**<MATLAB>**

**○MATLAB Graderの使い方**

・式をべたっと書くと「ans = 値」となって返ってくる

3 + 5 → ans = 8

・変数をべたっと書くと「変数 = 値」となって返ってくる

a = 10 → a = 10

・式の代入をすると「変数 = 値」となって返ってくる

b = 2 \* 3 → b = 6

・式の最後にセミコロンをつけると出力がなくなる

x = 10^3 → 出力なし

**○MATLABの基本**

・文字列はシングルクォーターで囲む「’文字列’」

c = ‘Hello World’ → Hello World

・Excelファイル、テキスト、写真、音楽などのファイルからデータを読み取りたいとき、「readtable(‘ファイル名.拡張子’)」とする

・変数の名前を出力せずに表示したいときは「disp(値)」とする

・コメントアウトするときは「%」をつける

・論理演算子

and → &, or → |, not → ~

・for文は「for ループ変数 = ループ範囲」から始まり、タブでスペースを空けてforと同じ段落で「end」で閉じる。ループ変数は配列の手前からひとつずつ使われる

for i = 1 : 4

A(i) = 2 \* i

end

→ A = [2 4 6 8]

・while文は「while 条件」から始まり「end」で閉じる

例. 10000を超えない範囲でフィボナッチ数列fを作る

f(1) = 1;

f(2) = 1;

i = 3

while f(i-1) + f(i-2) < 10000 (f(i) < 10000 と書くと10000を1つ超えるので注意)

f(i) = f(i-1) + f(i-2)

i = i + 1

end

・if文は「if 条件」で始まり「elseif」と「else」を必要ならば入れて「end」で閉じる

例. -10から10までのベクトルに対して正なら1を負なら-1を0なら0に値を変える

r = -10:10;

for i = 1 : length(r)

if(r(i) > 0) r(i) = 1;

elseif(r(i) < 0) r(i) = -1;

else r(i) = 0;

end

end

・ベクトルや行列の大きさを逐一変えるより事前割当をした方が計算量は減る

・「zero(a, b)」でa行b列の0のみ入った行列を作る

・「cell(a, b)」でa行b列の””のみ入った行列を作る

**○パブリッシュ**

・パブリッシュすることで書いたコードと実行結果を同時に確認できるhtmlやPDFが作成できる

・実行した後に「パブリッシュ」を押すだけで良い

・「%%」とすると段落を作ることができる

**〇ローカル関数**

・関数は「function 変数 = 関数名(引数)」から初めて本文を書いて「end」で占める

・変数は本文で用いるやつ。複数用いるときはリストにする。

・関数はメイン関数の下に書く

例、デルタ関数

メイン関数

function a = delta(n)

a = [zeros(1, n+2), 1, zeros(1, 7-n)];

end

使うときは「関数名(引数)」とする

**○ベクトル**

・1 2 3 4 5というベクトルを作成するときは「v = [1 2 3 4 5]」とする。囲いは()でないことと、値の間はカンマではなく、spaceとすることに注意

・ベクトルのインデックスの最初は0ではなく1であることに注意

v = [1 2 3 4 5]

v(2) → 2

・列ベクトルへの変換は「変数’」とシングルクォーターを後ろにつける

x = [1 2 3 4 5]

y = x’

・コロンを用いてベクトルを作る。「初期値 : 増分 : 最終値」とすると初期値から増分ごとに値が増え、最終地を越えることのないベクトルを生成する

v = 0 : 3 : 10 → 0 3 6 9

・増分1のベクトルを作るときは引数が2つでよい「初期値：最終値」

v2 = 0 : 5 → 0 1 2 3 4 5

・要素数を指定してベクトルをつくるときは「linspace(初期値, 最終値, 要素数)」

v3 = linspace(0, 10, 6) → 0 2 4 6 8 10

・ベクトルから要素を取り出したり代入したりするときは「名前(インデックス)」とする

v3(3) → 4

・ベクトルの長さは「length(名前)」とする

length(v3) → 6

・ベクトルに対して、比較演算子 < , > , == を用いると正しいときに1, そうでないときに0を入れる同じ形の論理値ベクトルを返す

v4 = [1 2 3 4 5]

v4 > 2 → [0 0 1 1 1]

・ベクトルの要素の中から条件に合うものを取り出すときは「名前(条件)」とする

v4(v4 > 2) → 3 4 5の行ベクトル（行列の場合は列ベクトルが出力されることに注意）

・ベクトル同士の演算がいつも通りできる

・（同じ大きさの）ベクトルの和

A = [2 2 2]

B = [1 2 3]

A + B → 3 4 5

・ベクトルのスカラー倍

A \* 2 → 4 4 4

・ベクトルの要素ごとの積はピリオドを前につける「.\*」列ベクトルごと、行ベクトルごと

A .\* B → [2 4 6]

・ベクトルの内積は「行ベクトル \* 列ベクトル」とする。順番が大事！！

・ベクトルの要素の合計値 sum(名前)

sum(1 : 100) → 5050

・ベクトルの要素の平均値 mean(名前)

・ベクトルの要素の標準偏差 std(名前)

・ベクトルの要素の最大値 max(名前)

・ベクトルの要素の最小値 min(名前)

・ベクトルの各要素に平方根をつける sqrt(名前)

・全ての値が0であるベクトルを作るときは「zeros(1, 要素数)」とする。この1は1次元であることを表す1だよ！

・全ての値が1である配列を作るときは「ones(1, 要素数)」

・ベクトルの中にベクトルを書くこともできる

例、デルタ関数（地域は-10から10までで、0のときに1となる。他は0）

Y = [zeros(1, 10), 1, zeros(1, 10)]

**○行列**

・m行n列の行列を作るときは[1 2 3 … m; 2 2 3 … m; 3 2 3 … m; … n 2 3 … m;]とする。「;」が行の区切りとなる

A = [1 2 3; 4 5 6] → 1 2 3

4 5 6

・行列から要素を取り出したり代入したりするときは「名前(インデックス1, インデックス2)」とする

A(2, 2) → 5

・インデックスに数字ではなく「:」とすると「全部」を意味する

A(2, :) = 7→ 1 2 3

7 7 7

・行列の大きさは「size(名前)」とすると次元ごとの要素数を返す

size(A) → 2 3

・全ての値が0である配列を作るときは「zeros(縦の要素数, 横の要素数)」

Z = zeros([2 4]) → 0 0 0 0

0 0 0 0

・すべての値が1である配列を作るときは「ones(縦の要素数, 横の要素数)」

・行列の要素の中から条件に合うものを取り出すときは「名前(条件)」とする。

B = [1 2 3; 4 5 6; 7 8 9]

B(2 < B & B < 6) → 3 4 5の列ベクトル

・「名前(条件) = 値」とすると条件に一致する要素が全て指定した値に更新される

・行列の和はそのまま A + B (サイズは同じじゃなきゃダメ)

・行列の積はそのまま A \* B (Aの列数とBの行数が等しくないとダメ)

A = [1 2; 3 4]

B = [1 2]’

A \* B → 5

11

・行列の要素ごとの積は演算子「.\*」を用いて A .\* B

A = [1 2 3; 4 5 6; 7 8 9]

B = [20 10 0; 3 2 1; 5 3 1]

A \* B → 20 20 0

12 10 6

35 24 9

・行列の要素の平均値 mean(名前, 次元) 次元を書いてないときは1となる

A = [1 2 3; 4 5 6; 7 8 9]

mean(A) → mean(A, 1) = 4 5 6

mean(A, 2) → 2

5

8

・行列の要素の最大値 max(名前, 次元) → 最大値を含む行ベクトルを返す

・行列のべき乗 A^2 = A \* A (Aは正方行列)

・逆行列 inv(A) (Aは正方行列)

**○数式**

・積は必ず「\*」をつけること！「2x」とかは認識してくれないので注意！

・べき乗はそのまま「2^5」とする

2^5 → 32

・平方根は「sqrt(値)」とする

sqrt(16) → 4

・円周率πは「pi」

・sin, cosはそのまま「sin(値)」とする。値はラジアン単位であることに注意

sin(pi / 2) → 1

・「y = sin(x)」としてxにベクトルを指定するとyもベクトルになる

・複素数はそのまま「a + bi」とする。積と異なり「\*」は必要ないことに注意！

・複素数の大きさや絶対値は「abs(値)」

・複素数の位相は「angle(値)」とする。出力はラジアン単位なので度数単位にしたいときは180 / piをかける

abs(1 + i) → 1.4142

angle(1 + i) → 0.7854

angle(1 + i) / pi \* 180 → 45

・複素数の図示は「compass(値)」

・ネイピア数eを底とする指数はそのまま「exp(値)」とする。値はベクトルでも良い

・ネイピア数eを底とする対数はそのまま「log(値)」とする。値はベクトルでも良い

**〇データの可視化（グラフ）**

・plot(X, Y)でグラフを書く（X, Yは同じ大きさのベクトル）

・x軸のラベル xlabel(“x軸の軸ラベル”)

・y軸のラベルylabel(“y軸の軸ラベル”)

・x軸の範囲 xlim([初期値 最終値])

・y軸の範囲 ylim([初期値 最終値])

・x軸とy軸の範囲を同時に指定するには「axis([x軸初期値 x軸最終値 y軸初期値 y軸最終値])」

例. y = sinx (0 < x < 7) のグラフをプロットする

x = 0 : pi/12 : 100;

y = sin(x);

plot(x, y)

xlabel("時間")

ylabel("振幅")

xlim([0 7])

ylim([-1.5 1.5])

・離散時間のデータをプロットしたいときはplotの代わりに「stem(X, Y)」とする。XとYは同じ大きさの

・離散時間フーリエ変換するときは「X = freqz(信号, 1, 周波数)」として、振幅スペクトルの表示は「plot(周波数, X)」、位相スペクトルの表示は「plot(周波数, angle(X))」とする

w = -pi:0.01:pi; %周波数の範囲 0.01ずつ区切ってグラフにする

x = 1/8\*[1 1 1 1 1 1 1 1]; %離散時間信号

X = freqz(x,1,w); %離散時間フーリエ変換

figure(1)

plot(w,abs(X)); %振幅スペクトルの図示

axis([-pi pi 0 1]); grid;

xlabel('Frequency \omega [rad]');

ylabel('|X(e^{j\omega})|');

figure(2)

plot(w,angle(X)); %位相スペクトルの図示

axis([-pi pi -pi pi]); grid;

xlabel('Frequency \omega [rad]');

ylabel('\angle X(e^{j\omega})');

・離散フーリエ変換の例

%（離散フーリエ変換の振幅スペクトルと位相スペクトル）

%離散フーリエ変換の計算例

%信号

x = [1 1 1 1 0 0 0 0]; %8点の信号

n = 0:length(x)-1; %時間のインデックス

%離散フーリエ変換 DFT

k = n; %周波数のインデックス

X = fft(x); %DFTの計算

magX = abs(X); %振幅スペクトル

maxX = max(magX);

figure(1)

stem(k, magX) %振幅スペクトルの図示

axis([0 length(k) 0 maxX]);

xlabel('Frequency k'); ylabel('|X(k)|');

figure(2)

stem(k, angle(X)) %位相スペクトルの図示

axis([0 length(k) -pi pi]);

xlabel('Frequency k'); ylabel('\angle X(k)');

・実行時間を計測するときはcputimeの差を用いる。事前に「tstart = cputime;」、処理が終わってから「tend = cputime;」として「time = tend - tstart」とする