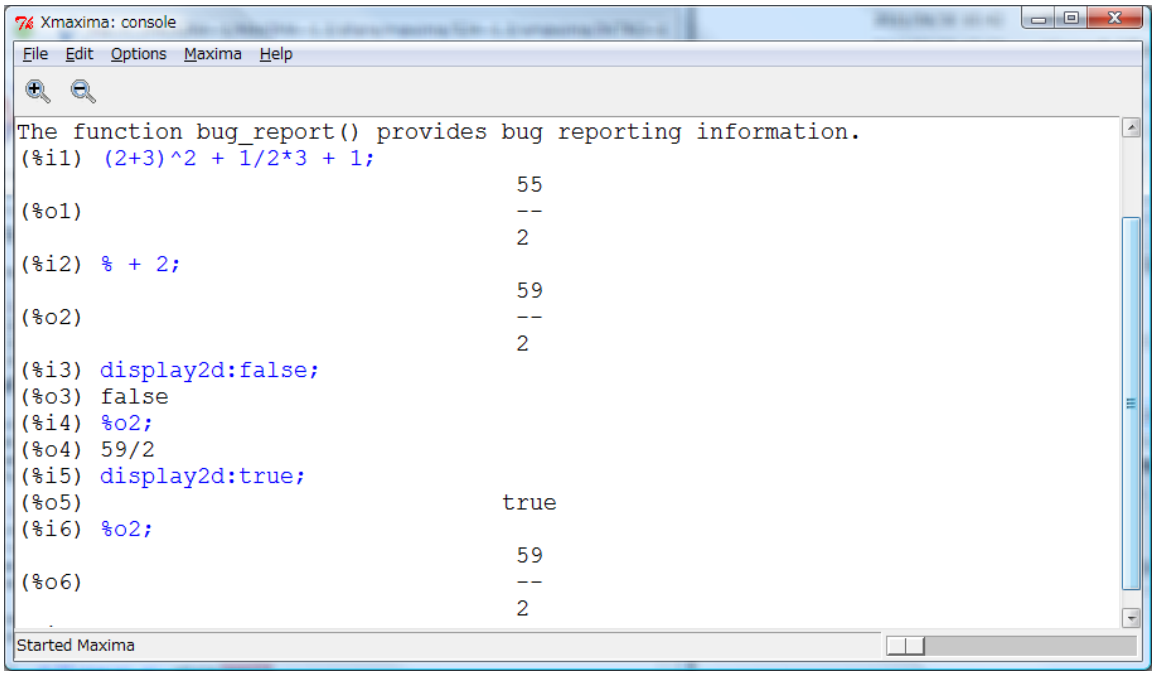
・グラフィカルインターフェース（高度）　wxMaxima
　すべてのプログラム　-> Maxima-5.x.x -> wxMaxima を選択・起動



3. Maxima 上での簡単な演算

(1)二項数式演算子
+ : a と b の和
- : a と b の差
* : a と b の可換積
/ : a の b による商
** または ^ : べき　(例 : a ** 2　a ^ b)
. : 非可換積　(例：a . b)　　^^: 非可換べき　(例：a ^^ b)
n!: 階乗　n!!: n が偶数の場合は n 以下の偶数の積，n が奇数の場合は n 以下の奇数の積

例）演算実行例



※　入力行の末尾には必ずセミコロン; を付すること

・display2d:true または　display2d:false　　表記方法の切り替え
・% は直前の出力値を示す.
・任意の入力値，出力値の参照は　%i*n*　または　%o*n* (*n*　はラベル番号)
・過去の出力値の参照 %th(*n*)　*n* 回前の出力値を参照
・kill(labels); すべてのラベルの値を破棄

(2)シンボル（変数）への値の割り当て

シンボル:値　例） a:2;

実行例)

```
(%i1) a:2;
(%o1)                          2
(%i2) b:3;
(%o2)                          3
(%i3) A:4;
(%o3)                          4
(%i4) a + b + A;
(%o4)                          9
(%i5) a ^ b + A;
(%o5)                         12
```

※大文字と小文字は区別される.
※kill(シンボル); 指定したシンボルの値を破棄
kill(all); 全ての値を破棄

４．演算子の定義

(1)前置演算子 prefix の定義

定義：prefix(“演算子名”);
演算子名 変数 := 式;

```
(%i1) prefix("squared");
(%o1)                                squared
(%i2) squared x:=x^2;
                                              2
(%o2)                           squared x := x
(%i3) squared 4;
(%o3)                                   16
```

(2)後置演算子 postfix の定義

定義: postfix(“演算子名”);
変数 演算子名 := 式;

```
(%i1) postfix("postsquared");
(%o1)                     postsquared
(%i2) x postsquared := x^2;
                                            2
(%o2)                x postsquared := x
(%i3) 4 postsquared;
(%o3)                          16
```

(3)内挿演算子　nary の定義

定義: nary (“演算子名”);
　　変数 1 演算子名 変数 2 := 式;

```
(%i1) nary("narysquared");
(%o1)                    narysquared
(%i2) a narysquared b := (a+b)^2;
                                                  2
(%o2)           a narysquared b := (a + b)
(%i3) 2 narysquared 3;
(%o3)                          25
```

５．数式の操作

(1)式の展開：　expand（式）

```
(%i1) (x+y)^4+(x+2*y)^2+(x+y);
                                      2           4
(%o1)                      (2 y + x)  + (y + x)  + y + x
(%i2) expand(%);
       4        3      2  2      2      3                   4    2
(%o2) y  + 4 x y  + 6 x  y  + 4 y  + 4 x  y + 4 x y + y + x  + x  + x
```

(2)式の因数分解：　factor(式)

```
(%i18) x^4+2*x^3*y + 2*x*y^3 -y^4;

                     4        3      3      4
(%o18)           - y  + 2 x y  + 2 x  y + x

(%i19) factor(%);

                     2    2    2            2
(%o19)          - (y  + x ) (y  - 2 x y - x )
```

(3)式の簡単化：　ratsimp(式)

```
(%i1) 1/(x-1)-1/(x^2+x+1);

                       1         1
(%o1)                ----- - ----------
                     x - 1    2
                             x  + x + 1

(%i2) ratsimp(%);
                           2
                          x  + 2
(%o2)                     ------
                           3
                          x  - 1
(%i3)
```

６．仮定

(1) declare(変数名，属性)
　例）
　　declare(k,integer)　　k を整数として宣言
　　declare(k,noninteger)　k を非整数として宣言
　　declare(k,even)　　k を偶数として宣言
　　declare(k,odd)　　k を奇数として宣言
　　declare(k,rational)　　k を有理数として宣言
　　declare(k,irrational)　k を無理数として宣言
　　declare(k,real)　　k を実数として宣言
　　declare(s,alphabetic)　s をアルファベットとして宣言　　など

例 1)

```
(%i1) expand((a*b)^k);
                                          k
(%o1)                                  (a b)
(%i2) declare(k,integer);
(%o2)                                   done
(%i3) expand((a*b)^k);
                                         k  k
(%o3)                                   a  b
```

(2) assume(式)
　例)
　　assume(x > 0)

例 2)

```
(%i1) sqrt(x^2);
(%o1)                        abs(x)
(%i2) assume(x>0);
(%o2)                        [x > 0]
(%i3) sqrt(x^2);
(%o3)                           x
```

７．方程式の解を求める

solve(式, 変数)

例 1)一元方程式

```
(%i1) (x+1)^3 = -1;
                                3
(%o1)                    (x + 1)  = - 1
(%i2) solve(%, x);

                sqrt(3) %i + 1      sqrt(3) %i - 1
(%o2) [x = - --------------, x = --------------, x = - 2]
                      2                   2
```

または　（直接，solve に数式を与えてもよい）

```
(%i3) solve((x+1)^3 = -1, x);

                sqrt(3) %i + 1      sqrt(3) %i - 1
(%o3) [x = - --------------, x = --------------, x = - 2]
                      2                   2
```

例 2）連立方程式

```
(%i1) 2*x+3*y=5;
(%o1)                    3 y + 2 x = 5
(%i2) 5*x+2*y=4;
(%o2)                    2 y + 5 x = 4
(%i3) solve([%o1, %o2], [x,y]);
                             2       17
(%o3)                [[x = --, y = --]]
                             11      11
```

※直接，2 式を入力してもよい．`solve([2*x+3*y=5, 5*x+2*y=4], [x,y]);`

例 3) 微分

```
(%i1) 2*x^2-3*x;
                              2
(%o1)                       2 x  - 3 x
(%i2) diff(%o1, x);
(%o2)                         4 x - 3
```

例 4) 2 階微分

```
(%i1) x^4+3*x+5;
                             4
(%o1)                       x  + 3 x + 5
(%i2) diff(%o1,x);
                               3
(%o2)                        4 x  + 3
(%i3) diff(%o1,x,2);
                                 2
(%o3)                          12 x
```

例 5) 不定積分

```
(%i1) sin(2*x+1);
(%o1)                      sin(2 x + 1)
(%i2) integrate(%,x);
                            cos(2 x + 1)
(%o2)                     - ------------
                                  2
```

例 6）定積分$\int_0^2 4x^3\,dx$

```
(%i1) integrate(4*x^3,x,0,2);
(%o1)                           16
```

例 7)常微分方程式$m\frac{d^2x(t)}{dt^2} = -kx(t)$　を解く

```
(%i1) ode2('diff(x(t),t,2)*m = -k*x(t), x(t), t);
Is  k m  positive, negative, or zero?

pos;
                       sqrt(k) t             sqrt(k) t
(%o1) x(t) = %k1 sin(---------) + %k2 cos(---------)
                        sqrt(m)               sqrt(m)

(%i2) ic2(%o1, t=0, x(t)=0, diff(x(t),t)=10);

                              sqrt(k) t
             10 sqrt(m) sin(---------)
                               sqrt(m)
(%o2) x(t) = ------------------------
                        sqrt(k)
                                                  sqrt(k) t
                                       + x(0) cos(---------)
                                                   sqrt(m)
```

※ode2 は一階または二階の常微分方程式の一般解を求める.
※'diff は導関数を表す.
※ic2 は二階微分方程式の初期値問題を解く.　(一階微分方程式の場合は ic1)

7. 行列

行列の定義　matrix（行 1, 行 2, ・・・）

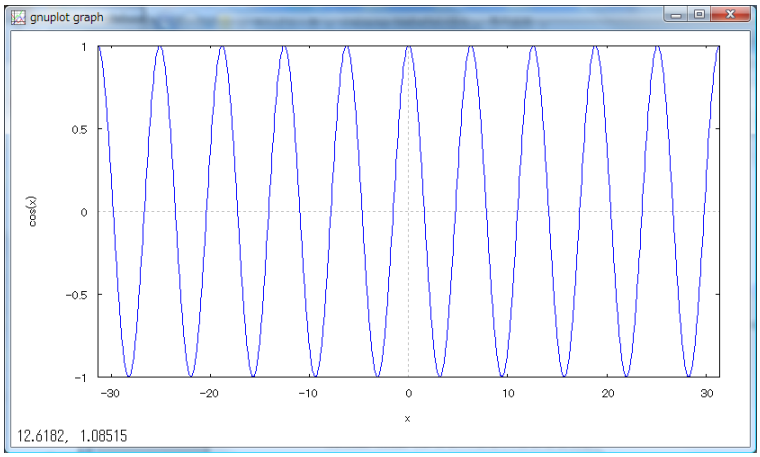
```
(%i3) A:matrix([1,2,3],[4,5,6],[7,8,9]);
                          [ 1  2  3 ]
                          [         ]
(%o3)                     [ 4  5  6 ]
                          [         ]
                          [ 7  8  9 ]
(%i4) B:matrix([9,8,7],[6,5,4],[3,2,1]);
                          [ 9  8  7 ]
                          [         ]
(%o4)                     [ 6  5  4 ]
                          [         ]
                          [ 3  2  1 ]
(%i5) A-B;
                       [ - 8  - 6  - 4 ]
                       [               ]
(%o5)                  [ - 2   0    2  ]
                       [               ]
                       [  4    6    8  ]
(%i6) A+B;
                        [ 10  10  10 ]
                        [            ]
(%o6)                   [ 10  10  10 ]
                        [            ]
                        [ 10  10  10 ]
(%i7) A . B;
                       [ 30   24   18 ]
                       [              ]
(%o7)                  [ 84   69   54 ]
                       [              ]
                       [ 138  114  90 ]
```

```
(%i8) A ^^ 2;
                        [ 30   36   42  ]
                        [               ]
(%o8)                   [ 66   81   96  ]
                        [               ]
                        [ 102  126  150 ]
```

8．行列

Octave 同様に gnuplot の命令を用いることができる.

```
(%i1) plot2d(cos(x), [x, -10*%pi, 10*%pi]);
```



9．Maxima の終了 (Command line Maxima の場合)

```
(%i1) quit();
```

演習課題

1．以下の演算子を定義し，そのコードおよび実行例を示しなさい.
(1)与えられた引数を半径とする円の面積を求める Prefix, Postfix 演算子をそれぞれ定義し，その実行例を示しなさい.
(2)演算子の前後それぞれの引数を二乗し，その和を求める関数を nary 演算子を定義し，実行例を示しなさい.

2．任意の 3 次以上の方程式を定義し，展開・因数分解およびその解を求めなさい.
3．サイクロイド曲線 $x = a(t - \sin t), y = a(1 - \cos t)$ $(0 \leqq t \leqq 2\pi)$ と x 軸で囲まれる面積 A を求めなさい.
4．任意の常微分方程式を用いて，その一般解および任意に設定した初期値に基づく特殊解を求めなさい.
次回講義までに上記を PDF ファイルに取りまとめ，oyojoho@gmail.com に提出すること.

※円周率πは定数 %pi を用いる.
※%pi を数値として出力するためには float( )を用いる.
今後の講義予定：
7/23(月)，8/1(水)，8/6(月)　※7/16(海の日)，7/30(ｵｰﾌﾟﾝｷｬﾝﾊﾟｽ)のため休講

(総合演習)遺伝的アルゴリズム

1. 遺伝的アルゴリズム(GA)とは

遺伝的アルゴリズム（Genetic Algorithm; GA）とは，生物が環境に適応しながら進化していくプロセスを模倣した学習的アルゴリズムである．ミシガン大学のジョン・ホランド（John H. Holland）により考案され 1975 年に発表された．自然淘汰のシミュレーションを行い，最適解を求めようとするものであり，組み合わせ数が多く計算が非常に困難な問題や非線形問題の最適化等において，その近似解を高速に求める方法として用いられている．

2. GA のプロセス

GA のプロセスは図-1 の通りである．

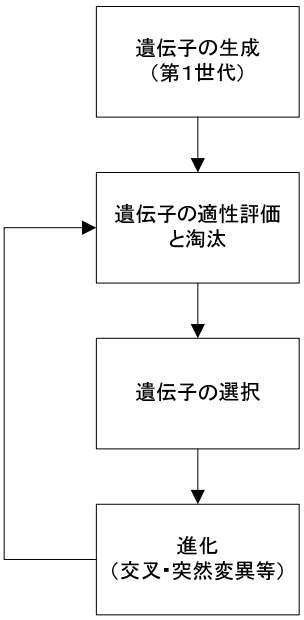


図-1　遺伝的アルゴリズム（GA）のプロセス

(1)遺伝子の生成
同一の遺伝子パターン（遺伝子型）の複数の個体を，乱数を用いて生成させる．

個体１　０１０００００１１
個体２　１１１０００００
個体３　１１０１１００１
個体４　１０１０１１００

(2)遺伝子の適性評価と淘汰
それぞれの遺伝子の個体に対して，環境への適応性を評価する．適応性の低いものについては削除（淘汰）する．

(3)遺伝子の選択
　各遺伝子の個体集団の中から交叉を行う個体を選択する．選択方法として以下のような方法がある．

・ルーレット方式（適応度に応じた比率で個体を選択する）
・エリート戦略（最も適応度の高いものを選択する）
・ランク方式（各個体を適応度が高い順にランク分けし，定義した関数に基づき選択する）
・トーナメント方式（集団から個体の組み合わせを作成し，適合度についてのトーナメントを行い，個体を選択する.）

(4)進化
・交叉：選択された２つの親の遺伝子を交叉させ，子を作る．

個体１（親）　０　１　０　０　１　０　１　１
　　↑交叉位置
個体２（親）　１　１　１　０　０　０　０　０
　　↑交叉位置

個体１（子）　０　１　０　０　０　０　０　０
　　（親１の遺伝子）↑（親２の遺伝子）
個体２（子）　１　１　１　０　１　０　１　１
　　（親２の遺伝子）↑（親１の遺伝子）

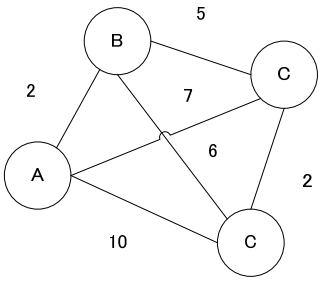
※複数点交叉も可

・突然変異：遺伝子を一定の確率で突然変異させる．（新しいものを試す）

個体　(A)　１　１　１　０　１　０　１　１
　　↓　　（突然変異させる）
個体２（B）　１　１　１　０　１　１　０　１

3. 代表的な GA の利用例―巡回セールスマン問題（TSP）

　サラリーマンが複数の都市を巡回訪問する場合の最短経路を求める問題である．GA による適用例として有用である．

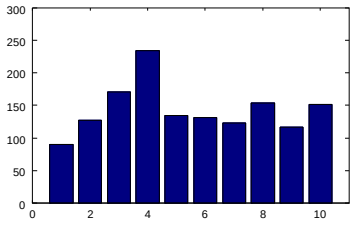


遺伝子へのエンコーディングの例
A→B→C→D：００００
A→D→B→C：０２００
A→B→D→C：００１０

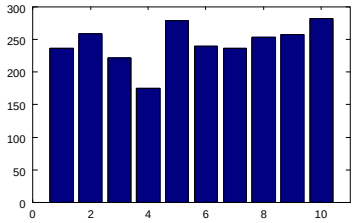
解：A→B→C→D　距離９

※具体的なアルゴリズムについては下記が参考になる（ただし言語は C#である）
http://www.microsoft.com/japan/msdn/academic/Articles/Algorithm/05/

4. GA の応用プログラムの例（植物の進化シミュレーション）

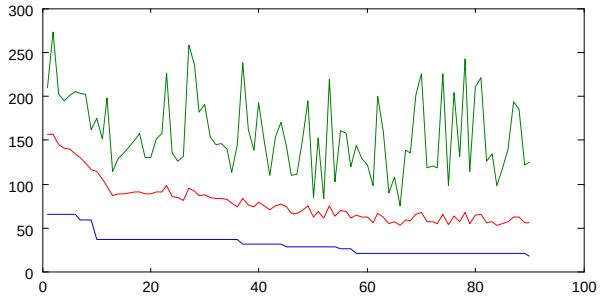


第 1 世代　適応度



第 90 世代　適応度

(各遺伝子の適応度を(300-differences)として描画)



青：相違度 differences 最小値（適応度：高）
緑：相違度 differences 最大値（適応度：低）
赤：相違度 differences 平均値

ファイル名：genetic.m

```
function [ ret ] = genetic1()
    #factors
      #Temperature:1 water:2, Sunlight:3, Nutrient:4
      #Beneficial Insect:5, Harmful Insect:6

      more off;
       seedN = 10;    #number of seeds
      maxLevel = 75  #maximum level of each factors
      factors = 6;  #number of factors
      p = 300;  #mutation potential
      criteria = 20; #criteria for break

     # rand("seed", 5);     #situation will be fixed (remove #)

      #Set current environment
      currentEnv = floor(maxLevel*rand(1,factors)+1)

      #Encodeing
      code = floor(maxLevel*rand(seedN,factors)+1)
```

```
      for n = [1:1000]
            for k = [1:seedN]
                  tmpfit = abs(code(k,:)-currentEnv);
                  differences(k,:) = [tmpfit sum(tmpfit)];
            endfor
            differences
            [s, i] = sort(differences(:,factors+1));


#        draw Fitness(300-differences) as the height of plants
#         (remove # )
#         if (mod(n, 10)==0)
#               bar(300-differences(:, factors+1 ))
#               axis([0,seedN+1, 0, 300], "square");
#               drawnow;
#           endif

            #Score Print
            printf("Generation : %d\n", n);
            printf("Minimum  differences : %d\n", s(1));
            printf("Maximum  differences : %d\n", s(seedN));
            printf("Mean differences : %d\n", mean(s));

            #Score Plot
            score(n, 1)=s(1);
            score(n, 2)=s(seedN);
            score(n, 3)= mean(s);
            plot(score);
            axis("auto");
            drawnow();

            #worst = code(i(seedN),:)
            currentEnv  #print currentEnv
            best = code(i(1),:)  #print the best code

            if (s(1) < criteria)
                  disp ("Differences has reached the criteria")
                  break;
            endif


            #Evolution

            #Selection - The worst is changed
            code(i(seedN),:)=floor(maxLevel*rand(1,factors)+1);
            code
            disp("The worst has been changed!");

            #Reproduction -The best is preserved
            #Crossover
            for j = [1:factors]
                  r = randperm(seedN-2)+1;
                  if (j == 1)
                        tmp = code(i(r), j);
                  else
                        tmp = [tmp code(i(r),j)];
                  endif
            endfor
            code(i(2:seedN-1),:) = tmp;
```

```
            #Mutation
            for k = [2:seedN-1]
                  for j = [1: factors]
                        r = floor(p * rand) + 1;
                        if (r == p)
                              printf("Code (%d, %d) is changed!\n", k, j)
                              code(i(k), j) = floor(maxLevel*rand+1);
                        endif
                  endfor
            endfor
      endfor
endfunction
```

※プログラムの強制停止：ctrl+s　⇒　プログラムの強制終了：ctrl+c
※randperm(n)　1 から n までをランダムに並べた数列を作る
※[s,i] = sort(x)　行列 x をソート（小さい順に並び変える）
戻り値 s は並び替えた後の行列，i は並び替えた順の要素番号(index)を示す行列を表す

```
octave:16> a = [4 5 2 1 3]
a =
 4   5   2   1   3
octave:17> [s, i]= sort(a)
s =
 1   2   3   4   5
i =
 4   3   5   1   2
```

※参考書：Devid M.Bourg, Glenn Seemann「ゲーム開発者のための AI 入門」オライリージャパン

【演習課題】
１．上記の遺伝的アルゴリズムのプログラムをもとに，任意の条件を設定し，その 10 回の試行結果について基準値(criteria)に到達するまでの世代数を調べなさい．
※試行結果はプログラムにより作成し，グラフとして示すことが望ましい．

２．以下で述べる総合課題についての草案を作成しなさい．
期限は８月１日講義開始前までに　oyojoho@gmail.com に提出する．

【総合課題(1)】
これまでに学習したことをもとに，土木工学，建築･社会環境工学に関連したプログラム（シミュレーション・ゲームなど）を作成しなさい．

・プログラムは Octave により構築する．
・例題形式として以下のフォーマットにより文書ファイル（必ず PDF ファイルとすること）を作成し，提出する．（文書内には学籍番号・氏名等は記入しない）
・コードおよび説明に必要な実行画面などを示すこと．
・文書にあわせて作成したプログラムの m ファイルを提出する．

最終期限は８月６日（月）８時５０分まで（必着），oyojoho@gmail.com に提出する．

８月６日（月）２時限にそれをもとにプレゼンテーションを行う予定である．

【総合課題(2)】
上記総合課題(1)をもとに，改良，考察等を加え，８月９日（木）24:00 までに最終課題として提出する（詳細は次回，改めて指示する）．

(総合課題(1)フォーマット：文書内に学籍番号・氏名などは記入しない)

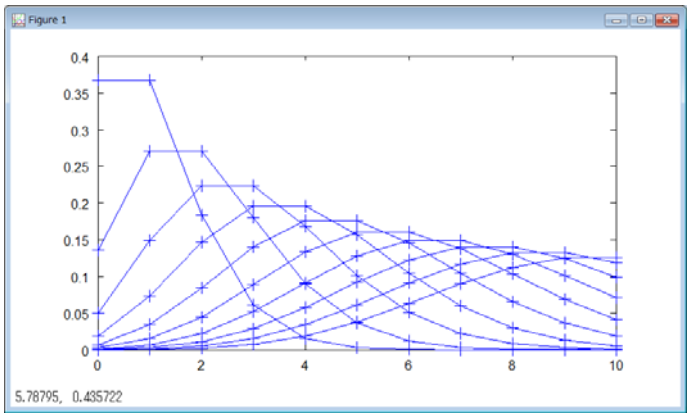
課題の作成例：
(1)交通量が少ない場合の車両の到着（発生）はポアソン分布で近似できることが知られている．ある料金所に 1 分間に平均 2 台の車両が到着するときに，60 分間の 1 分毎の車両の到着台数をシミュレートしなさい．到着台数を示すグラフ及びコードを示しなさい．
(2)の料金所において 1 分間に 2 台の処理が可能であるとする．(2)と同じ 60 分間の 1 分毎の待ち台数の数を(2)と同じグラフ上に示すプログラムを作成しなさい．

以下に説明文，コード，出力画面，考察を示すこと．
(ページ数に制限はない)

(1)

```
function [ ret ] = poisson2 ()
      hold on;
      for lambda = [0:10]
            x = [0:10];
            y = poisspdf(x, lambda);
            plot(x, y, '-@');
      endfor
endfunction
```

出力画面は以下の通りである．



(2)

```
function [ ret ] = poisson3 ()
      clf;
      hold on;
```

(略：枚数・字数などの制限はなし.)

考察：
自由に記入する．

システム設計

1. 情報システム開発の基本的項目

情報システム開発の流れにおける基本的項目は，以下の通りである．

①要求分析・定義　：利用者が要求するシステム機能仕様の定義
②外部設計　　：要求定義を満たすためのシステム機能の定義
③内部設計　　：システムの内部構造と処理ロジックの設計
④テスト　　：検証のための実施

2. 情報システムの開発プロセス

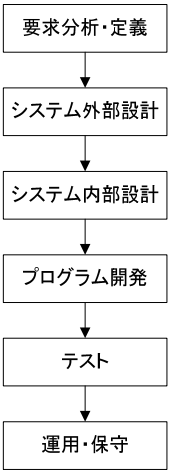
情報システムの開発プロセスモデルには，①ウォータフォールモデル，②プロトタイピングモデル，③スパイラルモデルの３つの手法がある．

①ウォータフォールモデル

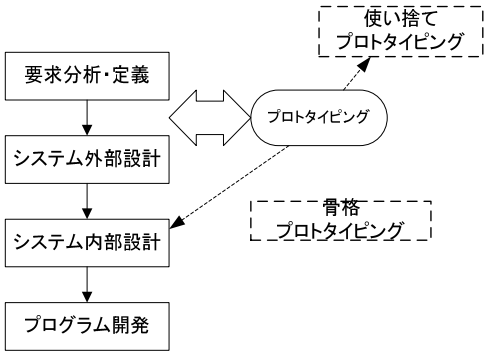
情報システムの開発をいくつかのフェーズに分割し，各フェーズにおける作業を完了した後，次の段階へと進む方法である．大規模な開発プロジェクトに向く．開発工程の後戻りを考えないので，開発管理は容易である．しかし，初期段階でシステム全体の重要な定義事項を明確に定める必要があり，要件が明確に定まらない場合には不適である．

②プロトタイピングモデル

システム開発の初期段階でプロトタイプ（試作品）を作成し，システム要件の確定と懸案事項の早期解決を図る．開発コストの削減や開発期間の短縮が期待できる．



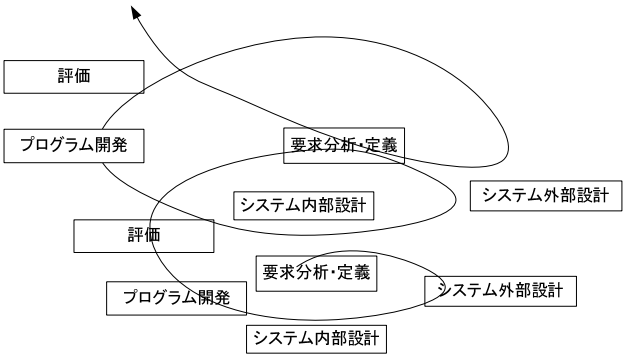
ウォータフォールモデル



プロトタイピングモデル

③スパイラルモデル

スパイラルモデルは，プロタイピングを利用し，短期間に要件定義から実際のシステム開発を行うことを繰り返しながら，システムの完成度を高めていく方法である．



スパイラルモデル

その他の便利なフリーウェア

(1)画像処理ソフトウェア　GIMP　(フリーウェア)
・ペイントや空間フィルタやトーンカーブを用いた画像処理，画像合成等を GUI を用いて容易に行うことができる．
・便利な機能：スタンプツール（画像合成や加工の場合の手作業に有用）

架空線地中化の例

(2)統計解析ソフトウェア　R　（フリーウェア）
・基礎統計量や相関，多変量解析（主成分分析，因子分析）などができる．
・スクリプトを用いたプログラミングも可能である．

(3)プログラミング環境 Microsoft Visual Basic
・Microsoft が開発した Basic をベースとしたオブジェクト指向の開発環境である．
・高級言語によるグラフィックスインターフェースを用いたプログラミングが可能である．
・公式ホームページ，インストールプログラムは下記より入手可能である．
http://www.microsoft.com/japan/msdn/vstudio/express/

【総合課題(1)】
これまでに学習したことをもとに，土木工学，建築･社会環境工学に関連したプログラム（シミュレーション・ゲームなど）を作成しなさい．

・プログラムは Octave により構築する．
・例題形式として以下のフォーマットにより文書ファイル（必ず PDF ファイルとすること）を作成し，提出する．（文書内には学籍番号・氏名等は記入しない）
・コードおよび説明に必要な実行画面などを示すこと．
・文書にあわせて作成したプログラムの m ファイルを提出する．

最終期限は８月６日（月）８時５０分まで（必着），oyojoho@gmail.com に提出する．

８月６日（月）２時限にそれをもとにプレゼンテーションを行う予定である．

【総合課題(2)】
上記総合課題(1)をもとに，改良や考察，課題などを加え，最終課題として提出する．

提出期限：　8 月 9 日（木）24:00 まで
提出先：　oyojoho@gmail.com

応用情報処理演習 B まとめ

1. 数値解析ソフトウェア Octave を用いたプログラミング

・本演習での学習内容
  ・使用方法と行列計算
  ・関数描画とグラフィックス
  ・画像処理－画像解析－
  ・画像処理－地形情報の処理－
  ・擬似乱数とモンテカルロシミュレーション
  ・経路探索アルゴリズム
  ・遺伝的アルゴリズム

・信号処理，画像処理，ビデオ処理，音声処理，統計解析等，さまざまなパッケージが用意されており，容易に処理を行うことができる．
・同様の言語を用いたシステムとして，以下がある．

MATLAB（MathWorks 社）
SciLab（フリーウェア） http://scilab.org

MATLAB，SciLab を用いる場合，下記の言語仕様の違いについて注意が必要．

[Octave]	for…endfor	if…endif	
[Matlab, Scilab]	for…end	if…end	※Octave でも可能

2. 数式処理システム Maxima の基礎

・展開，因数分解，方程式の解を容易に求めることができる．
・Octave 同様にスクリプトによるプログラミングも可能である．
・同様のソフトウェア Mathematica (Wolfram Research 社)

3. その他のソフトウェア

(1)画像処理ソフトウェア GIMP (フリーウェア)
(2)統計解析ソフトウェア R （フリーウェア）
(3)プログラミング環境 Microsoft Visual Basic

4. 土木における情報技術のこれから

(1)画像処理: OpenCV（動画像解析）
(2)図形処理(CAD，CG，GIS) :
GIS(QGIS，GRASS)，CG::OpenGL, Blender(CG),Unity(Game Engine)
(3)Augmented Reality (AR), Virtual Reality (VR):ARToolkit
(4)ファジー・遺伝的アルゴリズム（GA)，人工知能（AI:ニューラルネットワーク等）
(5)シミュレーション（マルチエージェントシミュレーション等）
(6)センサー技術（ヘルスモニタリング）
(7)ロボティクス（情報化施工，無人施工等）
(8)情報マネジメント(ITS, CALS/EC, 統合システム)

【総合課題(2)】

これまでに学習したことをもとに，土木工学，建築･社会環境工学に関連したプログラム（シミュレーション・ゲームなど）を作成しなさい．

・プログラムは Octave により構築する．
・例題形式として以下のフォーマットにより文書ファイル（必ず PDF ファイルとすること）を作成し，提出する．**（文書内には学籍番号・氏名を記入する）**
・コードおよび説明に必要な実行画面などを示すこと．
・文書にあわせて作成したプログラムの m ファイルを提出する．

8 月 6 日までに提出したものに，改良や考察，課題などを加え，最終課題として提出する．

提出期限： 8 月 9 日（木）24:00 まで
提出先： oyojoho@gmail.com

※提出した課題は，作成者明記の上，プログラムの例として利用する場合がある．