

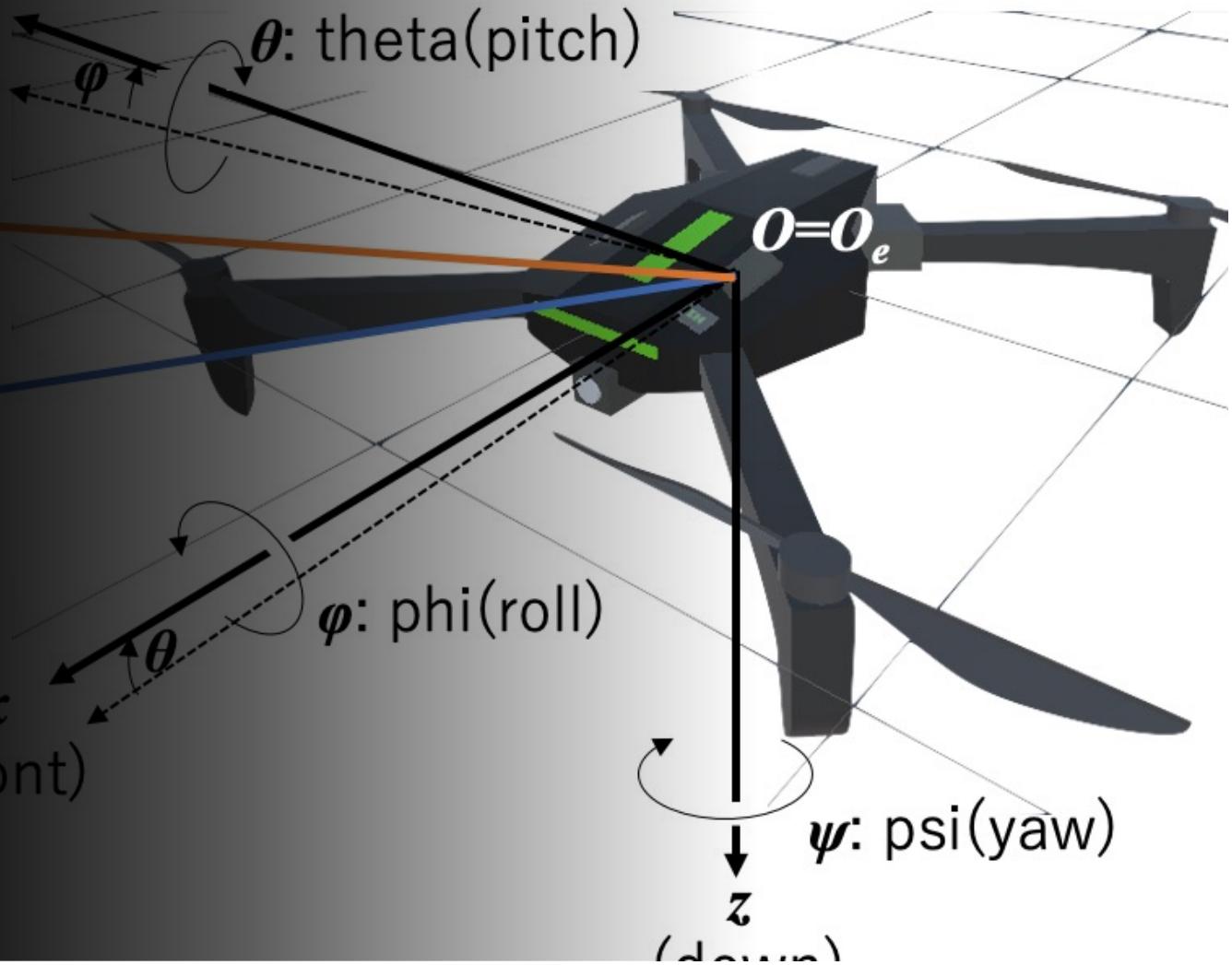
# ドローンの 数学・物理・ 制御基礎

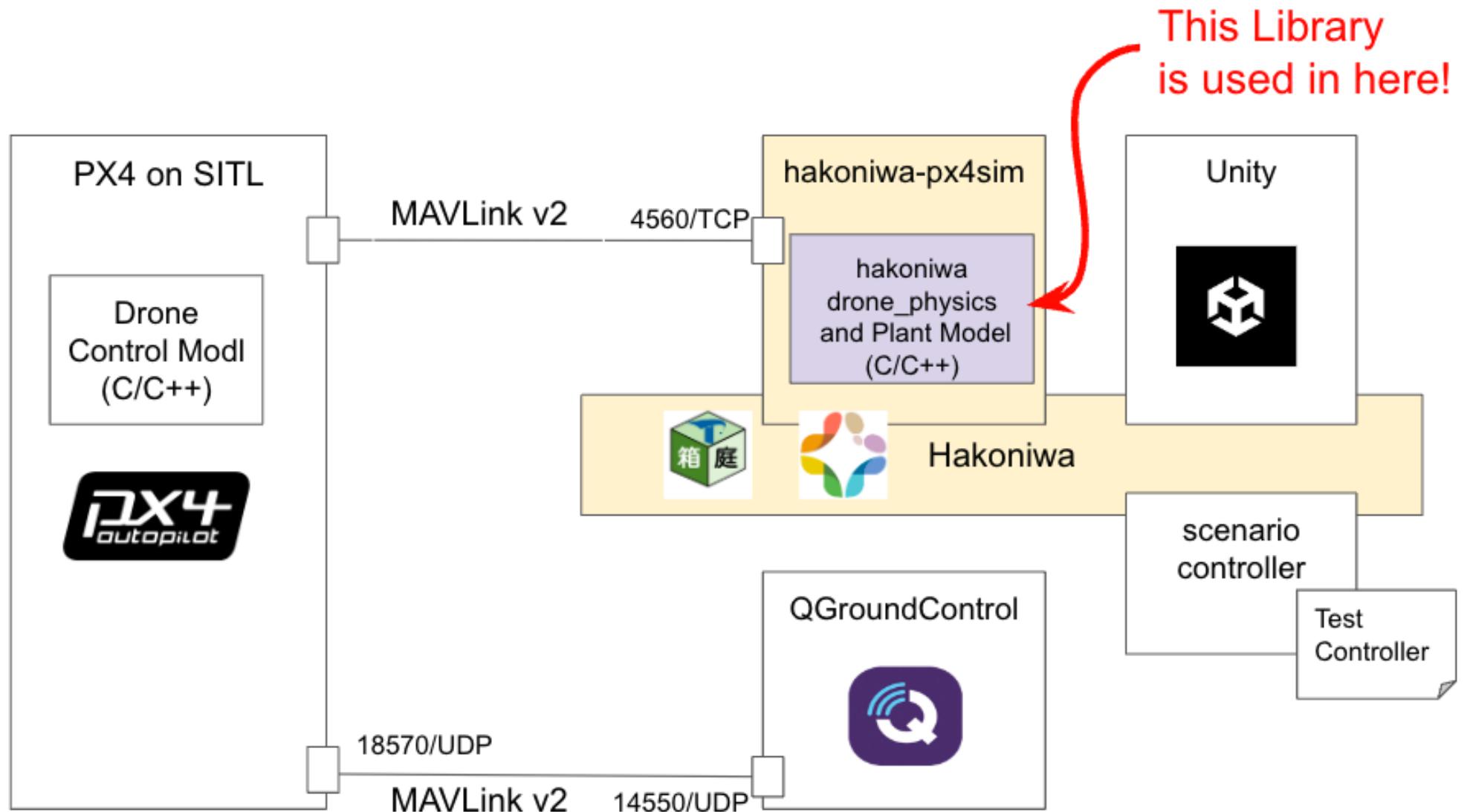
2024/2/15  
箱庭ラボ  
平鍋健児



(right)

*y*





# 本日のお品書き

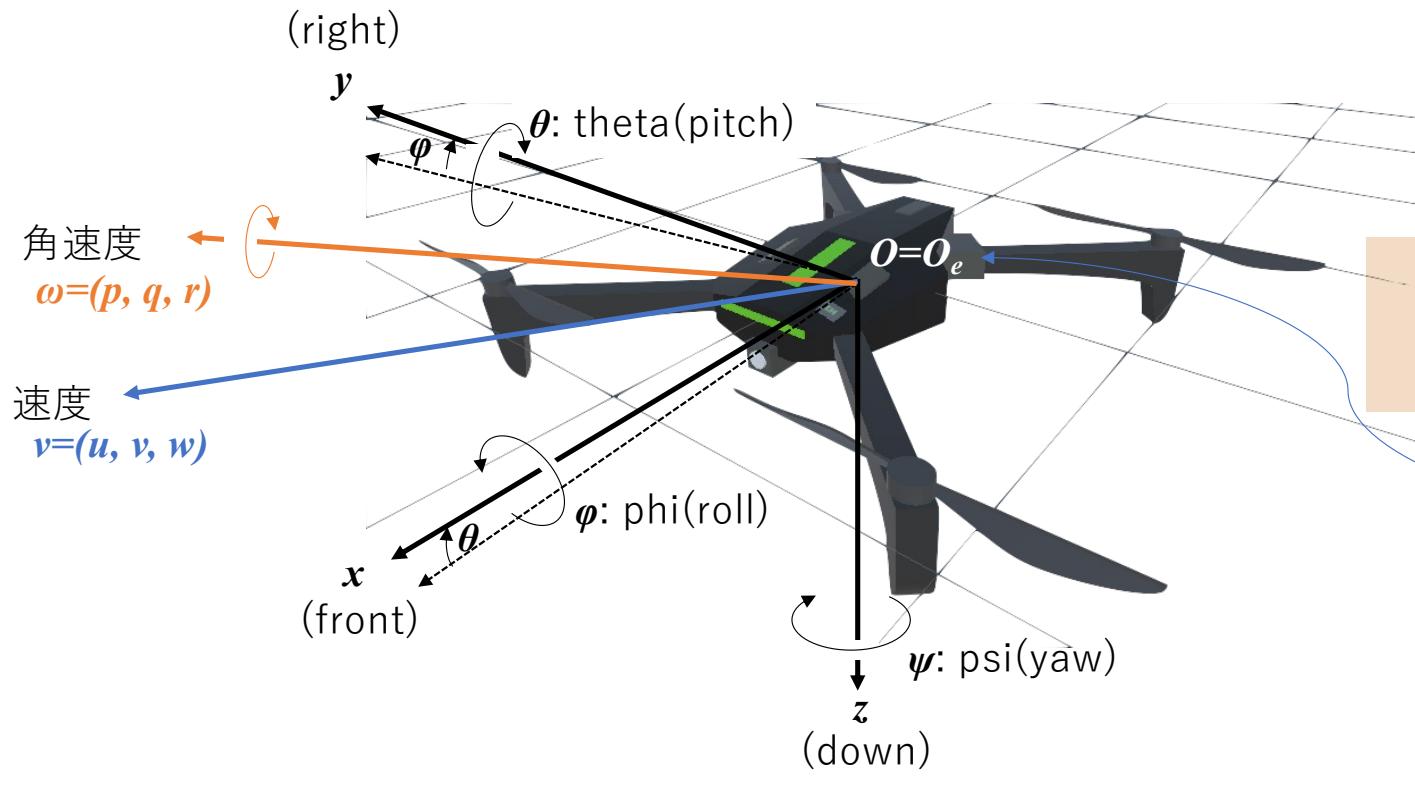
- 座標系の話（数学）
- 運動方程式の話（物理）
- 機体の物理モデル全体
- プラントとコントローラの話（制御）
- ソフトウェア設計の話
- その他雑感など. . .

詳しい説明はこちらから：

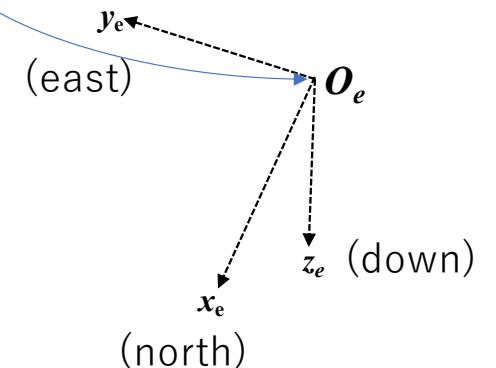
[https://github.com/toppers/hakoniwa-px4sim/blob/main/drone\\_physics/README-ja.md](https://github.com/toppers/hakoniwa-px4sim/blob/main/drone_physics/README-ja.md)

# Body Frame 機体座標

FRD



# Ground Frame 地上座標 NED



回転方向はすべて軸+に対して右ネジ  
地上枠を機体枠へ、 $\psi, \theta, \phi$ の順に回して重ねる=姿勢

原点を Body Frame の  
原点に重ねる ("Vehicle Carried NED")

# 座標変換

速度, 加速度の変換

(方向余弦行列 DCM: Direction Cosine Matrix)

機体座標系  $v = (u, v, w)^T$  から地上座標系  $v_e = (u_e, v_e, w_e)^T$  への変換行列は以下のようになります。加速度も同様です。

$$\text{Ground} \begin{bmatrix} u_e \\ v_e \\ w_e \end{bmatrix} = \begin{bmatrix} \cos \theta \cos \psi & \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi \\ \cos \theta \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \text{Body}$$

関数名は, `ground_vector_from_body`。逆変換は, `body_vector_from_ground`。

角速度 (回転) とオイラー角変化率の変換

$$R_z(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, \quad R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}$$

機体座標系の角速度  $(p, q, r)^T$  からオイラー角変化率  $(\dot{\phi}, \dot{\theta}, \dot{\psi})^T$  への変換行列は以下のようになります。

$$\text{Ground} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \text{Body}$$

関数名は, `euler_rate_from_body_angular_velocity`。逆変換は, `body_angular_velocity_from_euler_rate`。

# 運動方程式

Ground Frame

$$m\dot{v} = F$$

ニュートンの運動方程式

$$I\dot{\omega} = \tau$$

オイラーの運動方程式（このまま解くのは複雑）

Body Frame

$$m\dot{v} + [\omega \times mv] = F$$

位置が入っていないので、  
原点を一致させて考える。  
(Vehicle Carried NED)  
並進の慣性力は扱わない

$$I\dot{\omega} + [\omega \times I\omega] = \tau$$

ジャイロ効果

コリオリの力。  
遠心力は重心に原点を取っているので発生しない。

# 運動方程式

## Body Frame

### 速度, 加速度(並進)

ニュートンの運動方程式

$$\begin{aligned}\dot{u} &= -g \sin \theta - (qw - rv) - \frac{d}{m} u \\ \dot{v} &= g \cos \theta \sin \phi - (ru - pw) - \frac{d}{m} v \\ \dot{w} &= -\frac{T}{m} + g \cos \theta \cos \phi - (pv - qu) - \frac{d}{m} w\end{aligned}$$

関数名は, `acceleration_in_body_frame` .

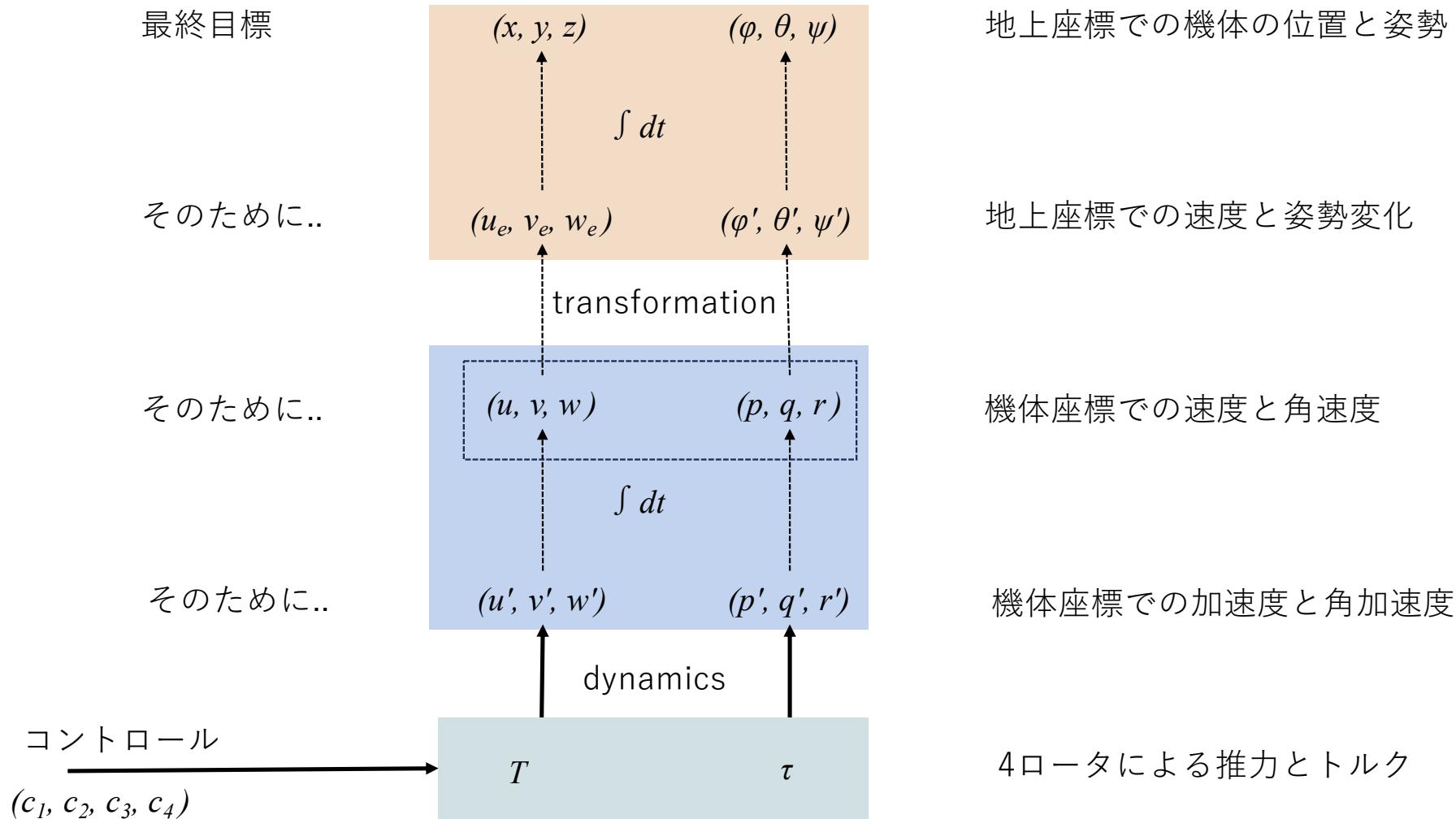
### 角速度, 角加速度(回転)

オイラーの運動方程式

$$\begin{aligned}\dot{p} &= (\tau_\phi - qr(I_{zz} - I_{yy}))/I_{xx} \\ \dot{q} &= (\tau_\theta - rp(I_{xx} - I_{zz}))/I_{yy} \\ \dot{r} &= (\tau_\psi - pq(I_{yy} - I_{xx}))/I_{zz}\end{aligned}$$

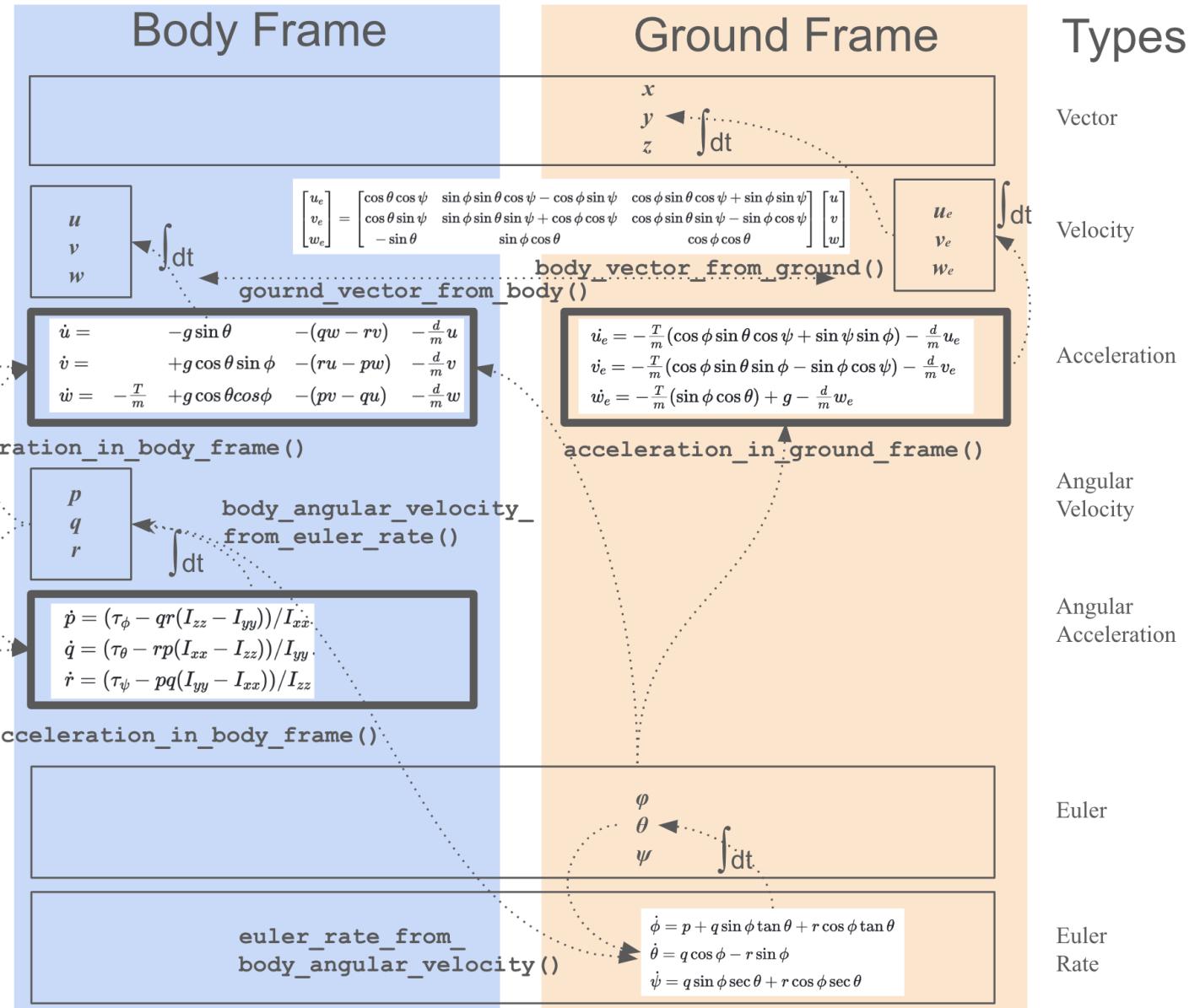
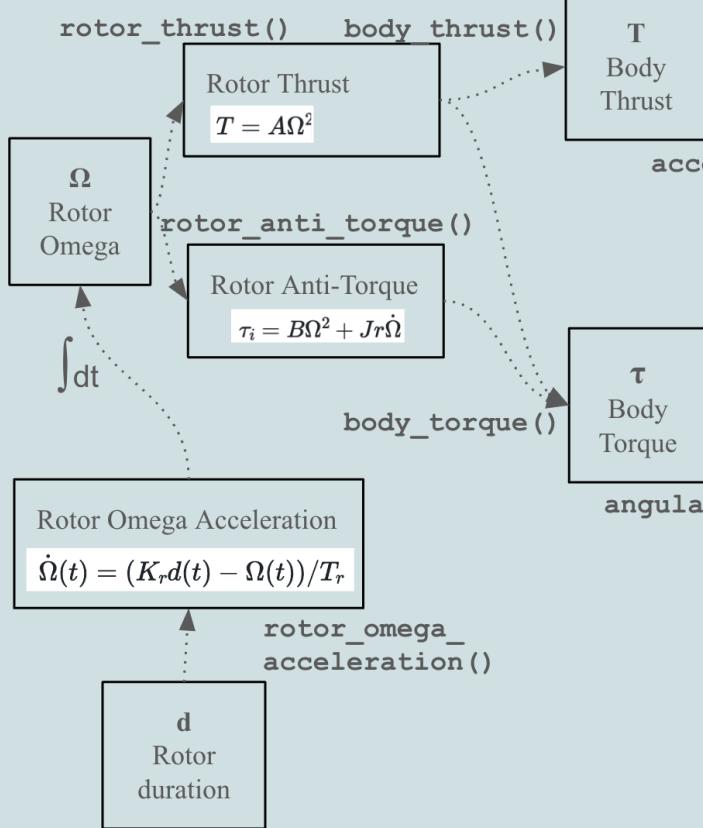
関数名は, `angular_acceleration_in_body_frame` .

# プラントのモデル



Note: 4入力6自由度なので、全部自由にはならない。  
例：ヨー角は高度を下げずに変えられるが、ピッチ角  
を変えると高度が一旦下がる。

# Rotor Physics



# 制御モデル(Plant+Controller)として

- P=箱庭ドローン, C= PX4のPID 制御



Pの並進の  
動力学

$$\begin{aligned}\dot{u} &= -g \sin \theta & -(qw - rv) & -\frac{d}{m} u \\ \dot{v} &= +g \cos \theta \sin \phi & -(ru - pw) & -\frac{d}{m} v \\ \dot{w} &= -\frac{T}{m} & +g \cos \theta \cos \phi & -(pv - qu) & -\frac{d}{m} w\end{aligned}$$

$$w = dz/dt$$

$$(\phi, \theta, \psi) = 0$$

$$(p, q, r) = 0$$

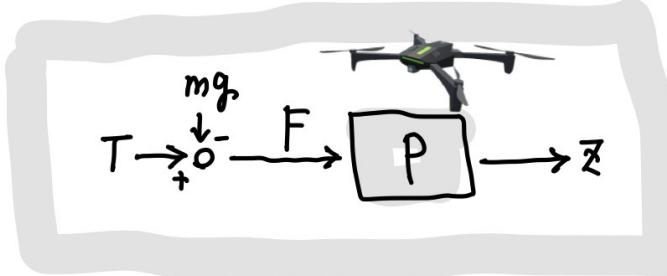
zの符号反転

簡略仕様

$$\ddot{z} = \frac{T(t)}{m} - g - \frac{d}{m} \dot{z}$$

(...回転の方は手に追えない)

# Plantのみ の応答 (開ループ)



$T$ ,  $z$ を上方とし、 $1\text{m/s}$ に標準化

$$m\ddot{z} = F = T(t) - mg - d\dot{z}$$

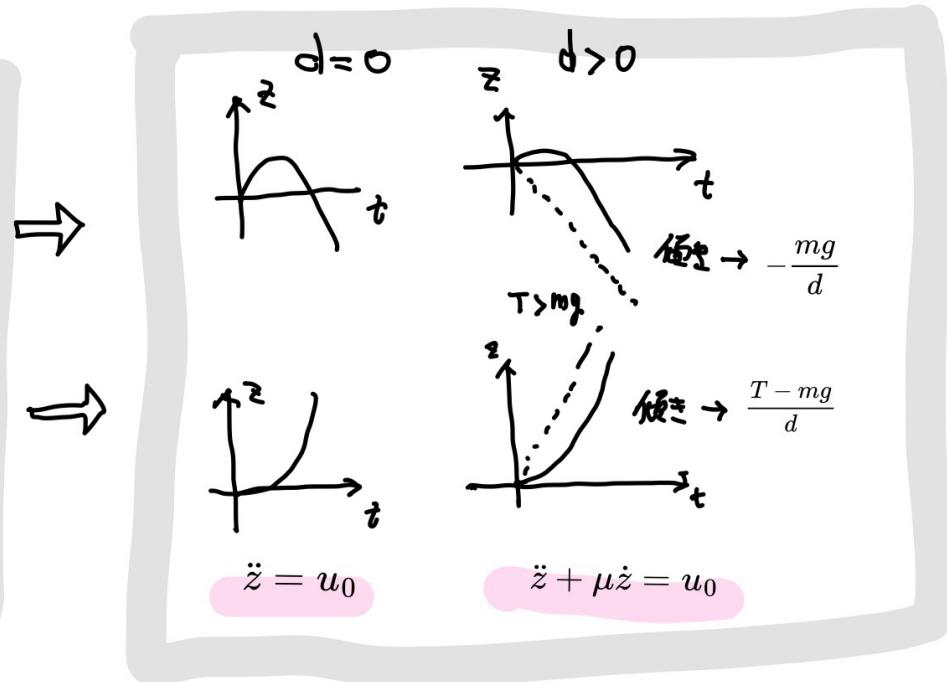
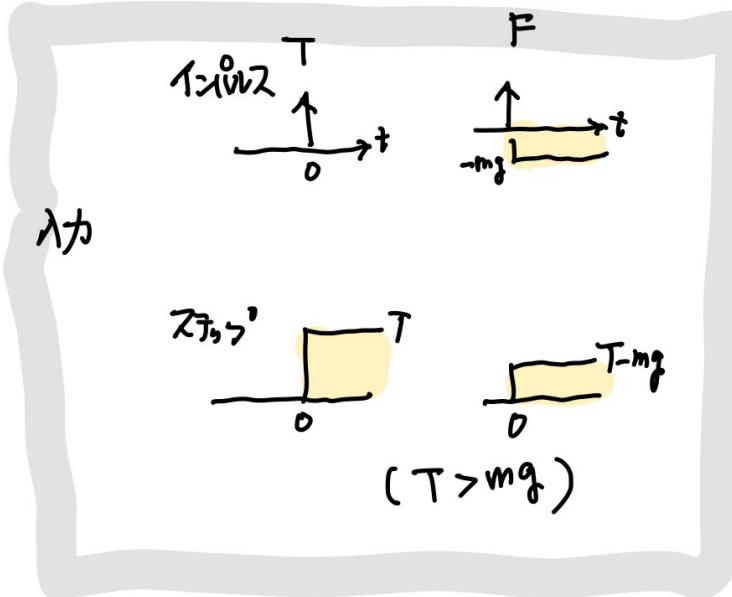
$$\ddot{z} = \frac{T(t)}{m} - g - \frac{d}{m}\dot{z}$$

$$\ddot{z} + \mu\dot{z} = u_0$$

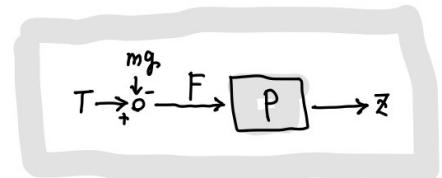
$$G(s) = \frac{u_0}{s(s + \mu)}$$

$$\begin{cases} u_0 = \frac{T}{m} - g \\ \mu = \frac{d}{m} \end{cases}$$

これを計算



2次不確定



$$m\ddot{z} = F = T(t) - mg - d\dot{z}$$

$$\ddot{z} = \frac{T(t)}{m} - g - \frac{d}{m}\dot{z}$$

$$\ddot{z} + \mu\dot{z} = u_0$$

RC2計算

$u(t) = \frac{T(t)}{m} - g$  は、 $T$  たゞ、 $g$  から  $z$  にともなう（全運動）

ゆえに  $u(t)$  は  $\lambda$  とし  $\lambda$  を扱う。

$$\ddot{z} + \mu\dot{z} = u_0, \quad z(0) = z'(0) = 0$$

ラプラス変換を用いる方法

$$\mathcal{L}(z) \rightarrow (s^2 + \mu s) Z = \frac{u_0}{s}$$

$$Z = \frac{u_0}{s^2(s+\mu)}$$

部分分式展開

$$Z = u_0 \left[ \frac{1}{\mu^2} \left( \frac{1}{s+\mu} \right) + \frac{1}{\mu} \left( \frac{1}{s^2} \right) - \frac{1}{\mu^2} \left( \frac{1}{s} \right) \right]$$

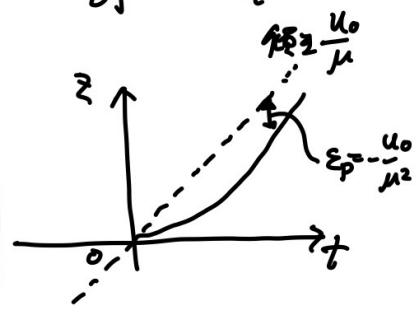
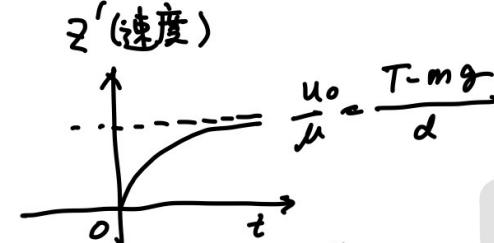
$$z(t) = u_0 \left( \frac{1}{\mu^2} e^{-\mu t} + \frac{1}{\mu} t - \frac{1}{\mu^2} \right)$$

$$z(t) = \frac{u_0}{\mu} \left\{ t - (1 - e^{-\mu t})/\mu \right\}$$

$$z'(t) = \frac{u_0}{\mu} (1 - e^{-\mu t})$$

$$\lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s) = \lim_{s \rightarrow 0} \frac{u_0}{s+\mu} = \infty$$

$$\lim_{t \rightarrow \infty} f'(t) = \lim_{s \rightarrow 0} s^2 F(s) = \lim_{s \rightarrow 0} \frac{u_0}{s^2 + \mu s} = \frac{u_0}{\mu}$$



	$t=0$	$t \rightarrow \infty$
$z(t)$	0	$\infty$
$z'(t)$	0	$kg/\mu = \frac{T-mg}{d}$

$$\Sigma_p(t) = -\frac{u_0}{\mu^2} (1 - e^{-\mu t})$$

$$\lim_{t \rightarrow \infty} \Sigma_p(t) = -\frac{u_0}{\mu^2} = -\frac{T-mg}{m\mu^2}$$

## 手計算×モード

### 時間領域で解く方法

#### 特性方程式

$$P(\lambda) = \lambda^2 + \mu\lambda = \lambda(\lambda + \mu)$$

$$\text{固有値}: (0, -\mu), \text{ 特解}: z_p = \frac{u_0}{\mu} t$$

$$\text{有次解} \quad Z_n = C_1 + C_2 e^{-\mu t}$$

$$\text{-一般解} \quad Z = Z_p + Z_n = \frac{u_0}{\mu} t + C_1 + C_2 e^{-\mu t}$$

$$Z'(t) = \frac{u_0}{\mu} - \mu C_2 e^{-\mu t}$$

$$\text{初期値} \quad Z(0) = C_1 + C_2 = 0, \quad Z'(0) = \frac{u_0}{\mu} - \mu C_2 = 0$$

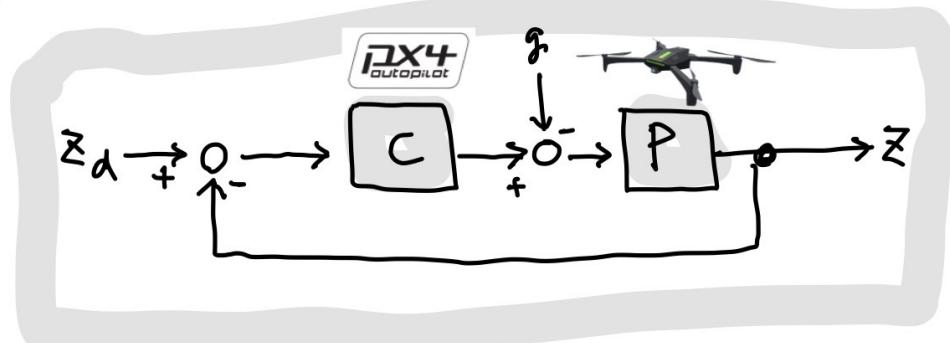
$$\therefore C_2 = \frac{u_0}{\mu^2}, \quad C_1 = -\frac{u_0}{\mu^2}$$

$$\therefore Z(t) = \frac{u_0}{\mu^2} (1 - e^{-\mu t} + \mu t)$$

$$Z'(t) = \frac{u_0}{\mu} (1 - e^{-\mu t})$$

# フィードバック（閉ループ）応答

$z_d$ : 目標値  
(desired),  $C = K$  (定数ゲイン: P制御)



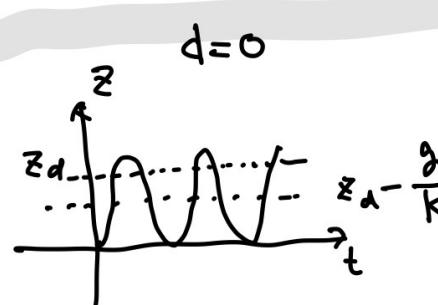
$$\ddot{z} = K(z_d - z) - g - \frac{d}{m}z$$

$$\ddot{z} + \frac{d}{m}\dot{z} + Kz = Kz_d - g$$

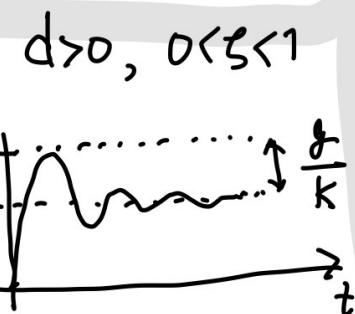
$$\ddot{z} + 2\zeta\omega_n\dot{z} + \omega_n^2 z = u_0$$

CC計算

$$G(s) = \frac{u_0}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad \left\{ \begin{array}{l} u_0 = Kz_d - g \\ \omega_n = \sqrt{K} \\ \zeta = \frac{d}{2m\omega_n} = \frac{d}{2m\sqrt{K}} \end{array} \right.$$

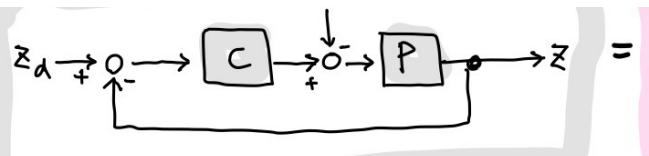


$$\ddot{z} + \omega_n^2 z = u_0$$



$$\ddot{z} + 2\zeta\omega_n\dot{z} + \omega_n^2 z = u_0$$

2次振動



$$\ddot{z} + 2\zeta\omega_n\dot{z} + \omega_n^2 z = u_0$$

$$\left\{ \begin{array}{l} u_0 = Kz_d - g \\ \omega_n = \sqrt{K} \\ \zeta = \frac{d}{2m\omega_n} = \frac{d}{2m\sqrt{K}} \end{array} \right.$$

$$C = K\alpha \zeta z.$$

$$u_0 = Kz_d - g \quad z_d \text{ は目標位置 } u_0 \text{ は実際の位置 } \lambda \text{ は減衰係数} \quad (0 < \zeta < 1)$$

$$\ddot{z} + 2\zeta\omega_n\dot{z} + \omega_n^2 z = u_0, \quad z(0) = z'(0) = 0$$

7. 位置変換  $\zeta = 2$ .

$$(s^2 + 2\zeta\omega_n s + \omega_n^2) \ddot{z} = \frac{u_0}{s}$$

$$z = \frac{u_0}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \quad (\text{3次系})$$

部分  
分数

$$z = \frac{u_0}{\omega_n^2} \left\{ \frac{1}{s} - \frac{s + 2\zeta\omega_n}{s^2 + 2\zeta\omega_n s + \omega_n^2} \right\} \quad (\alpha = \sqrt{1 - \zeta^2})$$

$$z = \frac{u_0}{\omega_n^2} \left\{ \frac{1}{s} - \frac{(s + \zeta\omega_n) + i\omega_n}{(s + \zeta\omega_n)^2 + \omega_n^2 \alpha^2} \right\} \quad \varphi = \tan^{-1}(\alpha/\zeta)$$

$$z(t) = \frac{u_0}{\omega_n^2} \left[ 1 - e^{-\zeta\omega_n t} \{ \cos(\omega_n \alpha t) + \frac{i}{\alpha} \sin(\omega_n \alpha t) \} \right]$$

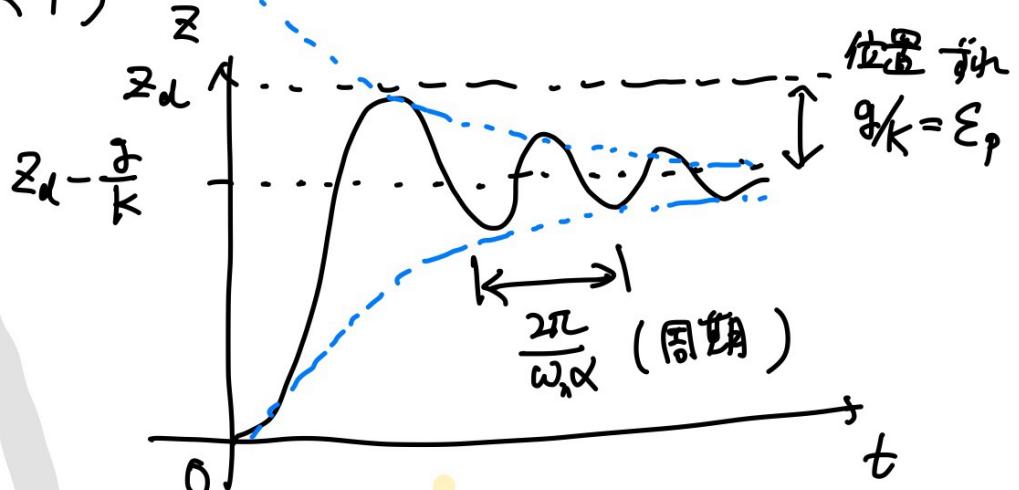
$$z(t) = \frac{u_0}{\omega_n^2} \left[ 1 - \frac{1}{\alpha} e^{-\zeta\omega_n t} \sin(\omega_n \alpha t + \varphi) \right] \quad (\text{位相形式 } \varphi)$$

$$z'(t) = \frac{u_0}{\omega_n} e^{-\zeta\omega_n t} \left\{ \frac{1}{\alpha} \sin(\omega_n \alpha t) \right\} \quad (\text{減衰振動の計算式})$$

$$\alpha = \sqrt{1 - \zeta^2} = \sqrt{1 - (d/2m\sqrt{K})^2}$$

$$\omega_n \alpha = \sqrt{\omega_n^2 - (d/2m)^2}$$

角周波数



$$t=0 \quad t \rightarrow \infty$$

$$z(t) \quad 0$$

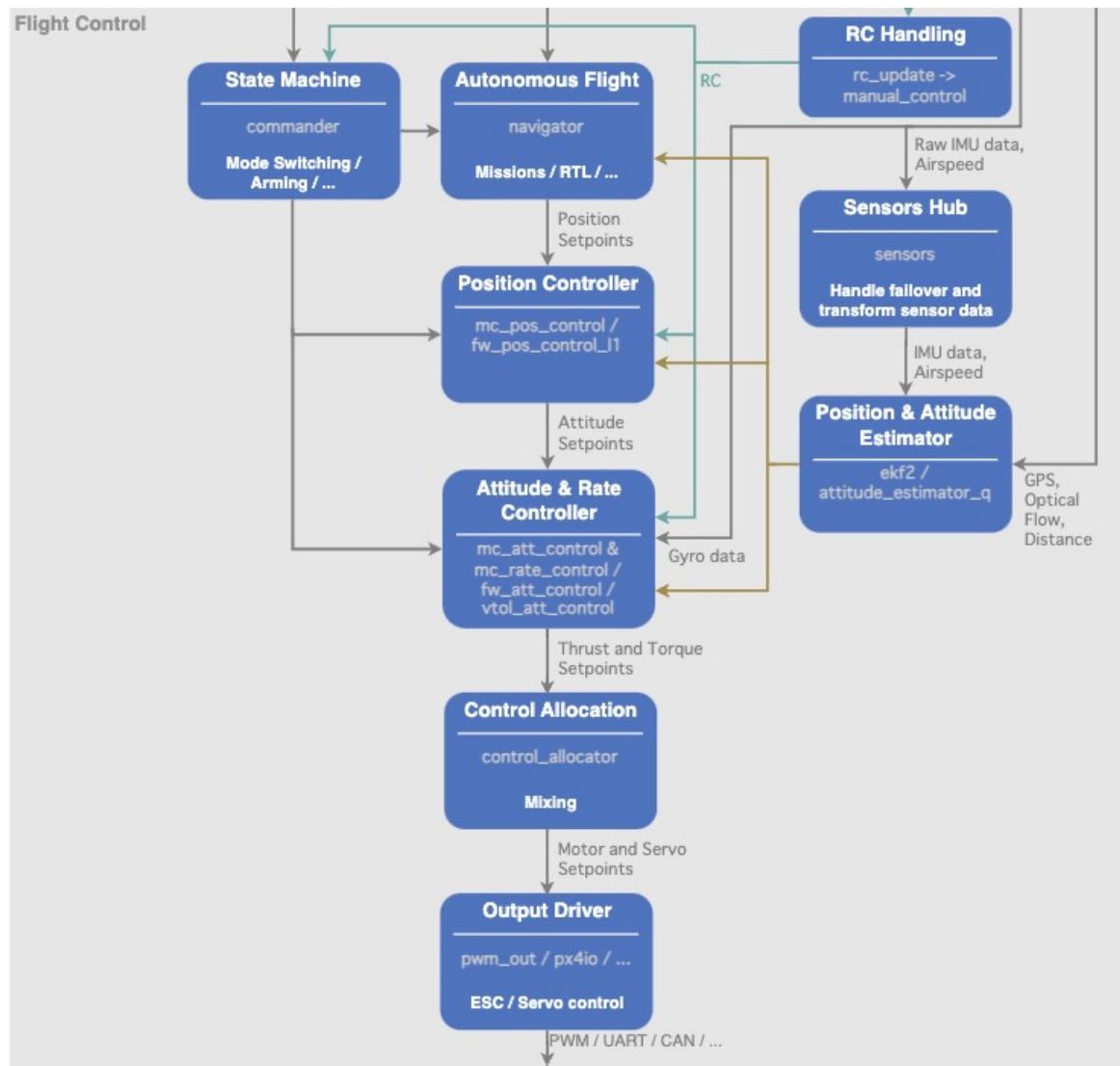
$$0$$

$$u_0/\omega_n^2 = z_d - g/k \quad (\text{定常偏差})$$

計算式

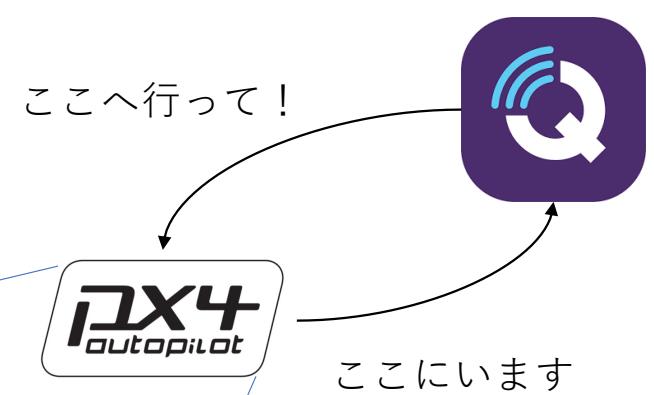
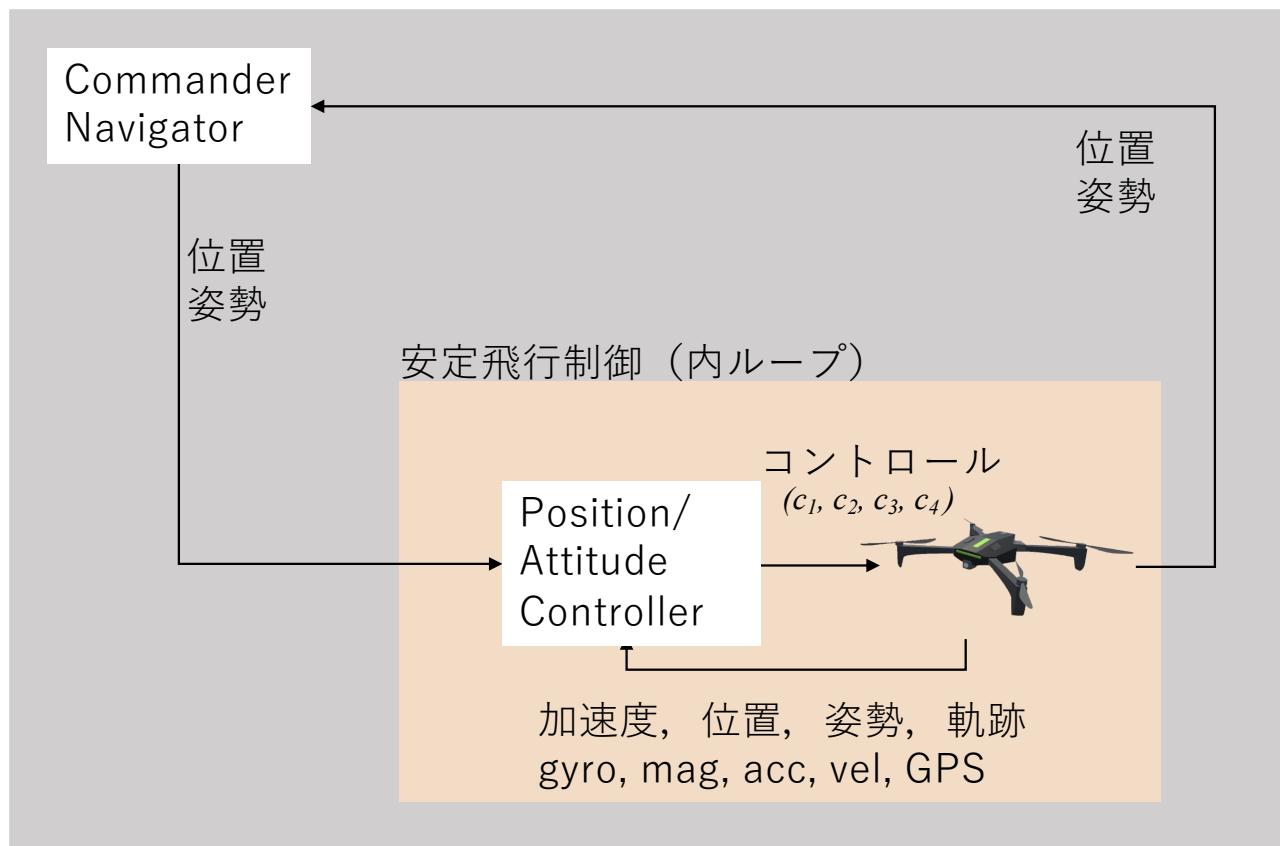
## 制御モデル(Plant+Controller)として

- Pのみは不安定（静止できない，地面があれば別）.
- Pのみは振動なしで発散（バネのように位置で変化する力がない）.
- PとCのフィードバックによって振動を伴って制御可能となる.
- さらに上位のミッションを受けた戦略（ステートマシン）で制御.



# 飛行計画と安定飛行制御

飛行計画（外ループ）



# ソフトウェア的な話

- 一文字の間違い (+/- とか sin/cos とか) でコンパイルエラーなしに動作時バグとなるので「レビュー容易性」「テスト容易性」が肝.

## 1. 数式とコード

数式が命. コード上, もっとも見やすく, レビューしやすくなれた.

## 2. クラスでなく関数の集合

テスト容易性のため, このレベルでは状態を持たなくなつた.

## 3. 単体テスト

本体と切り離して単体テストできる環境を最初から作り, 内部assert, 外部 assert (ユニットテスト) しまくった.

## 4. アルゴリズムの突合

機体と地上, 別のアルゴリズムで計算できる場合, 両方作って答え合わせをした.

```
#include <iostream>
#include "drone_physics.hpp"

int main() {
// このライブラリの名前空間
using namespace hako::drone_physics;
// 機体座標系を Euler 角で指定
VectorType frame = {0, 0, M_PI/2};
VelocityType body_velocity = {100, 200, 300};
// 機体座標系から地上座標系への速度変換
VelocityType ground_velocity = ground_vector_from_body(body_velocity, frame);

// x,y,z 座標を取り出す
auto [u, v, w] = ground_velocity;

std::cout << "u = " << u << ", v = " << v << ", w = " << w << std::endl;
// output: u = 200, v = -100, w = 300

// 逆変換して戻す
VelocityType body_velocity2 = body_vector_from_ground( {u, v, w}, {0, 0, M_PI/2});

auto [u2, v2, w2] = body_velocity2;
std::cout << "u2 = " << u2 << ", v2 = " << v2 << ", w2 = " << w2 << std::endl;
// output: u2 = 100, v2 = 200, w2 = 300, back again.
}
```

# Hello World(C++版)

# 1. ユニットテスト

```
drone_physics % ./utest
-----start unit test-----
test_frame_all_unit_vectors_with_angle0... PASS
test_frame_all_unit_vectors_with_some_angles... PASS
test_frame_matrix_is_unitary... PASS
...
-----all standard test PASSED!!---
test_issue_89_yaw_angle_bug... PASS
...
-----all bug issue test PASSED!!---
All(4667) asserts passed.
```

実装レベルの確認テスト

バグがでると再現テスト追加  
これは、issue 89 のヨー角のバグ

実数が入力なので、網を掛けて  
assert 文 70, 1回のbuildで 5,000 くらい、それでも安心できる。

## 2. 数式とコード(1/2)

```
/* acceleration in body frame based on mV' + w x mV = F ... eq.(1.136),(2.31)*/
AccelerationType acceleration_in_body_frame(
const VelocityType& body_velocity,
const EulerType& angle,
const AngularVelocityType& body_angular_velocity,
double thrust, double mass /* 0 is not allowed */,
double gravity, /* usually 9.8 > 0*/
double drag, /* air friction of 1-st order(-d1*v) counter to velocity */
{
    assert(!is_zero(mass));
    using std::sin; using std::cos;

    const auto
        c_phi = cos(angle.phi), s_phi = sin(angle.phi),
        c_theta = cos(angle.theta), s_theta = sin(angle.theta);
    const auto [u, v, w] = body_velocity;
    const auto [p, q, r] = body_angular_velocity;
    const auto T = thrust;
    const auto m = mass;
    const auto g = gravity; //...
```

引数は、型名と合わせ技で  
意味を伝える名前  
(ソフトウェアエンジニア向け)

内部でも assert かけまくる

引数を数式で使われている  
変数文字に全部バラす

## 2. 数式とコード

```
/*
 * See nonami's book eq.(1.136),(2.31)
 * Colioris's force is (p, q, r) x (u, v, w).
 *
 * Difference with the 'ground' version is that
 * (1) 'g' is broken down to x, y, z components.
 * (2) T is only relevant to z-axis.
 * (3) Coriolis force(using uvw,pqr) IS needed(because the frame is rotating!)
 */

double dot_u = - g * s_theta - (q*w - r*v) - d/m * u;
double dot_v = + g * c_theta * s_phi - (r*u - p*w) - d/m * v;
double dot_w = -T/m + g * c_theta * c_phi - (p*v - q*u) - d/m * w;

return {dot_u, dot_v, dot_w};
}
```

教科書の式番号

最も目レビューしやすいように  
できるだけ数式そのまま。  
(制御エンジニア向け)

$\dot{u} = -g \sin \theta \quad -(qw - rv) \quad -\frac{d}{m}u$

$\dot{v} = +g \cos \theta \sin \phi \quad -(ru - pw) \quad -\frac{d}{m}v$

$\dot{w} = -\frac{T}{m} \quad +g \cos \theta \cos \phi \quad -(pv - qu) \quad -\frac{d}{m}w$

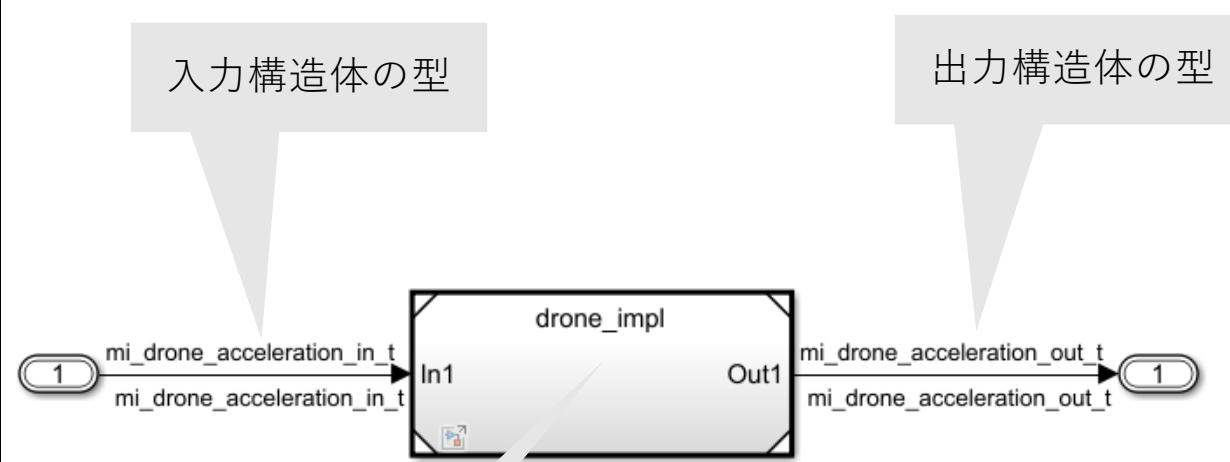
# その他感想など

- グラフ化はLLM 活用で  
open interpreter と会話してグラフ化
- コードもドキュメントも Copilot で生産性爆上がり（体感3倍）  
ドキュメントは（どんどん量は書けるが）冗長になりがち注意。
- アルゴリズムよりも機体パラメータ合わせ重要！！！  
そもそも物理的に制御できない、大きさやイナーシャとトルクのバランスなど  
なかなか安定して飛ばずに一番苦労した  
森さんが、パラメータ探索のためのコードを書いた
- C言語 I/F もあります。
- MATLAB 実装と差し替え可能。
- [README.ja](#)

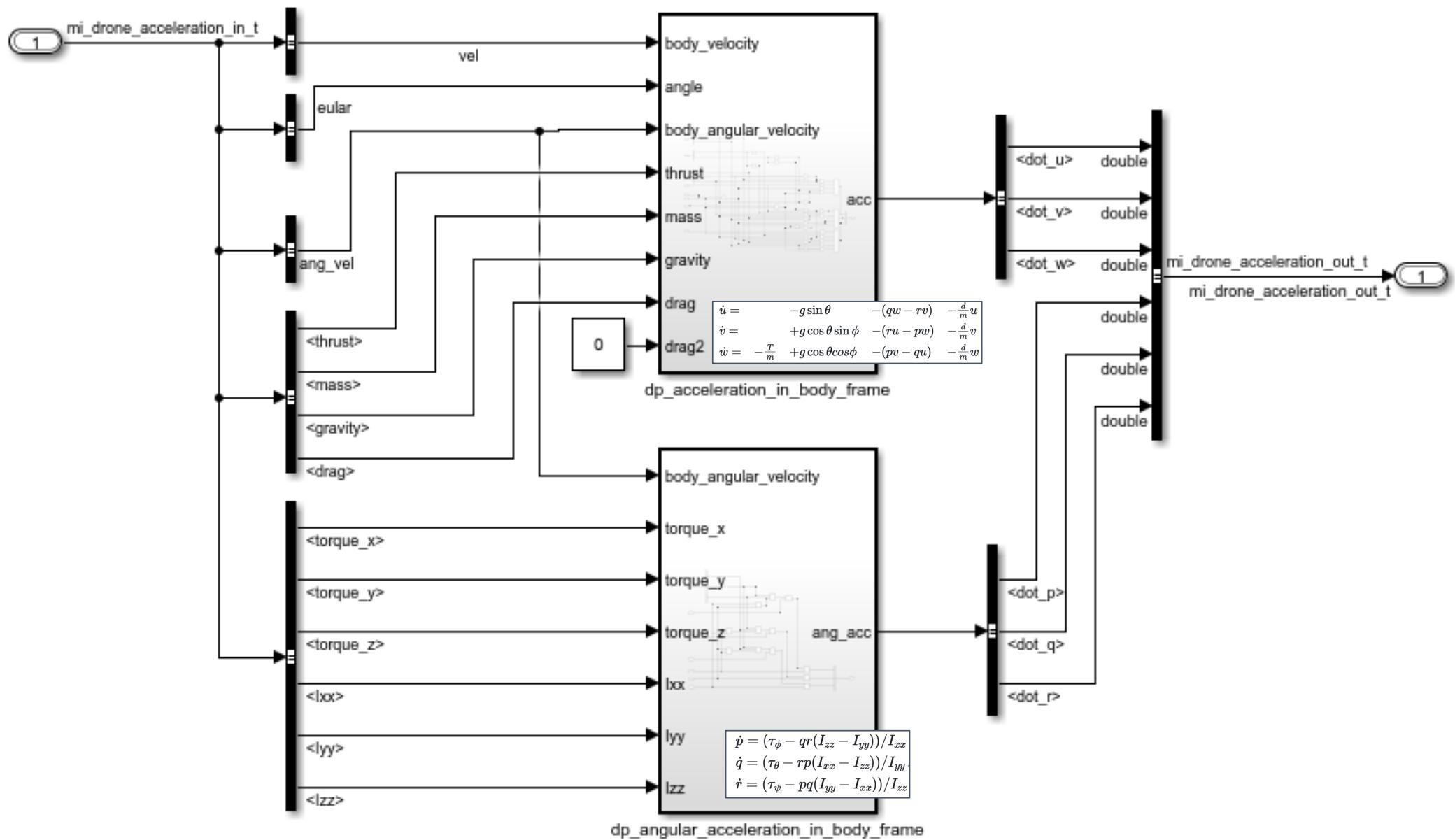
# MATLAB インターフェイス

Thanks ! > 三浦功也  
@MathWorks 😊

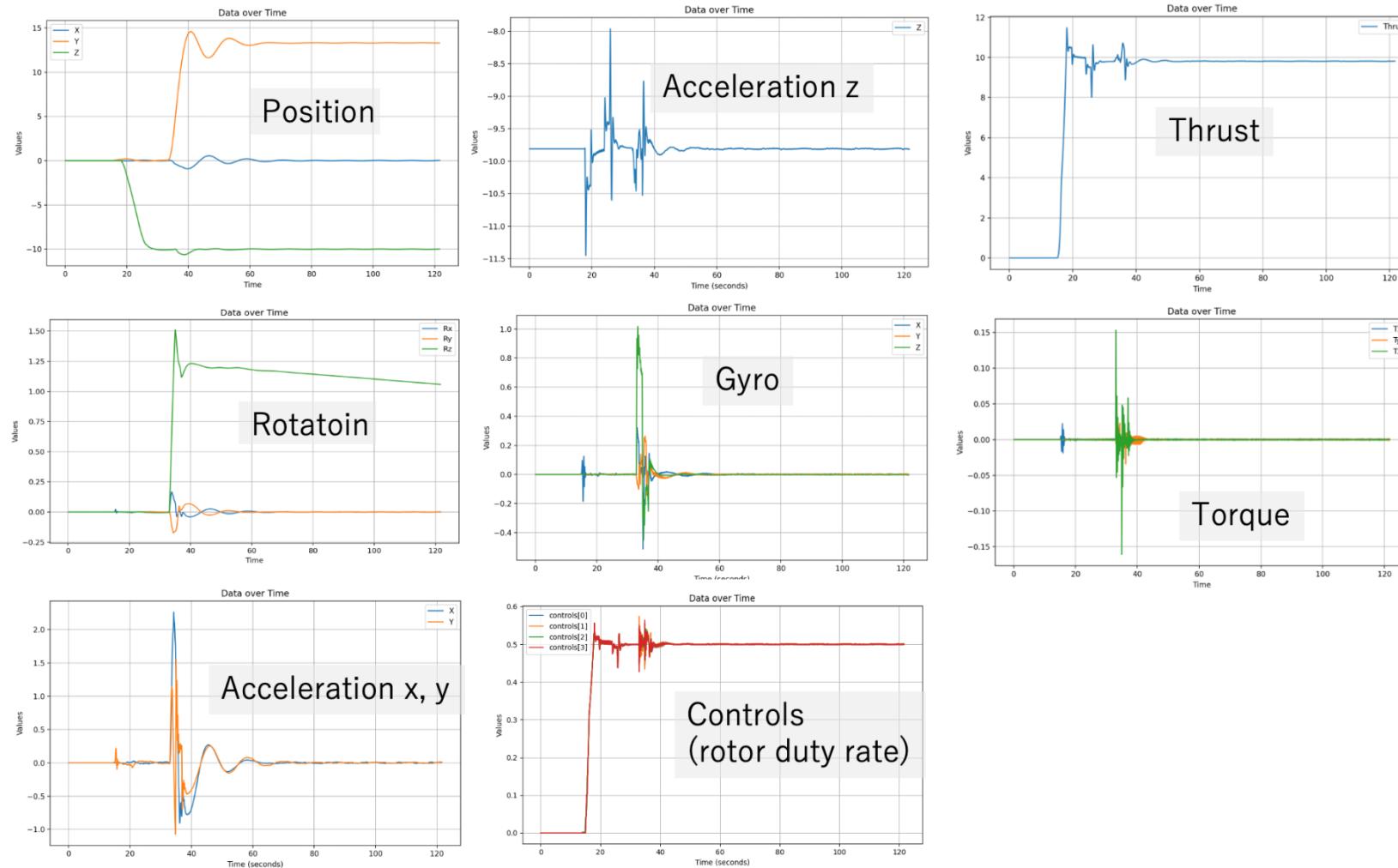
```
typedef struct mi_drone_acceleration_in_t {  
double phi; /* euler angle x */  
double theta; /* euler angle y */  
double psi; /* euler angle z */  
double u; /* velocity x */  
double v; /* velocity y */  
double w; /* velocity z */  
double p; /* angular velocity x */  
double q; /* angular velocity y */  
double r; /* angular velocity z */  
/* force and torque */  
double thrust;  
double torque_x;  
double torque_y;  
double torque_z;  
/* other constants */  
double mass;  
double Ixx;  
double Iyy;  
double Izz;  
double gravity;  
double drag;  
} mi_drone_acceleration_in_t;
```



```
typedef struct mi_drone_acceleration_out_t {  
double du; /* dot u = dotdot x = x - acceleration */  
double dv; /* dot v = dotdot y = y - acceleration */  
double dw; /* dot w = dotdot z = z - acceleration */  
double dp; /* dot p = x coordinate of angular vector acceleration */  
double dq; /* dot q = y coordinate of angular vector acceleration */  
double dr; /* dot r = z coordinate of angular vector acceleration */  
} mi_drone_acceleration_out_t;
```



## 2. グラフ化



# 参考文献

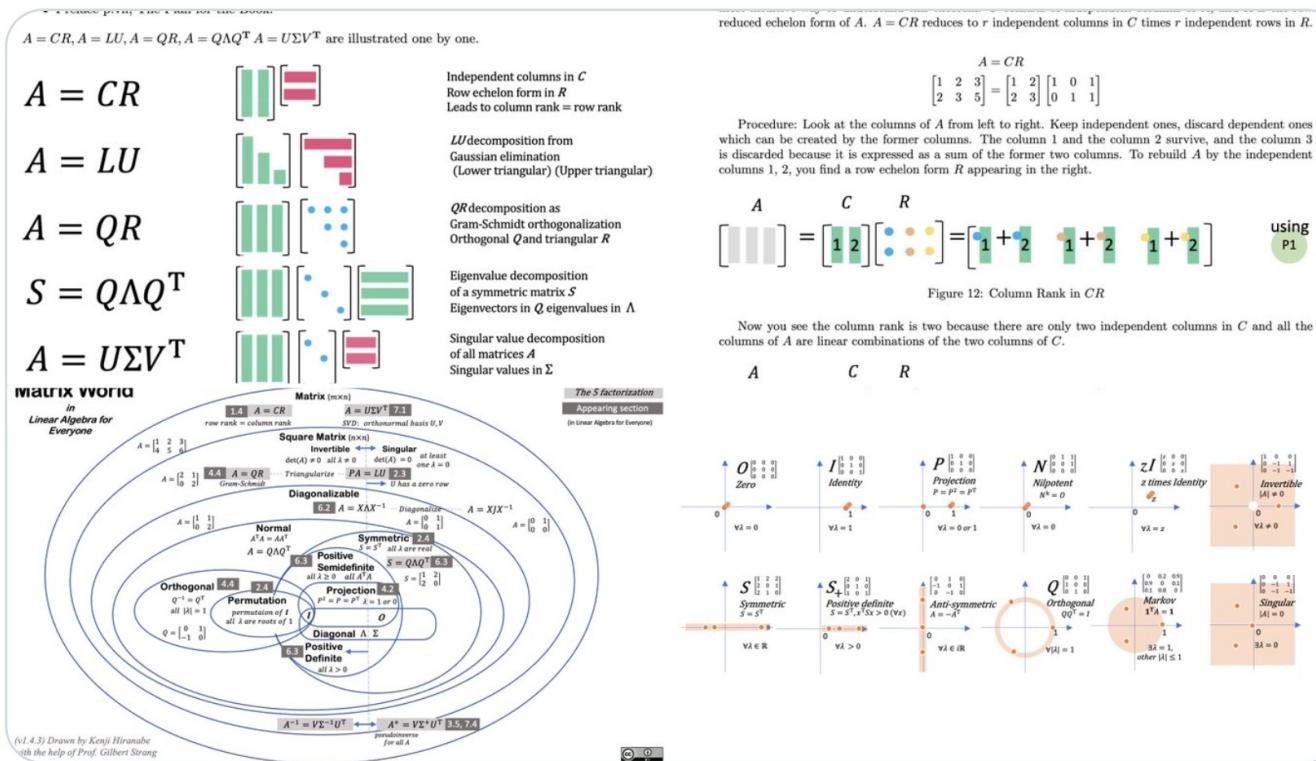
- [ドローン工学入門（野波健蔵博士）](#)
- [小型無人航空機の厳密・簡易なモデリングとモデルベース制御（野波健蔵博士）](#)
- [Drone control Lecture\(Seongheon Lee\)](#)
- [剛体の回転運動を支配するオイラーの運動方程式（スカイ技術研究所ブログ）](#)
- [オイラー角とは？定義と性質、回転行列・角速度ベクトルとの関係（スカイ技術研究所ブログ）](#)
- [飛行力学における機体座標系の定義\(@mtk\\_birdman\)](#)
- [「マルチコプタの運動と制御」基礎のきそ（伊藤恒平）](#)
- [線形代数の可視化 \(@kenjihiranabe\)](#)
- [Euler Angles and the Euler Rotation sequence\(Christopher Lum\), \[YouTube\]](#)





**Kenji Hiranabe**  
@hiranabe

My matrix visualizations with MIT Gilbert Strang. I compiled the graphics in a short(14-page) PDF paper!  
[github.com/kenjihiranabe/...](https://github.com/kenjihiranabe/)



<https://github.com/kenjihiranabe/The-Art-of-Linear-Algebra/blob/main/The-Art-of-Linear-Algebra-j.pdf>

...

この機会に線形代数、  
学び直しませんか？

