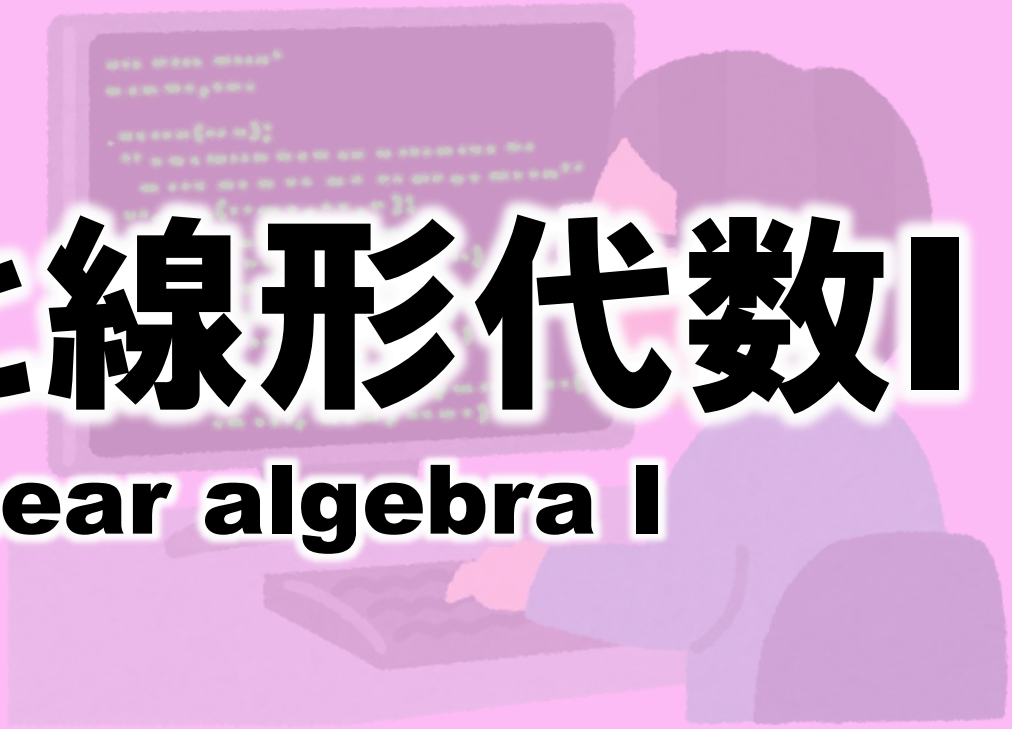
2018年6月4日更新



3. 行列と線形代数I

Matrices and linear algebra I

東北大学 大学院工学研究科
嶋田 慶太 shimada@m.tohoku.ac.jp

この副資料の場所:
http://www.pm.mech.tohoku.ac.jp/member/class

TOHOKU UNIVERSITY

本日の内容

- 行列の成分，行ベクトル，列ベクトル
- 単位行列，零行列，全要素1の行列，乱数行列
- 行列の演算（数学的定義～Octave内の独自演算）
- 計算例
- ノルム
- 演習問題

Octaveを用いた
行列操作・演算の基礎

Matrices (行列: 単数形はmatrix)

▶データは基本的に行列・ベクトルで表現される.

個別入力

```
>> A=[1,2,3;2,3,4]
A =
     1  2  3
     2  3  4
```

カンマ「,」
次の要素

セミコロン「;」
次の行へ.

行ごとの要素数が異なる
▸エラーとなる.

```
>> C=[1,2;2,3,4]
error: vertical dimensions mismatch (1x2 vs 1x3)
```

Matrices (行列: 単数形はmatrix)

▶データは基本的に行列・ベクトルで表現される.

レンジによる入力 (1)

```
>> B=[1:2:9]
B =
     1   3   5   7   9
```

コロンで区切る. 引数が3つの場合,
順に 初期値, 増分, 終端

レンジによる入力 (2)

```
>> C=[1:5]
C =
     1   2   3   4   5
```

引数が2つの場合, 増分を1として扱う

特殊な行列

単位行列 `eye(m,n)` クロネッカーのデルタ, δ_{ij}

単位行列 I (Identity matrix) から.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

全要素1の行列 `ones(m,n)`

$$\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

零行列 `zeros(m,n)`

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

乱数行列 `rand(m,n)` および `randn(m,n)`

各要素が
[0,1]の一様乱数

各要素が
平均0, 分散1の
正規分布となる乱数

いずれも引数を1つにする(すなわちnを省略する)と正方行列になる.

成分，行ベクトル，列ベクトルの操作(1)

A	1列	2列	3列
1行	1	2	3
2行	2	3	4

すべて
このAに対して
はじめて行う操作
と仮定して.

成分の選択

行 列

```
>> A(2,3)
ans = 4
>> A(1,2)
ans = 2
```

行ベクトルの選択

コロンで全成分選択

```
>> A(2,:)
ans =
   2 3 4
```

列ベクトルの選択

```
>> A(:,1)
ans =
   1
   2
```

部分行列の選択

```
>> A(1:2,1:2)
ans =
   1 2
   2 3
```

列ベクトルへ変換

```
>> A(:)
ans =
   1
   2
   2
   3
   3
   4
```

成分，行ベクトル，列ベクトルの操作(2)

A	1列	2列	3列
1行	1	2	3
2行	2	3	4

すべて
このAに対して
はじめて行う操作
と仮定して.

成分の代入

行 列

```
>> A(2,3)=5
A =
   1  2  3
   2  3  5
```

行ベクトルの代入

コロンで全成分選択

```
>> A(2,:)=[4,5,6]
A =
   1  2  3
   4  5  6
```

ベクトルの削除

```
>> A(:,2)=[]
A =
   1  3
   2  4
```

▸要素のみの削除はできない

列ベクトルの代入

コロンで全成分選択

```
>> A(:,1)=[3;4]
A =
   3  2  3
   4  3  4
```

成分，行ベクトル，列ベクトルの操作(3)

A	1列	2列	3列
1行	1	2	3
2行	2	3	4

すべて
このAに対して
はじめて行う操作
と仮定して.

範囲外の成分の追加

```
>> A(3,4) = 1
A =
   1   2   3   0
   2   3   4   0
   0   0   0   1
```

▸他をゼロ埋めして追加する

行ベクトルの追加

コロンで全成分選択

```
>> A(3,:)=[4,5,6]
A =
   1   2   3
   2   3   4
   4   5   6
```

列ベクトルの追加

コロンで全成分選択

```
>> A(:,5)=[3;4]
A =
   1   2   3   0   3
   2   3   4   0   4
```

▸連続していない場合は
他をゼロ埋めして追加する

行列用の関数

A	1列	2列	3列
1行	1	2	3
2行	2	3	4

size：行列のサイズを見る

```
>> size(A)
ans =
     2  3
```

◀行数・列数

応用

Aと同サイズで1埋めの行列

```
>> ones(size(A))
ans =
     1  1  1
     1  1  1
```

➥ 2 3

sum：行列の和を計算

```
>> sum(A)
ans =
     3  5  7
>> sum(A,2)
ans =
     6
     9
```

◀デフォルトは列方向の和

▲2をつけると行方向の和（1が列方向）

応用

行列成分の総和

```
>> sum(sum(A))
ans =
    15
>> sum(A(:))
ans =
    15
```

似た動作をする関数にmean（平均）がある

算術演算子(行列演算の場合1)

$A_{m\times n} = [a_{ij}]$, $B_{m\times n} = [b_{ij}]$

AとBは同じ行数・列数

＋

足し算

－

引き算

$A + B = [a_{ij} + b_{ij}]$

$A + n = [a_{ij} + n]$

要素ごとに和(差)を取る.

全要素にスカラー値を加える

スカラー(1×1行列)

$A_{l\times m} = [a_{ij}]$, $B_{m\times n} = [b_{ij}]$

Aの列数と
Bの行数が同じ

＊

掛け算

$A * B = [\Sigma\, a_{ik} * b_{kj}]$

$A * n = [n*a_{ij}]$

スカラー(1×1行列)

$A_{n\times n} = [a_{ij}]$

正方行列

＾

べき乗

$A \wedge 2 = [\Sigma a_{ik} * a_{kj}]$

算術演算子(行列演算の場合2)

$A_{m\times n} = [a_{ij}]$, $B_{m\times n} = [b_{ij}]$

AとBは同じ行数・列数
もしくは列ベクトルと行ベクトル

.*	./	.^
要素ごと掛け算	要素ごと割り算	要素ごとべき乗

$A .* B = [a_{ij} * b_{ij}]$
$A ./ B = [a_{ij} / b_{ij}]$
$A .^ 2 = [a_{ij} ^ 2]$
$A .^ B = [a_{ij} ^ b_{ij}]$

要素ごとに
積・商・冪をとる.

ピリオド記号により, 要素ごと(elementwise)計算を行う.
*和・差でもピリオド付きにすることができるが, 結果は同じ.

'	.'	.''
複素共役転置行列	転置行列	複素共役行列

Aが実行列ならば,
A.' = A'

この授業(というか実数を扱う限り)ではほぼ ' で事足りる.

関係演算

< <= > >= == ~=

小なり，以下，大なり，以上，等しい，等しくない

A

```
1  2  3
2  3  4
```

B

```
2  1  4
6  2  3
```

```
>> A > 2
ans =
   0  0  1
   0  1  1
```

```
>> A >= 2
ans =
   0  1  1
   1  1  1
```

```
>> A == 2
ans =
   0  1  0
   1  0  0
```

```
>> A > B
ans =
   0  1  0
   0  1  1
```

行列 同士

- 要素ごとに比較

Octaveの特殊な計算の例

A	1列	2列
1行	1	3
2行	2	4

B	1列	2列
1行	5	7
2行	6	8

A .* B

$$\begin{bmatrix} 1*5 & 3*7 \\ 2*6 & 4*8 \end{bmatrix} = \begin{bmatrix} 5 & 21 \\ 12 & 32 \end{bmatrix}$$

A+1

$$\begin{bmatrix} 1+1 & 3+1 \\ 2+1 & 4+1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$$

C	1列	2列
1行	1	2

D	1列
1行	5
2行	7

C .* D

$$\begin{bmatrix} 1*5 & 2*5 \\ 1*7 & 2*7 \end{bmatrix} = \begin{bmatrix} 5 & 10 \\ 10 & 14 \end{bmatrix}$$

C + D

$$\begin{bmatrix} 1+5 & 2+5 \\ 1+7 & 2+7 \end{bmatrix} = \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$$

Octaveの特殊な計算の例

A	1列	2列
1行	1	3
2行	2	4

B	1列	2列
1行	5	7
2行	6	8

$$A + C \quad \begin{bmatrix} 1+1 & 3+2 \\ 2+1 & 4+2 \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$$A + D \quad \begin{bmatrix} 1+5 & 3+5 \\ 2+7 & 4+7 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 9 & 11 \end{bmatrix}$$

C	1列	2列
1行	1	2

D	1列
1行	5
2行	7

$$A .* C \quad \begin{bmatrix} 1*1 & 3*2 \\ 2*1 & 4*2 \end{bmatrix} = \begin{bmatrix} 1 & 6 \\ 2 & 8 \end{bmatrix}$$

$$A .\hat{} C \quad \begin{bmatrix} 1\hat{}1 & 3\hat{}2 \\ 2\hat{}1 & 4\hat{}2 \end{bmatrix} = \begin{bmatrix} 1 & 9 \\ 2 & 16 \end{bmatrix}$$

以上のほかにもOctaveには「裏技」のような演算のルールがあるので，
（正確にはBroadcastingという）
いろいろ試しておくと，あとでスクリプトが簡単に書けるようになる．

逆行列関連（行列の除算）

inv(A)

Aの逆行列（A^{-1}）

◂ **Inv**ertible matrix　インヴォリュート関数ではない．

A/B = A * inv(B)

右除算

ただし，内部では，
逆行列を作ることなく計算している．

A¥B = inv(A) * B

左除算

日本語符号では¥だが，欧米では＼なのでわかりやすい
（どちらが分母側として潜り込んでいるかわかる，という意味）

特殊関数の引数を行列とした場合

基本的に要素ごと演算を行う

sin, cos, tan

$\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$

三角関数，逆三角関数

$\sin(A) = [\sin(a_{ij})]$
$\cos(A) = [\cos(a_{ij})]$
$\tan(A) = [\tan(a_{ij})]$

$\mathrm{asin}(A) = [\mathrm{asin}(a_{ij})]$
$\mathrm{acos}(A) = [\mathrm{acos}(a_{ij})]$
$\mathrm{atan}(A) = [\mathrm{atan}(a_{ij})]$

exp, log, $\log_{10}$

指数関数，対数関数

$\exp(A) = [\exp(a_{ij})]$
$\log(A) = [\log(a_{ij})]$
$\mathrm{log10}(A) = [\mathrm{log10}(a_{ij})]$

例1: 三角関数のグラフを描く


```
x = [0:1/1000:1];
y = sin(2*pi*x);
plot(x,y)
xlabel("X-axis");
ylabel("Y-axis")
```

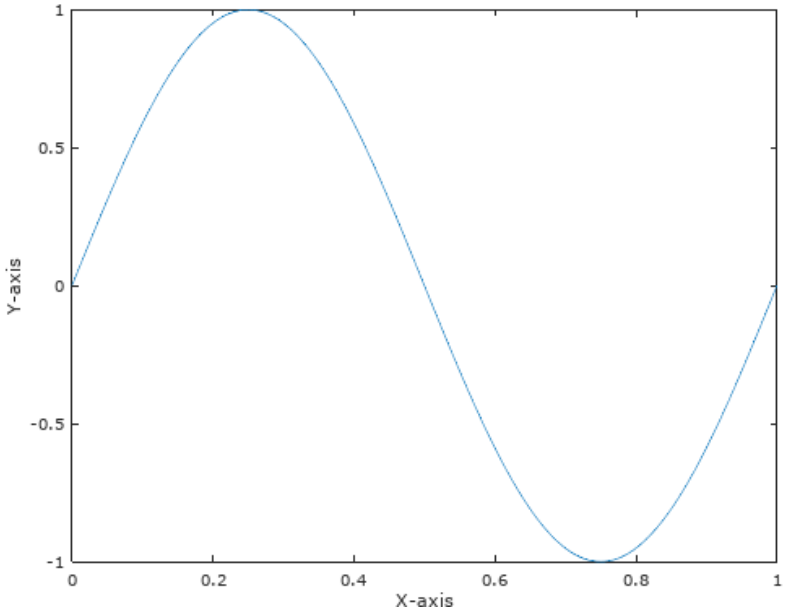
◀ 横軸区切り(とりあえず0から1まで区間)

▲行ベクトルで表現されている

ワークスペース

フィルタ

名前	クラス	次元	値	属性
x	double	1x1001	[0, 0.0010000, 0.0020000, 0.0030000, ...	
y	double	1x1001	[0, 0.0062831, 0.012566, 0.018848, 0....	



plot: プロットをする関数 ▸ グラフを書く関数

xlabel: x軸のラベル

ylabel: y軸のラベル

例2: Fourier級数のGibbs現象

$$x(t) = \frac{4}{\pi}\sum_{k=1}^{\infty} \sin\frac{(2\pi(2k-1)ft)}{2k-1}$$

矩形波のフーリエ級数展開

```
t = [0:1/1000:1];
f = 1;
k = [1:100]';
x = 4/pi * sin(2*pi*(2*k-1)*f.*t)./(2*k-1);
plot(t,sum(x))
```

時間区切り

周波数（好きに決められる）

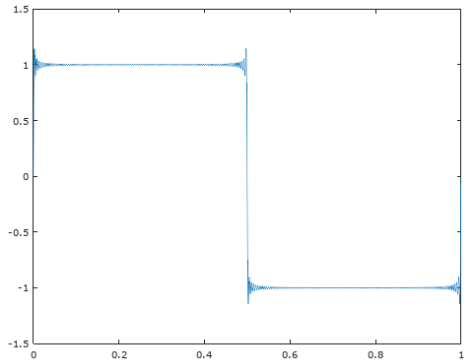
係数関係（転置に注意）

t = 0　0.001　0.002　0.003 … 1

k = 0
1
2
100

x（100×1000行列）

sum ▶（1×1000行列）

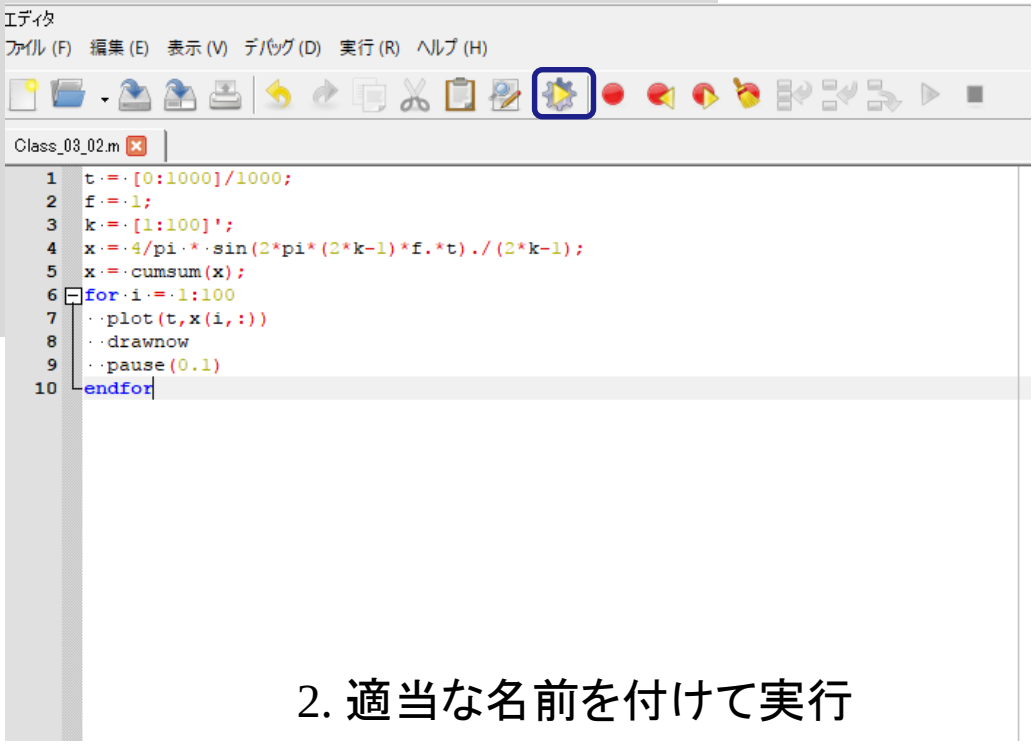


この100,000個の計算も0.01秒で終わる
（秒数はスペックによるけど）

例3: グラフのアニメーション化


```
t = [0:1000]/1000;
f = 1;
k = [1:100]';
x = 4/pi * sin(2*pi*(2*k-1)*f.*t)./(2*k-1);
x = cumsum(x);
for i = 1:100
  plot(t,x(i,:))
  ttl = sprintf("k = %d",i);
  title(ttl)
  drawnow
  pause(0.1)
endfor
```

1. エディタにコピーして



2. 適当な名前を付けて実行

プログラム解説


```
t = [0:1000]/1000;
f = 1;
k = [1:100]';
x = 4/pi * sin(2*pi*(2*k-1)*f.*t)./(2*k-1);
x = cumsum(x);
for i = 1:100
  plot(t,x(i,:))
  ttl = sprintf("k = %d",i);
  title(ttl)
  drawnow
  pause(0.1)
endfor
```

コロン二つを使った前のページの区切りと同じ.

cumsum: 出力行列の(i, j): 入力行列の (1, j) から (i, j) までの総和

%dに代入される

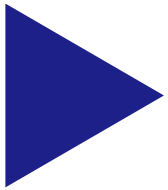
C言語のsprintf と似ている.

ただし指定子は%d（整数）と%f（浮動小数点）

グラフにタイトルを付ける関数

アドバイス: for文は(なるべく)使わない

for 文: ループして順次計算
行列演算: 並列計算が可能



(メモリが足りていれば)
行列演算にしたほうが速い

さらに，インデックスを用いた演算を駆使すれば if文も削れる

```
i=[1:100];
x=i(mod(i,2)==1);
```

条件式が真となる部分だけを抜き出し x に格納できる.

たとえば前回の課題だと，計算時間は1000回平均で下記の違いが生まれた.
for文: 5.2 ms (σ = 0.20 ms)
行列: 0.081 ms (σ = 0.0139 ms)

適当に乱数行列を作って「遊んで」慣れる

上から順に実行してみて
表示される数字の規則を見てみよう.
（最初の2行はただの乱数なので意味ないけど）

```
x=rand(3)
y=rand(3)
x>0.3
x(x>0.3)
y(x>0.3)
x>y
x(:)
```

```
x=rand(3,1)
y=rand(3,1)
x+y'
```

線形代数

実用上のベクトル

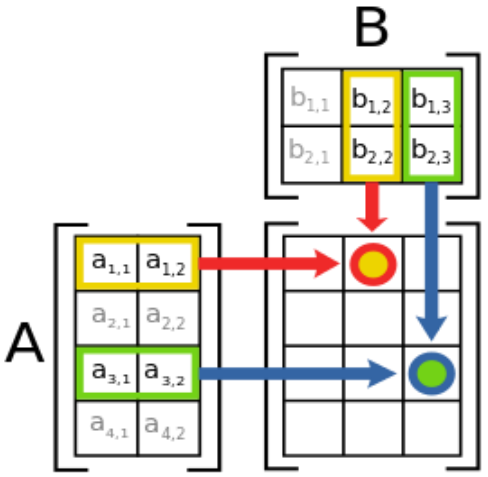
ベクトル ～ データのかたまり

図形的なベクトルももちろん正しいが,
本講義ではデータのかたまりとしてとらえたほうが適当.

内積 ～ 類似度

内積は二つのベクトルの類似度を表す指標と考えるとよい
（本当はそれを自身の大きさで割って規格化すべきだが）

行列 ～ 線形変換の係数を書き出したもの



https://en.wikipedia.org/wiki/Matrix_(mathematics)

行と列の奇妙に見える演算ルールも,「変換」という目的があってのもの
（しりとりルールによって行列の型が変化していくことを考えよう）

線形方程式の解

$$A = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & -1 \\ 2 & -1 & -3 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 0 \\ 3 \\ -1 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$A\boldsymbol{x} = \boldsymbol{b}$ の解を求める▼

```
>> A=[2,2,1;3,-1,3;2,-1,-3]
A =
    2 2 1
    3 -1 3
    2 -1 -3
>> b=[0;3;-1]
b =
    0
    3
    -1
```

```
>> A¥b
ans =
    0.19512
    -0.51220
    0.63415
>> inv(A)*b
ans =
    0.19512
    -0.51220
    0.63415
```

左除法

逆行列

左除法と逆行列，
いずれでも解けるが，
サイズが大きくなると
左除法の方が速い．

ベクトルのノルム

norm(x,p)

ベクトル x のp-ノルム

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{m} x_i^p}$$

```
>> x=[1,3,2];
>> norm(x)
ans = 3.7417
>> norm(x,2)
ans = 3.7417
>> norm(x,1)
ans = 6
>> norm(x,inf)
ans = 3
```

pの値を省略した場合は2
（デフォルト値が2）

$$\|x\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}$$

ユークリッドノルム

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|$$

$$\|x\|_{\mathrm{inf}} = \max(|x_i|)_{i=1,2,\ldots,n}$$

行列のノルム

norm(X,p)

行列Xのp-ノルム

```
>> x=[1,4,2;2,6,4];
>> norm(X,1)
ans = 10
>> norm(X,inf)
ans = 12
>> norm(x,“fro”)
ans = 8.7750
>> norm(X,2)
ans = 8.7601
```

$$\begin{bmatrix} 1 & 4 & 2 \\ 2 & 6 & 4 \end{bmatrix} \begin{matrix} \blacktriangleright 7 \\ \blacktriangleright 12 \end{matrix}$$

$$\begin{matrix} \blacktriangledown & \blacktriangledown & \blacktriangledown \\ 3 & 10 & 6 \end{matrix}$$

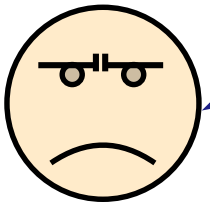
p = inf: 最大行和

p = 1: 最大列和

p = “fro”: フロベニウスノルム（Forbenius norm）

$$\|x\|_{\mathrm{fro}} = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}^2}$$

p = 2（デフォルト）: 最大特異値（A^*Aの最大固有値の平方根）



特異値？

10回目の授業で活躍します.

課題 3.1

3次元空間上の3点

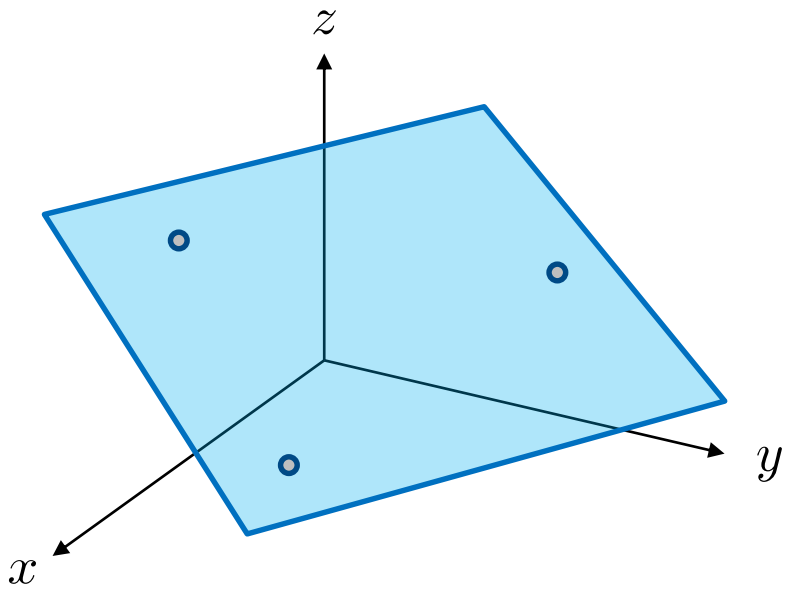
(x,y,z) = (24.0, -10.0, 12.0), (-30.0, -45.0, 0.0), (0.0, U, V) について考える.

ここでU, V は学籍番号の4桁の数字部分のそれぞれ前2桁，後2桁とする.

（例えばB7TB1357であれば U =13, V = 57）

(1) この3点を通る方程式を求めよ.（必須）

(2) グラフに平面を描け.（任意）



ヒント

(1)

原点 (0,0,0) を通らない平面の式:

$$ax + by + cz = 1$$

▸ 3点から未知係数に関する方程式が立つ

(2)

Octaveで3次元プロットの方法:

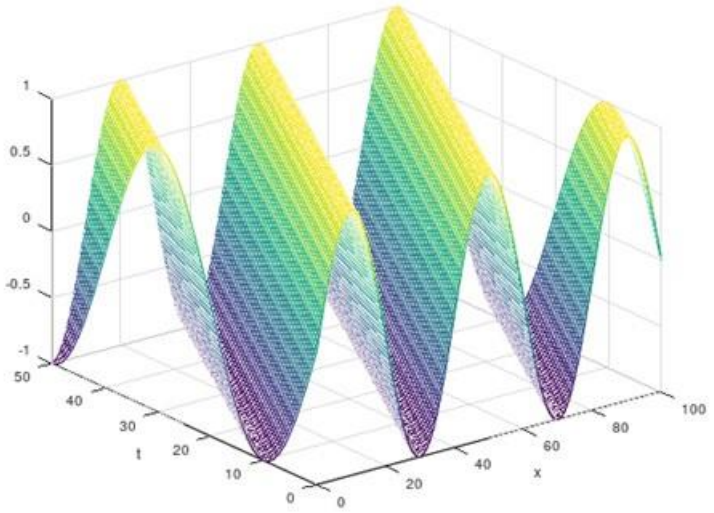
▸ mesh(x, y, z) 関数を使う.

サンプルコード

```
x=[0:100];
t=[0:50];
mesh(x,t,sin(pi/20*(x-t')))
xlabel('x')
ylabel('t')
```

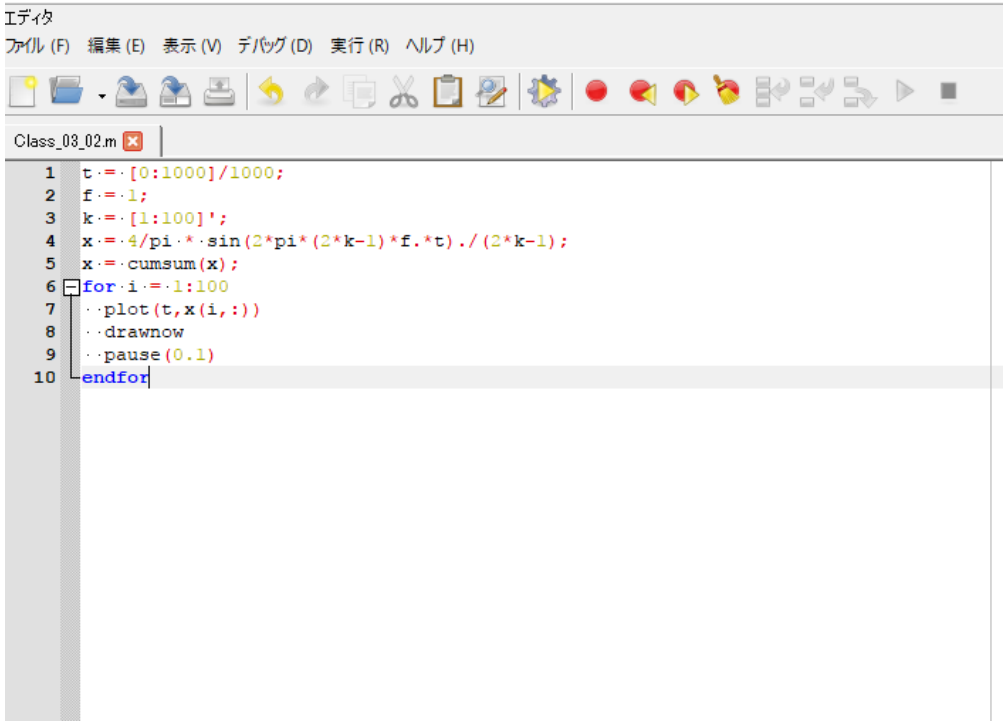
ブロードキャスティング機能により拡張される

zを陽とする3次元関数
（ここでは例として三角関数）



▸ 上記方程式が平面の方程式なので…

補足



- 提出はエディタで
  作成した.mファイルとし，
  ファイル名を
  B〇TB 〇 〇 〇 〇_03_01
  とする（すべて半角文字）．

- 提出前に実行可能かを確かめること

提出するファイルのきまり

教科書のサンプルコードを手本にして（場合によってはコピペをして）問題を解く

- 問題ない

中間のグレーゾーンをどうするか

友達のプログラムをコピーし名前を書き換えて提出する.

- 不正とします.（当たり前）

提出するファイルのきまり

教科書のサンプルコードを手本にして（場合によってはコピペをして）問題を解く

- 問題ない

友達に教えてもらいながら問題を解く

- 問題ない しかし下記に抵触するとまずい

友達に教えるためにプログラムをみせる/みてもらう

- 不正とします．（ちょっと厳しいけど，結局見分けがつかない．）

友達のプログラムをコピーし名前を書き換えて提出する．

- 不正とします．（当たり前）

去年の不正事例

- 変数の文字を変えただけ（x ▶ a，y ▶ b，z ▶ c）
- 無意味に改行をはさむ，行末のセミコロンを外したりつけたりする．
- 指定していない課題に取り組み，指定した課題に取り組まない．
  （他組間での不正：基本的に奇数回の授業は
  教科書と課題を少しずつ変えています．）

不正をすべて見つけることはできないが，
見つけた分は対処します．