数値計算：MATLAB

平井 慎一

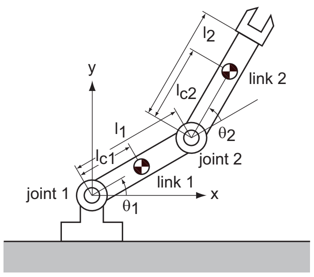
立命館大学 ロボティクス学科

講義の流れ

1. 行列とベクトル
2. グラフ
3. 常微分方程式
4. 最適化
5. パラメータの引き渡し
6. 乱数
7. まとめ

問題

2 リンク開ループ機構を関節 PID 制御で駆動する．
このときの動作をシミュレーションせよ．



問題

Step 1. 動作を表す運動方程式を導く（機構学）

Step 2. 運動方程式を数値的に解く

Step 3. 数値解をグラフや動画で表す
（ビジュアライゼーション）

Step 4. シミュレートした運動を分析する

問題

連立一次方程式

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{12} & H_{22} \end{bmatrix} \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

を解く．

$$\Downarrow$$

微分方程式

$$\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} \alpha_1(\theta_1, \theta_2, \omega_1, \omega_2) \\ \alpha_2(\theta_1, \theta_2, \omega_1, \omega_2) \end{bmatrix}$$

を解く．

ベクトルと行列

列ベクトル

```
x = [ 2; 3; -1 ];
```

行ベクトル

```
y = [ 2, 3, -1 ];
```

行列

```
A = [  4, -2,  1; ...
      -2,  5,  2; ...
      -2,  3,  2 ];
```


ベクトルと行列

記号 . . . は文が続くことを表す．

列ベクトル

```
x = [ 2; ...
      3; ...
     -1 ];
```

列ベクトル

```
x = [ 2; 3; -1 ];
```


ベクトルと行列

乗算

```
p = A*x;
q = y*A;
```

```
>> p

p =

     1
     9
     3

>>
```

ベクトルと行列

乗算

```
p = A*x;
q = y*A;
```

```
>> q

q =

     4     8     6

>>
```


行列の操作

```
>> A

A =

     4    -2     1
    -2     5     2
    -2     3     2

>> A(3,2)

ans =

     3
```


行列の操作

```
>> A
A =
     4    -2     1
    -2     5     2
    -2     3     2

>> A(3,2) = 6;

>> A
A =
     4    -2     1
    -2     5     2
    -2     6     2
```


行列の操作

```
>> A
A =

     4    -2     1
    -2     5     2
    -2     3     2

>> A(3,:)

ans =

    -2     3     2
```


行列の操作

```
>> A
A =

     4    -2     1
    -2     5     2
    -2     3     2

>> A(:,2)

ans =

    -2
     5
     3
```


行列の操作

```
>> A
A =
     4    -2     1
    -2     5     2
    -2     3     2

>> A(:,2) = [ 0; 2; 1 ];

>> A
A =
     4     0     1
    -2     2     2
    -2     1     2
```


行列の操作

```
>> A
A =
     4    -2     1
    -2     5     2
    -2     3     2

>> A(3,:) = [ 3, -5, -1 ];

>> A
A =
     4    -2     1
    -2     5     2
     3    -5    -1
```


行列の操作

```
>> A
A =
     4    -2     1
    -2     5     2
    -2     3     2

>> B = A([1,3],:);

>> B
B =
     4    -2     1
    -2     3     2
```

行列の操作

```
>> A
A =
     4    -2     1
    -2     5     2
    -2     3     2

>> C = A(:,[2,1]);

>> C
C =
    -2     4
     5    -2
     3    -2
```


基本行操作

`A(3,:) = 5*A(3,:);`	3 行目を 5 倍する
`A(1,:) = A(1,:) + 4*A(2,:);`	1 行目に 2 行目の 4 倍を加える
`A([3,1],:) = A([1,3],:);`	1 行目と 3 行目を交換する

連立一次方程式を解く

```
A = [  4, -2,  1; ...
      -2,  5,  2; ...
      -2,  3,  2 ];
p = [ 1; 9; 3 ];
```

連立一次方程式 $Ax = \boldsymbol{p}$ を解く

```
>> x = A\p;
>> x

x =
     2
     3
    -1

>> A*x
```


連立一次方程式を解く

スクリプトファイル linear.m

```
A = [  4, -2,  1; ...
      -2,  5,  2; ...
      -2,  3,  2 ];
p = [ 1; 9; 3 ];
x = A\p;
x
```

を作成し，実行せよ．

```
>> A*x
```

を実行し，解を確認せよ．

連立一次方程式を解く

- オペレータ \ は汎用であるが効率的ではない
- 係数行列が正定対称の場合は，コレスキー分解を使う
- 慣性行列は正定対称

コレスキー分解

正定対称行列 A は，上三角行列 U を用いて

$$A = U^\top U$$

と分解できる

$$A\boldsymbol{x} = \boldsymbol{p} \implies U^\top U\boldsymbol{x} = \boldsymbol{p} \implies \begin{cases} U^\top \boldsymbol{y} = \boldsymbol{p} \\ U\boldsymbol{x} = \boldsymbol{y} \end{cases}$$

コレスキー分解

プログラム Cholesky.m

```
fprintf('Cholesky decomposition\n');

A = [ 4, -2, -2; ...
     -2,  2,  0; ...
     -2,  0,  3 ];

U = chol(A);
U

U'*U
```


コレスキー分解

```
>> Cholesky
Cholesky decomposition

U =
     2    -1    -1
     0     1    -1
     0     0     1

ans =
     4    -2    -2
    -2     2     0
    -2     0     3
```


コレスキー分解

プログラム

```
p = [ 6; -4; 0 ];
y = U'\p;
x = U\y;
x
```

計算結果

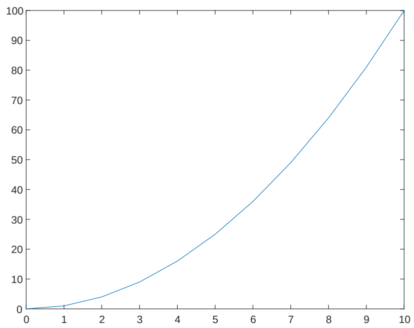
```
x =
     3
     1
     2
```

グラフ

```
>> x = [0:10]'
x =
      0
      1
      2
      3
     ...
>> f = x.*x
f =
      0
      1
      4
      9
     ...
```


グラフ

```
>> plot(x,f)
```



要素単位の演算

演算子 .∗ や ./ は，要素単位で乗算や除算を実行

$$\begin{bmatrix} 2 \\ 5 \\ -3 \end{bmatrix} .* \begin{bmatrix} 3 \\ -1 \\ -3 \end{bmatrix} = \begin{bmatrix} 6 \\ -5 \\ 9 \end{bmatrix}$$

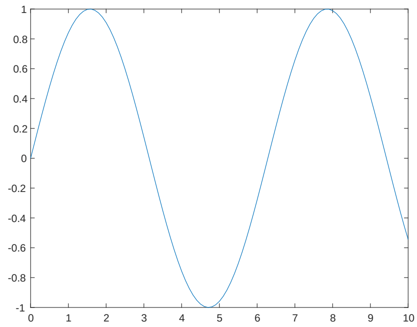
$$\begin{bmatrix} 6 \\ -5 \\ 1 \end{bmatrix} ./ \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ 1/2 \end{bmatrix}$$

グラフ

```
>> t = [0:0.1:10]'
t =
         0
    0.1000
    0.2000
    0.3000
     ...
>> x = sin(t)
x =
         0
    0.0998
    0.1987
    0.2955
     ...
```


グラフ

```
>> plot(t,x)
```



ベクトル化関数

関数 cos，sin，exp，log 等は，ベクトルを引数とすることができる．

$$\sin \begin{bmatrix} 0 \\ \pi/6 \\ \pi/3 \end{bmatrix} = \begin{bmatrix} \sin(0) \\ \sin(\pi/6) \\ \sin(\pi/3) \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \\ \sqrt{3}/2 \end{bmatrix}$$

$$\exp \begin{bmatrix} 0 \\ \log 2 \\ \log 3 \end{bmatrix} = \begin{bmatrix} \exp(0) \\ \exp(\log 2) \\ \exp(\log 3) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

グラフ

ファイル draw_graph.m

```
t = [0:0.1:10]';
x = sin(t);
plot(t,x);
title('グラフ');     % グラフの表題
xlabel('time');      % 横軸のラベル
ylabel('position'); % 縦軸のラベル
ylim([-1.5,1.5]);    % 縦軸の範囲
saveas(gcf,'draw_sine_graph.png'); % グラフの保存
```

ファイル draw_graph.m を実行すると，グラフを描き，
描いたグラフをファイルに保存する．

常微分方程式を数値的に解く

ファンデルポール (van der Pol) 方程式

$$\ddot{x} - 2(1 - x^2)\dot{x} + x = 0$$

$$\Downarrow$$

$$\begin{cases} \dot{x} = v \\ \dot{v} = 2(1 - x^2)v - x \end{cases}$$

$$\Downarrow$$

$$\boldsymbol{q} = \begin{bmatrix} x \\ v \end{bmatrix}, \qquad \dot{\boldsymbol{q}} = \boldsymbol{f}(t, \boldsymbol{q}) = \begin{bmatrix} v \\ 2(1 - x^2)v - x \end{bmatrix}$$

常微分方程式を数値的に解く

関数 $\boldsymbol{f}(t, \boldsymbol{q})$ を定義するファイル `van_der_Pol.m`
ファイルの名前 "van_der_Pol" と関数の名前 "van_der_Pol" を一致させる

```
function dotq = van_der_Pol (t, q)
    x = q(1);
    v = q(2);
    dotx = v;
    dotv = 2*(1-x^2)*v - x;
    dotq = [dotx; dotv];
end
```


常微分方程式を数値的に解く

関数ファイル `van_der_Pol.m` を作成せよ．

```
>> q = [ 2; 0]

>> van_der_Pol(0,q)
```

を実行せよ．作成した関数を用いることができる．

常微分方程式を数値的に解く

スクリプトプログラム `van_der_Pol_solve.m`

```
interval = 0.00:0.10:10.00;
qinit = [ 2.00; 0.00 ];
[time, q] = ode45(@van_der_Pol, interval, qinit);
```

ファイル `van_der_Pol_solve.m` を作成し，実行せよ．

常微分方程式を数値的に解く

時刻 t と変数 x の関係をグラフで表す

```
plot(time, q(:,1), '-');
```

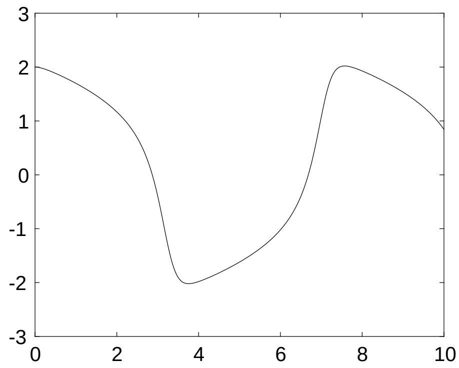
時刻 t と変数 v の関係をグラフで表す

```
plot(time, q(:,2), '-');
```

'-'	実線
'--'	破線
'-.'	一点破線
':'	点線

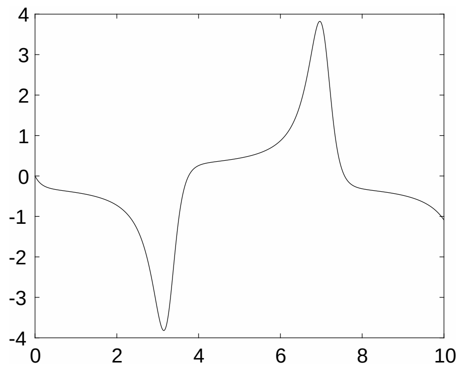
常微分方程式を数値的に解く

時刻 t と変数 x のグラフ



常微分方程式を数値的に解く

時刻 t と変数 v のグラフ



最適化

ローゼンブロック関数 (Rosenbrock function) の最小化

$$\text{minimize} \quad f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

ファイル `Rosenbrock.m`

```
function f = Rosenbrock( x )
    x1 = x(1); x2 = x(2);
    f = 100*(x2 - x1^2)^2 + (1 - x1)^2;
end
```


最適化

ファイル `Rosenbrock_minimize.m`

```
xinit = [ -1.2; 1.0 ];
[xmin, fmin] = fminsearch(@Rosenbrock, xinit);
xmin
fmin
```

実行結果

```
>> Rosenbrock_minimize
xmin =
    1.0000
    1.0000
fmin =
   8.1777e-10
```

パラメータを有する常微分方程式

微分方程式

$$\ddot{x} + b\dot{x} + 9x = 0$$

b はパラメータ

$$\Downarrow$$

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -bv - 9x \end{aligned}$$

大域変数

関数

```
    function dotq = damped_vibration_global (t, q)
        global b;
        x = q(1); v = q(2);
        dotx = v; dotv = -b*v - 9*x;
        dotq = [dotx; dotv];
    end
```

プログラム

```
global b;
interval = [0,10];
qinit = [2.00;0.00];
b = 1.00;
[time,q] = ode45(@damped_vibration_global,interval,qinit)
```


入れ子関数

時刻，状態変数ベクトル，パラメータを引数とする関数

```
    function dotq = damped_vibration_param (t, q, b)
        x = q(1); v = q(2);
        dotx = v; dotv = -b*v - 9*x;
        dotq = [dotx; dotv];
    end
```

プログラム

```
interval = [0,10];
qinit = [2.00;0.00];
b = 1.00;
damped_vibration = @(t,q) damped_vibration_param (t,q,b);
[time,q] = ode45(damped_vibration,interval,qinit);
```


大域変数 vs 入れ子関数

大域変数

プログラムが単純
大域変数が他の変数と重複する恐れがある

入れ子関数

プログラムがやや複雑
パラメータの値が変わるたびに，関数定義を再実行する必要
他の変数と重複しない

一様乱数

区間 $(0, 1)$ 内の一様乱数

```
    rng('shuffle', 'twister');
    for k=1:10
        x = rand;
        s = num2str(x);
        disp(s);
    end
```

記号'shuffle'：プログラムを実行するたびに異なる乱数を生成

一様乱数

区間 $(0, 1)$ 内の一様乱数

```
    rng(0, 'twister');
    for k=1:10
        x = rand;
        s = num2str(x);
        disp(s);
    end
```

種の指定：プログラムを実行するたびに同じ乱数を生成

dice.m

```
function k = dice()
%   simulating a dice
    x = rand;
    if x < 1/6.00         k = 1;
    elseif x < 2/6.00     k = 2;
    elseif x < 3/6.00     k = 3;
    elseif x < 4/6.00     k = 4;
    elseif x < 5/6.00     k = 5;
    else                  k = 6;
    end
end
```


dice_run.m

```
for i=1:10
    s = [];
    for j=1:10
        k = dice();
        s = [s, ' ', num2str(k)];
    end
    disp(s);
end
```

dice_run.m

```
>> dice_run
 2 4 6 5 6 3 3 4 2 5
 4 4 2 1 3 3 5 5 5 1
 1 4 2 6 6 2 5 6 4 6
 6 4 5 4 1 3 4 4 3 6
 5 6 3 4 6 6 5 2 4 1
 3 5 6 5 3 5 3 6 6 6
 3 3 2 5 6 6 4 4 1 6
 3 2 6 5 6 2 5 4 1 3
 2 5 2 6 5 3 3 5 6 4
 4 2 3 5 6 5 1 5 3 3
>> dice_run
 1 5 2 2 3 3 4 4 3 3
 4 4 6 5 3 5 1 1 1 1
 2 2 1 4 1 1 4 6 6 4
 6 4 4 2 3 3 1 6 1 3
```


まとめ

MATLAB による数値計算

- ベクトルと行列の計算
- 連立一次方程式を解く
- 常微分方程式を数値的に解く
- 最適化
- パラメータの引き渡し
- 乱数

レポート (MATLAB Grader)

MATLAB grader 「MATLAB の使い方の練習」
締切：5 月 13 日（月曜）01:00 AM

1. 「スクリプト」欄に解答を入力
2. 「スクリプトを実行」をクリックし，解答を実行
3. 「出力」欄で解答を確認
4. 「提出」をクリックし，解答を提出
（解答の提出回数の制限に注意）

https://jp.mathworks.com/ の「サイト内検索」で
「単位行列」や「ゼロ行列」を検索せよ．