第 5 章

結論

本論文のまとめ

本研究では，従来のマルチロータヘリコプタでは実現できない，回転運動と並進運動の独立制御が可能な非平面マルチロータヘリコプタを対象とし，角速度制御系，姿勢制御系，速度制御系および位置制御系を構築し，実機実験によって，非平面マルチロータヘリコプタの有効性を実証した．

非平面マルチロータヘリコプタは，ロータを傾けて取り付けるだけで，アクチュエータの追加をせず，シンプルな機体構造を維持しながら，回転運動と並進運動の独立制御性を有している．機体構造の妥当性について，2 種類の非平面マルチロータヘリコプタと従来のマルチロータヘリコプタ，これら 3 機の運動性能を比較することで検証を行い，提案した機体構造の妥当性を確認した．機体構造の比較から，特に，z 軸周りに大きなトルクを生成可能であることが大きな利点となることを示した．

モデルベースト手法の最適制御の一種である LQI 制御により，回転運動と並進運動独立に制御可能な制御系を設計・構築し，実機実験によって非平面マルチロータヘリコプタの特徴である回転運動と並進運動の独立制御可能性を実証した．さらに，従来のマルチロータヘリコプタと位置決め性能を比較する実験を行い，並進運動時に推力ベクトルを傾けるために機体姿勢を変動させる予備動作が不要であるため，応答性が向上しており，位置決め性能の向上を実証するとともに，外乱に対して位置の揺動が抑制できるという性質があることを確認した．この性質が，点検任務や接触作業においては，非常に有利であると考えられ，実際の任務への利用が期待される結果が得られたと言える．

以上のように，回転運動と並進運動を独立制御可能な非平面マルチロータヘリコプタを開発し，実際の任務での利用を見据え，自律化に関する検討を行った．

今後の展望

最後に，今後の課題について述べる．非平面マルチロータヘリコプタの運動制御について，非線形制御手法を採用するなど，現在の運動制御系の性能向上を図り，ロバスト性の向上を目指す．

また，非平面マルチロータヘリコプタの並進運動については，従来機同様の並進運動と独立な並進運動による独自の並進運動の組み合わせにより実現される．そのため，非平面マル

チロータヘリコプタ独自の誘導制御手法が必要である．その概要を付録に示す．以上のことから，誘導制御について，最適性の検討，制約条件の設定から最適なモーションプランニング手法，経路生成手法，誘導制御手法を開発し，非平面マルチロータヘリコプタの完全自律化を目指す．

従来のマルチロータヘリコプタと異なる機体構造である提案機体構造について研究を進めることで，マルチロータヘリコプタの最適機体構造に関する知見を得ることも今後の課題である．

付録

角速度制御

角速度制御系設計 m ファイル

```
%% 2自由度PID 制御による角速度制御系設計

clear all;                                                  % ワークスペースリセット
format long;

%% パラメータ
Ts = 0.01;                                                  % 制御周期
f_F = 3;                                                    % フィードフォワード部のローパスフィルタのカットオフ周波数
f_D = 20;                                                   % 微分器のローパスフィルタのカットオフ周波数
f_cnt = 2;                                                  % PID コントローラの極パラメータ
J = 0.0504;                                                 % 慣性モーメント
K_dist = 0.5;                                               % ピークゲイン

%% PID ゲイン計算
Kp = 1 / K_dist;                                            % P ゲイン
Ki = pi*f_cnt*Kp;                                           % I ゲイン
Kd = (Kp / (4*pi*f_cnt)) - J;                               % D ゲイン

%% 微分器のローパスフィルタ
LPF = tf([1],[1/(2*pi*f_D) 1]);                             % LPF(s)

%% フィードフォワード部のローパスフィルタ
LPF1   = tf([1],[1/(2*pi*f_F) 1]);                          % LPF1(s),LPF2(s)
LPF1_d = c2d(LPF1,Ts,'zoh');                                % 離散化

%% フィードフォワード部全体
s_   = tf([1 0],[1]);                                       % s
FF   = LPF1*J*s_;                                           % FF(s)
FF_d = c2d(FF,Ts,'zoh');                                    % 離散化

%% PID コントローラ
Ctrl   = (tf([Kp],[1]))+((Ki*tf([1],[1 0])))+(LPF*Kd*s_);   % Ctrl(s)
Ctrl_d = c2d(Ctrl,Ts,'zoh');                                % 離散化

```

```
return
```

伝達関数計算ソースコード

```
int calcZfunction(const float* input_gain, float* input_var, const float* output_gain,
     float* output_var, int input_size, int output_size, float input_data){

    // Z 領域での伝達関数計算
    //
    //        a0 + a1*z^-1 + a2*z^-2 + ⋯
    // X[n] = ---------------------------- * U[n]
    //        b0 + b1*z^-1 + ⋯
    // を扱う.現時刻をn としたとき，各変数は以下とする。
    // input_gain  = {a0, a1, ⋯, a(input_size-1)}
    // input_var   = {U[n-1], U[n-2], U[n-3], ⋯}
    // output_gain = {b0, b1, ⋯, b(output_size-1)}
    // input_var   = {X[n-1], X[n-2], X[n-3], ⋯}
    // input_data  = U[n]
    // 本関数を実行すると,変数がアップデートされる
    // input_gain  = {a0, a1, ⋯, a(input_size-1)}
    // input_var   = {U[n], U[n-1], U[n-2], ⋯}
    // output_gain = {b0, b1, ⋯, b(output_size-1)}
    // input_var   = {X[n], X[n-1], X[n-2], ⋯}

    int i;

    if(input_size < 1 || output_size < 1)
                return -1;

    // input データを 1 ステップ分ずらす。
    for(i=0; i<(input_size - 1); i++) input_var[i+1] = input_var[i];
    input_var[0] = input_data;

    // output データを 1 ステップ分ずらす。
    for(i=0; i<(output_size - 1); i++) output_var[i+1] = output_var[i];

    //output 値更新
    output_var[0] = 0.0;
    for(i=0;i < input_size;i++){
        output_var[0] += input_gain[i] * input_var[i];
    }
    for(i=1;i < output_size;i++){
        output_var[0] += -1.0 * output_gain[i] * output_var[i];
    }
    output_var[0] = output_var[0] / output_gain[0];

    return 0;
}
```

運動性能評価シミュレーション

```
%% 運動性能評価シミュレーション

clear all;                                                  % ワークスペースリセット

%% パラメータ
A_tilt = 30;                                               % モータ取り付け角度  [deg]
L      = 0.34;                                             % 機体中心-ロータ中心間距離 [m]
Kf     = 3.104E-7;                                         % 推力係数 [N/rpm^2]
Kt     = 7.171E-9;                                         % 反トルク係数 [N/rpm^2]

S = sin(A_tilt * pi/180);
C = cos(A_tilt * pi/180);

%% 変換行列
%% 内外傾斜型
% 推力由来
T_thrust_inout =
[ sqrt(3)*Kf*S/2       0   -sqrt(3)*Kf*S/2    sqrt(3)*Kf*S/2       0  -sqrt(3)*Kf*S/2;
       -1*Kf*S/2    Kf*S         -1*Kf*S/2         -1*Kf*S/2    Kf*S        -1*Kf*S/2;
           -Kf*C   -Kf*C             -Kf*C             -Kf*C   -Kf*C            -Kf*C;
        L/2*Kf*C  L*Kf*C          L/2*Kf*C         -L/2*Kf*C -L*Kf*C        -L/2*Kf*C;
sqrt(3)*L/2*Kf*C       0 -sqrt(3)*L/2*Kf*C -sqrt(3)*L/2*Kf*C       0 sqrt(3)*L/2*Kf*C;
               0       0                 0                 0       0                0
];
% 反トルク由来
T_torque_inout =
[            0     0               0               0    0              0;
             0     0               0               0    0              0;
             0     0               0               0    0              0;
sqrt(3)*Kt*S/2     0 -sqrt(3)*Kt*S/2 -sqrt(3)*Kt*S/2    0 sqrt(3)*Kt*S/2;
     -1*Kt*S/2  Kt*S       -1*Kt*S/2       -1*Kt*S/2 Kt*S      -1*Kt*S/2;
          Kt*C -Kt*C            Kt*C           -Kt*C Kt*C          -Kt*C
];

% 内外傾斜型変換行列
T_inout = T_thrust_inout + T_torque_inout;

%% ねじり型
% 推力由来
T_thrust_twist =
[       1/2*Kf*S   -Kf*S          1/2*Kf*S          1/2*Kf*S   -Kf*S         1/2*Kf*S;
  sqrt(3)/2*Kf*S       0   -sqrt(3)/2*Kf*S    sqrt(3)/2*Kf*S       0  -sqrt(3)/2*Kf*S;
           -Kf*C   -Kf*C             -Kf*C             -Kf*C   -Kf*C            -Kf*C;
        L/2*Kf*C  L*Kf*C          L/2*Kf*C         -L/2*Kf*C -L*Kf*C        -L/2*Kf*C;
sqrt(3)/2*L*Kf*C       0 -sqrt(3)/2*L*Kf*C -sqrt(3)/2*L*Kf*C       0 sqrt(3)/2*L*Kf*C;
          L*Kf*S -L*Kf*S            L*Kf*S           -L*Kf*S  L*Kf*S          -L*Kf*S
];
```

```
% 反トルク由来
T_torque_twist =
[             0     0              0              0    0               0;
              0     0              0              0    0               0;
              0     0              0              0    0               0;
      -1/2*Kt*S -Kt*S      -1/2*Kt*S       1/2*Kt*S Kt*S        1/2*Kt*S;
-sqrt(3)/2*Kt*S     0 sqrt(3)/2*Kt*S sqrt(3)/2*Kt*S    0 -sqrt(3)/2*Kt*S;
           Kt*C -Kt*C           Kt*C          -Kt*C Kt*C           -Kt*C
];
% ねじり型変換行列
T_twist = T_thrust_twist + T_torque_twist;

%% 平面型
% 推力由来
T_thrust_planar =
[            0    0               0               0     0              0;
             0    0               0               0     0              0;
           -Kf  -Kf             -Kf             -Kf   -Kf            -Kf;
        L/2*Kf L*Kf          L/2*Kf         -L/2*Kf -L*Kf        -L/2*Kf;
sqrt(3)*L/2*Kf    0 -sqrt(3)*L/2*Kf -sqrt(3)*L/2*Kf     0 sqrt(3)*L/2*Kf;
             0    0               0               0     0              0
];
% 反トルク由来
T_torque_planar =
[ 0   0  0   0  0   0;
  0   0  0   0  0   0;
  0   0  0   0  0   0;
  0   0  0   0  0   0;
  0   0  0   0  0   0;
 Kt -Kt Kt -Kt Kt -Kt
 ];
% 変換行列
T_planar = T_thrust_planar + T_torque_planar;

%% x 軸トルクシミュレーション

% 入力u_m 生成
i = 0;
for width = 1:180
    i = i+1;
    if (width < 25)
        width_rplus  = 1590;
        width_rminus = 1590;
    end
    if (25 <= width && width < 40)
        width_rplus  = 1590 + (width - 25) * 100/15;
        width_rminus = 1590 - (width - 25) * 100/15;
    end
    if (40 <= width && width < 65)
        width_rplus  = 1690;
        width_rminus = 1490;
```

```
    end
    if (65 <= width && width < 80)
        width_rplus  = 1690 - (width - 65) * 100/15;
        width_rminus = 1490 + (width - 65) * 100/15;
    end
    if (80 <= width && width <100)
        width_rplus  = 1590;
        width_rminus = 1590;
    end
    if (100 <= width && width < 115)
        width_rplus  = 1590 - (width - 100) * 100/15;
        width_rminus = 1590 + (width - 100) * 100/15;
    end
    if (115 <= width && width < 140)
        width_rplus  = 1490;
        width_rminus = 1690;
    end
    if (140 <= width && width < 155)
        width_rplus  = 1490 + (width - 140) * 100/15;
        width_rminus = 1690 - (width - 140) * 100/15;
    end
    if (155 <= width)
        width_rplus  = 1590;
        width_rminus = 1590;
    end

    rpm12(i) = ((width_rplus / 0.084) - 12254)^2;
    rpm22(i) = ((width_rplus / 0.084) - 12254)^2;
    rpm32(i) = ((width_rplus / 0.084) - 12254)^2;
    rpm42(i) = ((width_rminus / 0.084) - 12254)^2;
    rpm52(i) = ((width_rminus / 0.084) - 12254)^2;
    rpm62(i) = ((width_rminus / 0.084) - 12254)^2;

    tau_x_inout(i)  =
        T_inout(4,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
    tau_x_twist(i)  =
        T_twist(4,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
    tau_x_planar(i) =
        T_planar(4,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
end

% 時系列生成
n = length(rpm12);
t = [0:1:n-1]';

% 描画
% 入力
figure(1)
plot(t,sqrt(rpm12),t,sqrt(rpm42))
grid on;
% x 軸トルク
```

```
figure(2)
plot(t,tau_x_planar,t,tau_x_inout,t,tau_x_twist)
grid on;

% y 軸シミュレーション
i = 0;
for width = 1:180
    i = i+1;
    if (width < 25)
        width_pplus  = 1590;
        width_pminus = 1590;
    end
    if (25 <= width && width < 40)
        width_pplus  = 1590 + (width - 25) * 100/15;
        width_pminus = 1590 - (width - 25) * 100/15;
    end
    if (40 <= width && width < 65)
        width_pplus  = 1690;
        width_pminus = 1490;
    end
    if (65 <= width && width < 80)
        width_pplus  = 1690 - (width - 65) * 100/15;
        width_pminus = 1490 + (width - 65) * 100/15;
    end
    if (80 <= width && width <100)
        width_pplus  = 1590;
        width_pminus = 1590;
    end
    if (100 <= width && width < 115)
        width_pplus  = 1590 - (width - 100) * 100/15;
        width_pminus = 1590 + (width - 100) * 100/15;
    end
    if (115 <= width && width < 140)
        width_pplus  = 1490;
        width_pminus = 1690;
    end
    if (140 <= width && width < 155)
        width_pplus  = 1490 + (width - 140) * 100/15;
        width_pminus = 1690 - (width - 140) * 100/15;
    end
    if (155 <= width)
        width_pplus  = 1590;
        width_pminus = 1590;
    end

    rpm12(i) = ((width_pplus / 0.084) - 12254)^2;
    rpm22(i) = ((1590 / 0.084) - 12254)^2;
    rpm32(i) = ((width_pminus / 0.084) - 12254)^2;
    rpm42(i) = ((width_pminus / 0.084) - 12254)^2;
    rpm52(i) = ((1590 / 0.084) - 12254)^2;
    rpm62(i) = ((width_pplus / 0.084) - 12254)^2;
```

```
201
202     tau_pitch_inout(i)  =
203         T_inout(5,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
204     tau_pitch_twist(i)  =
205         T_twist(5,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
206     tau_pitch_planar(i) =
207         T_planar(5,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
208 end
209
210 % 時系列生成
211 n = length(rpm12);
212 t = [0:1:n-1]';
213
214 % 描画
215 % 入力
216 figure(3)
217 plot(t,sqrt(rpm12),t,sqrt(rpm42),t,sqrt(rpm22))
218 grid on;
219 % y 軸トルク
220 figure(4)
221 plot(t,tau_pitch_planar,t,tau_pitch_inout,t,tau_pitch_twist)
222 grid on;
223
224 % z 軸シミュレーション
225 i = 0;
226 for width = 1:180
227     i = i+1;
228     if (width < 25)
229         width_rplus  = 1590;
230         width_rminus = 1590;
231     end
232     if (25 <= width && width < 40)
233         width_rplus  = 1590 + (width - 25) * 100/15;
234         width_rminus = 1590 - (width - 25) * 100/15;
235     end
236     if (40 <= width && width < 65)
237         width_rplus  = 1690;
238         width_rminus = 1490;
239     end
240     if (65 <= width && width < 80)
241         width_rplus  = 1690 - (width - 65) * 100/15;
242         width_rminus = 1490 + (width - 65) * 100/15;
243     end
244     if (80 <= width && width <100)
245         width_rplus  = 1590;
246         width_rminus = 1590;
247     end
248     if (100 <= width && width < 115)
249         width_rplus  = 1590 - (width - 100) * 100/15;
250         width_rminus = 1590 + (width - 100) * 100/15;
251     end
```

```
    if (115 <= width && width < 140)
        width_rplus  = 1490;
        width_rminus = 1690;
    end
    if (140 <= width && width < 155)
        width_rplus  = 1490 + (width - 140) * 100/15;
        width_rminus = 1690 - (width - 140) * 100/15;
    end
    if (155 <= width)
        width_rplus  = 1590;
        width_rminus = 1590;
    end

    rpm12(i) = ((width_rplus / 0.084) - 12254)^2;
    rpm22(i) = ((width_rminus / 0.084) - 12254)^2;
    rpm32(i) = ((width_rplus / 0.084) - 12254)^2;
    rpm42(i) = ((width_rminus / 0.084) - 12254)^2;
    rpm52(i) = ((width_rplus / 0.084) - 12254)^2;
    rpm62(i) = ((width_rminus / 0.084) - 12254)^2;

    tau_yaw_inout(i) =
        T_inout(6,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
    tau_yaw_twist(i) =
        T_twist(6,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
    tau_yaw_planar(i) =
        T_planar(6,:) * [rpm12(i) rpm22(i) rpm32(i) rpm42(i) rpm52(i) rpm62(i)]';
end

% 時系列生成
n = length(rpm12);
t = [0:1:n-1]';

% 入力
figure(5)
plot(t,sqrt(rpm12),t,sqrt(rpm42))
grid on;
% z 軸トルク
figure(6)
plot(t,tau_yaw_planar,t,tau_yaw_inout,t,tau_yaw_twist)
grid on;

return
```

姿勢制御

姿勢モデル同定 m ファイル

```
%% 角速度指令値から機体姿勢までのモデルを求める

clear all;                                      % ワークスペースリセット
close all;                                      % 描画ウインドウを閉じる

%% 実験データ読み込み
dat = load('Filename');

%% モデルパラメータ
K           = 1;
T_omegab    = 0.083;
zeta        = 0.9;

%% 角速度モデル
% モデルを 2次の伝達関数で近似する
%                 K                          <-- num
% -----------------------------------
% (T_omegab*s^2 + 2*ζ*T_omegab*s + 1)    <-- den

% 伝達関数の分母分子生成
num = [K];                                      % 分子
den = [T_omegab 2*zeta*T_omegab 1];             % 分母
G   = tf(num, den);                             % 伝達関数

% 伝達関数への入力生成
u = reference;                                  % 角速度指令値を入力とする

% 時系列生成
n = length(u);                                  % データ点数
t = [0:0.01:0.01*(n-1)]';                       % 時系列生成 データサンプリング 100Hz

% 伝達関数を用いたシミュレーション
Omegab_tf = lsim(G, u, t);                      % 伝達関数G に入力

% 実験データ(omegab) とモデル出力 (Omegab_tf) を比較
% 結果を描画
figure(1)
plot(t, omegab, t,Omegab_tf)
grid on;

%% 姿勢モデル
% 角速度モデルを積分
den_att = conv([1 0], den);

%% 伝達関数から状態空間モデルへ
```

```
[A_att, B_att, C_att, D_att] = tf2ss(num,den_att);

return
```

姿勢制御系設計 m ファイル

```
%% LQI 制御による姿勢制御系設計

clear all;                                          % ワークスペースリセット

%% 姿勢モデル読み込み
load('Filename');

%% ゲイン
% 評価関数ゲイン
Q = [1  0       0    0;
     0 10       0    0;
     0  0 6000000    0;
     0  0       0 6000];
R = [7.0];

% オブザーバゲイン
Q_obs = [0.5];
R_obs = [0.5];

%% サーボ拡大系の構築
% 行列の受け渡し
A = A_model;
B = B_model;
C = C_model;

% モデルサイズ設定
size_a    = size(A);
size_a    = size_a(1);
[m_b,n_b] = size(B);
[m_c,n_c] = size(C);

% サーボ拡大系
As = [ A zeros(size_a,m_c)
       -C zeros(m_c,m_c)];
Bs = [B
      zeros(m_c,n_b)];

% サーボ拡大系のモデルサイズ設定
size_as     = size(As);
[m_bs,n_bs] = size(Bs);

%% lqr によるフードバックゲイン計算
K_cnt  = lqr(As,Bs,Q,R);                            % フィードバックゲイン
K_cnt1 = K_cnt(1:n_b,1:size_a);                     % 状態にかかるフィードバックゲイン
```

```
K_cnt2 = K_cnt(1:n_b,(size_a+1):size_as);          % 目標値と出力の偏差の積分値にかかるフィードバックゲイン

%% オブザーバ
% オブザーバゲイン計算
K_obs = lqe(A,B,C,Q_obs,R_obs);

% オブザーバつきコントローラ
A_cnt = [A-B*K_cnt1-K_obs*C  -B*K_cnt2
              zeros(m_c,n_c)  zeros(m_c,m_c)];
B_cnt = [K_obs zeros(m_b,m_c)
         -eye(m_c,m_c) eye(m_c,m_c)];
C_cnt = [-K_cnt1 -K_cnt2
         zeros(m_c,size_a),eye(m_c)];
D_cnt = zeros(n_b+m_c,2*m_c);

% 離散化
st = 0.02;                                          % 姿勢制御周期
[Arisan,Brisan,Crisan,Drisan]=c2dm(A_cnt,B_cnt,C_cnt,D_cnt,st,'zoh');

%% シミュレーション
sim('AttitudeController_Sim.mdl',10);

return
```

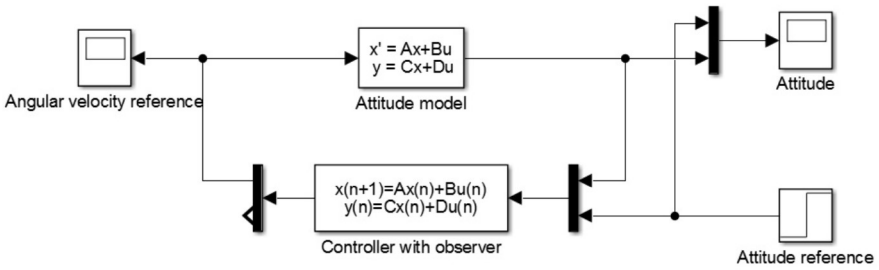

Fig. 1 Simulation of attitude controller(AttitudeController_Sim.mdl)

速度制御

速度モデル同定 m ファイル

```
%% 力指令値から機体速度までのモデルを求める

clear all;                    % ワークスペースリセット
close all;                    % 描画ウインドウを閉じる

%% 実験データ読み込み
dat = load('Filename');

%% モデルパラメータ
M   = 5.0;                    % 機体質量
Ta  = 0.084;                  % 加速度の時定数
Tv  = 1.5;                    % 速度の時定数

% 時系列生成
n = length(P);
t = [0:0.01:0.01*(n-1)]';    %データサンプリング 100Hz

%% 加速度モデル
A = [-1/Ta];
B = [1/(M*Ta)]';              % 力指令値を加速度指令値に変換
C = [1];
D = [0];

% 伝達関数への入力生成
u = reference;                % 力指令値を入力とする

% 状態空間モデルシミュレーション
Sys_Acc = ss(A,B,C,D);
Acc_sim = lsim(Sys_Acc,u,t);

% 実験データ(Acc)とモデル出力(Acc_sim)を比較
% 結果を描画
figure(1)
plot(t,Acc,t,Acc_sim)
grid on;

%% 速度モデル
A = [-1/Ta     0;
       1/Tv -1/Tv];
B = [1/(M*Ta) 0]';
C = [0 1];
D = [0];

% 状態空間モデルを用いたシミュレーション
Sys_Vel = ss(A,B,C,D);
```

```
Vel_sim = lsim(Sys_Vel,u,t);

% 実験データ(Vel) とモデル出力 (Vel_sim) を比較
% 結果を描画
figure(2)
plot(t,Vel,t,Vel_sim)
grid on;

%% 状態抽出
A_vel = A;
B_vel = B;
C_vel = C;
D_vel = D;

return
```

速度制御系設計 m ファイル

```
%% LQI 制御による速度制御系設計

clear all;                                          % ワークスペースリセット

%% 姿勢モデル読み込み
load('Filename');

%% ゲイン
% 評価関数ゲイン
Q = [1000      0         0;
        0 270000         0;
        0      0 10000000];
R = [50.0];

% オブザーバゲイン
Q_obs = [10];
R_obs = [1    0;
         0 0.01];

%% サーボ拡大系の構築
% 行列の受け渡し
A = A_model;
B = B_model;
C = C_model;

% モデルサイズ設定
size_a    = size(A);
size_a    = size_a(1);
[m_b,n_b] = size(B);
[m_c,n_c] = size(C);

% サーボ拡大系
```

```
As = [ A zeros(size_a,m_c)
      -C zeros(m_c,m_c)];
Bs = [B
      zeros(m_c,n_b)];

% サーボ拡大系のモデルサイズ設定
size_as     = size(As);
[m_bs,n_bs] = size(Bs);

%% lqr によるフードバックゲイン計算
K_cnt  = lqr(As,Bs,Q,R);
K_cnt1 = K_cnt(1:n_b,1:a);
K_cnt2 = K_cnt(1:n_b,(a+1):as);

% 加速度と速度をオブザーバへの入力とするため
C_newmodel = [1 0;
              0 1];

%% オブザーバ
% オブザーバゲイン計算
K_obs = lqe(A,B,C_newmodel,Q_obs,R_obs);

% オブザーバつきコントローラ
A_cnt = [A-B*K_cnt1-K_obs*C_newmodel  -B*K_cnt2
       zeros(m_c,n_c)  zeros(m_c,m_c)];
B_cnt = [K_obs zeros(2,1)
         0 -1 1];
C_cnt = [-K_cnt1 -K_cnt2
          zeros(m_c,a),eye(m_c)];
D_cnt = zeros(2,3);

% 離散化
st = 0.02;                                              % 速度制御周期
[Arisan,Brisan,Crisan,Drisan]=c2dm(A_cnt,B_cnt,C_cnt,D_cnt,st,'zoh');

%% シミュレーション
sim('VelocityController_Sim.mdl',10);

return
```

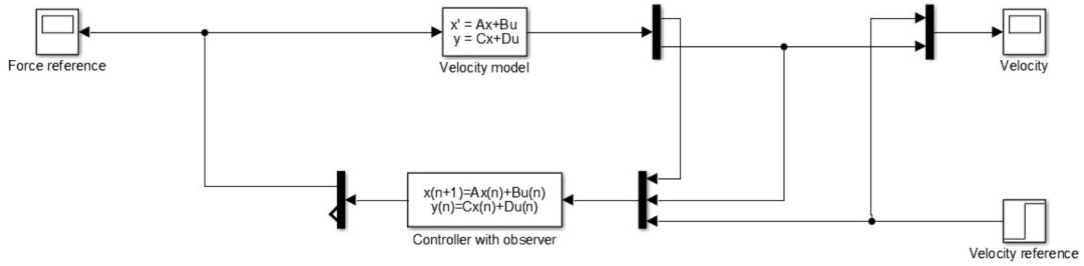

Fig. 2 Simulation of velocity controller(VelocityController_Sim.mdl)

非平面マルチロータヘリコプタの誘導制御

マルチロータヘリコプタの実用化にともない，自律化が求められる．農薬散布，災害調査，点検などの任務ではウェイポイント飛行が行われる．Fig.3，4 に示すように地図上にウェイポイントを設定し，そのウェイポイントを通過するように経路を生成し，経路に追従するように飛行する．また，DJI 社 [34] は TAP AND GO ウェイポイントとして，各ポイントで，高度の変更やカメラの回転などを設定できるシステムを提供している．このように，経路へ追従させるために，機体速度や姿勢を制御することを誘導制御といい，ロケットや宇宙機で広く用いられている [35] [36]．筆者らの研究グループにおいても，誘導制御についての研究がされている [37] [38]．特に，文献 [38] では，衝突回避を考慮した，マルチロータヘリコプタの誘導制御について研究している．

誘導制御において，制御対象のダイナミクスを考える．これまでに本論文で述べたように，非平面マルチロータヘリコプタは回転運動と並進運動を独立に制御できるという従来のマルチロータヘリコプタと異なる性質をもち，ダイナミクスについても固有の表現がされる．そのため，非平面マルチロータヘリコプタ固有のダイナミクスを考慮した誘導制御手法および経路生成手法を考える必要がある．

以降，非平面マルチロータヘリコプタの誘導制御手法について，将来の展望を含めて述べる． 機体の位置と姿勢を要素にもつ状態ベクトル $\boldsymbol{x}$ を定義する．

$$\boldsymbol{x} = \begin{bmatrix} X_O \\ Y_O \\ Z_O \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{1}$$

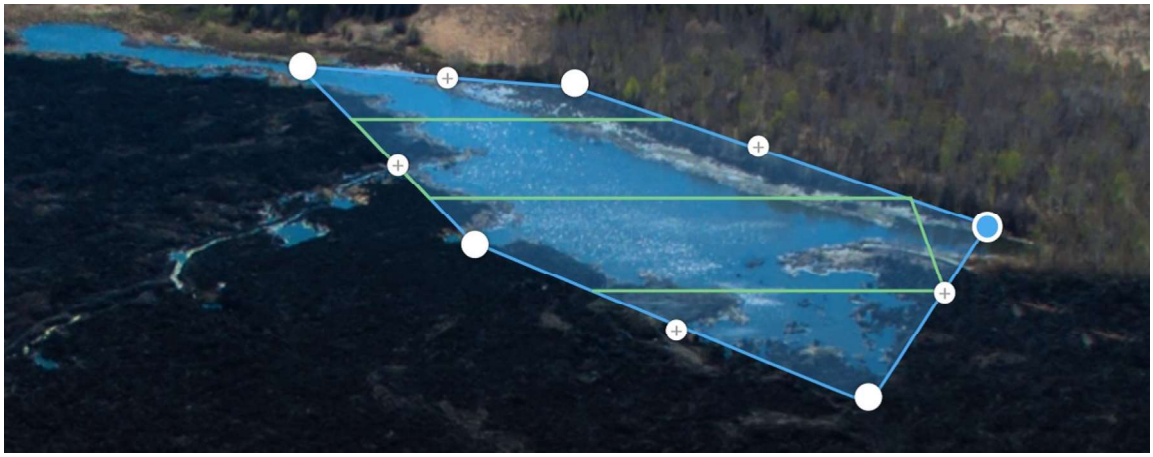

Fig. 3 Way point [33]

Fig. 4 Way point (TAP AND GO) [33]

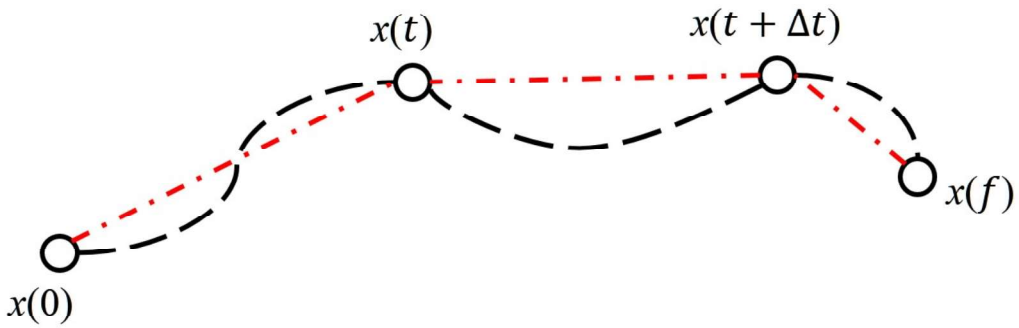

Fig. 5 Guidance control and path planning

さらに，所望の状態 $\boldsymbol{x}_d$ を定義する．

$$\boldsymbol{x}_d = \begin{bmatrix} X_{O_d} \\ Y_{O_d} \\ Z_{O_d} \\ \phi_d \\ \theta_d \\ \psi_d \end{bmatrix} \tag{2}$$

誘導制御の概要を Fig.5 に示す．誘導制御は，Fig.5 中に示す Δt 秒後の状態 $\boldsymbol{x}(t+\Delta t)$ を指定し，各時間での状態を実現し，$\boldsymbol{x}$ を $\boldsymbol{x}_d$ に一致させるように，機体姿勢および位置を制御することである．

$$\boldsymbol{x} \rightarrow \boldsymbol{x}_d \tag{3}$$

また，このとき，Fig.5 中の赤い一点鎖線のように，各点を最短距離で結ぶ経路や，黒い破線のように，各点をなめらかにつなぐ経路など，様々な経路の候補が考えられる．このように，移動距離の長さ，障害物との距離などを考慮した設計思想のもと，各点での機体位置を実現する経路を生成する問題を経路生成問題という．

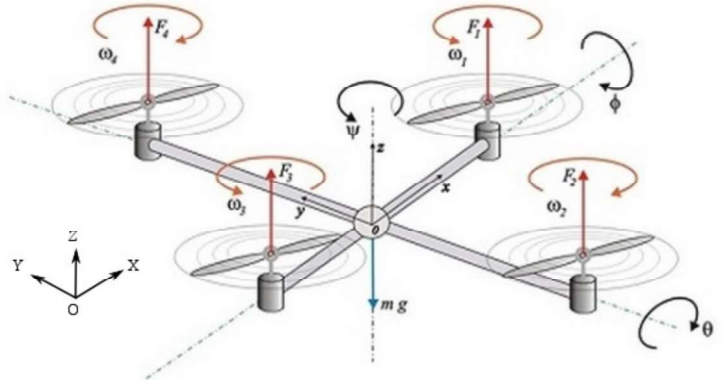

Fig. 6 Quadrotor Dynamics [39]

非平面マルチロータヘリコプタの並進運動

従来のマルチロータヘリコプタの並進運動は，機体姿勢を変え，ロータが生成する推力の方向を傾けることで実現する．Fig.6 に示すマルチロータヘリコプタの並進のダイナミクスは式 (4) で表すことができる [39] [40].

$$\begin{aligned}
\ddot{X} &= (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\frac{U_1}{m} \\
\ddot{Y} &= (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\frac{U_1}{m} \\
\ddot{Z} &= -g + (\cos\theta\cos\phi)\frac{U_1}{m}
\end{aligned} \tag{4}$$

ただし，m は機体質量，g は重力加速度，U_1 は推力係数 b を用いて式 (5) で定義される．

$$U_1 = b\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right) \tag{5}$$

式 (4) のように，機体姿勢をパラメータとしてダイナミクスが表現される．

つづいて，非平面マルチロータヘリコプタのダイナミクスを考える．本論文で定義した座標系において並進のダイナミクスを表現すると以下の式 (6) が得られる．

$$\begin{aligned}
\begin{bmatrix} \ddot{X}_O \\ \ddot{Y}_O \\ \ddot{Z}_O - g \end{bmatrix} &= \frac{1}{M}\begin{bmatrix} F_{X_O} \\ F_{Y_O} \\ F_{Z_O} \end{bmatrix} \\
&= \frac{1}{M}\boldsymbol{R}_z \cdot \begin{bmatrix} F_{X_H} \\ F_{Y_H} \\ F_{Z_H} \end{bmatrix} = \frac{1}{M}\boldsymbol{R}_z \cdot \boldsymbol{R}_x \cdot \boldsymbol{R}_y \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}
\end{aligned} \tag{6}$$

$$\begin{aligned}
\boldsymbol{R}_{zxy} &= \boldsymbol{R}_z \cdot \boldsymbol{R}_x \cdot \boldsymbol{R}_y \\
&= \begin{bmatrix} \cos\theta\cos\psi - \sin\phi\sin\theta\sin\psi & -\cos\phi\sin\psi & \sin\theta\cos\psi + \sin\phi\cos\theta\sin\psi \\ \cos\theta\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi & \sin\theta\sin\psi - \sin\phi\cos\theta\cos\psi \\ -\cos\phi\sin\theta & \sin\theta & \cos\phi\cos\theta \end{bmatrix}
\end{aligned} \tag{7}$$

ただし，Z_O 軸には重力補償が入っている．いま，非平面マルチロータヘリコプタは機体座標系 x，y，z の全軸に力を生成できるため，ダイナミクスは f_x，f_y，f_z からなる行列を用いた表現となる．対して，文献 [39]，[40] で導出された式 (4) の従来のマルチロータヘリコプタのダイナミクスはにおいては，U_1 が f_z に相当する．式 (6) において，$f_x = f_y = 0$ とすることで，同じ形で表現することはできる．この，f_x，f_y，f_z のすべてを活用し，並進運動を実現できる．

誘導制御

ここで，初期状態 $\boldsymbol{x}(0)$ から時刻 t での状態 $\boldsymbol{x}(t)$ を通して終端状態 $\boldsymbol{x}(f)$ までの状態遷移を考える．従来のマルチロータヘリコプタであれば，回転運動と並進運動が連成するため，経路が決まれば状態を一意に決めることができる．それに対して，非平面マルチロータは所望の経路に対して，状態が一意に決まらない．初期条件と終端条件が式 (8) で与えられたときを考える．

$$\begin{aligned} \boldsymbol{x}(0) &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{x}(f) &= \begin{bmatrix} 5 & 5 & 1 & \frac{1}{6}\pi & -\frac{1}{12}\pi & \frac{1}{2}\pi \end{bmatrix}^T \end{aligned} \tag{8}$$

このとき，状態の遷移として式 (9)，(10)，(11) のように様々な挙動が実現できる．式 (9) のように，位置を合わせ，姿勢を合わせる方法，式 (10) のように，姿勢を合わせ，位置を合わせる方法，式 (11) のように全ての状態を同時に遷移する方法がある．

$$\boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow \quad \boldsymbol{x}(t) = \begin{bmatrix} 5 \\ 5 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow \quad \boldsymbol{x}(f) = \begin{bmatrix} 5 \\ 5 \\ 1 \\ \frac{1}{6}\pi \\ -\frac{1}{12}\pi \\ \frac{1}{2}\pi \end{bmatrix} \tag{9}$$

$$\boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow \quad \boldsymbol{x}(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{6}\pi \\ -\frac{1}{12}\pi \\ \frac{1}{2}\pi \end{bmatrix} \quad \rightarrow \quad \boldsymbol{x}(f) = \begin{bmatrix} 5 \\ 5 \\ 1 \\ \frac{1}{6}\pi \\ -\frac{1}{12}\pi \\ \frac{1}{2}\pi \end{bmatrix} \tag{10}$$

$$\boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow \quad \boldsymbol{x}(t) = \begin{bmatrix} 2 \\ 3 \\ 0.7 \\ \frac{1}{3}\pi \\ -\frac{1}{8}\pi \\ \frac{1}{3}\pi \end{bmatrix} \quad \rightarrow \quad \boldsymbol{x}(f) = \begin{bmatrix} 5 \\ 5 \\ 1 \\ \frac{1}{6}\pi \\ -\frac{1}{12}\pi \\ \frac{1}{2}\pi \end{bmatrix} \tag{11}$$

以上のように，状態を遷移させる手順，つまり，経路や姿勢変動を自由に設計できる点が非平面マルチロータヘリコプタの特徴であり，このことを考慮して誘導制御を考えることが重要である．この特徴が制御の自由度を高くしている一方で，経路や各点での状態の最適性を考える最適化問題を複雑にする要因でもある．以上から，非平面マルチロータヘリコプタの誘導制御問題は，各時刻 t での回転運動と並進運動の最適な組み合わせを探す問題といえる．文献 [9]，[41] は，この問題について研究しており，非平面マルチロータヘリコプタの高精度な位置決めができる性質をふまえて，目標地と遠い場合には回転運動を制御し，目標値付近では並進運動を制御する制御方針を立てることで，切り替え制御手法を提案している．しかし，制御の切り替え条件は，目標地との距離のみをパラメータとしており，経路および運動の最適性については考えられていない．

モータの出力飽和や，消費電力を抑えることなど，様々な制約条件のもと，最大限のパフォーマンスを発揮することを目的とした，非平面マルチロータヘリコプタ独自の最適経路生成手法および誘導制御手法が必要となる．

参考文献

[1] ”MM 総研,” https://www.m2ri.jp/

[2] ”ドローン国内市場規模調査,” https://www.m2ri.jp/news/detail.html?id=221

[3] ”小型無人機を活用したインフラ点検の現状と課題,” 年報 NTT ファシリティーズ総研レポート，No. 27 2016.6

[4] 次世代社会インフラ用ロボット現場検証委員会 橋梁維持管理部会，”橋梁維持管理技術の現場検証・評価の結果,” 2016.3.30.

[5] Oner, K., Cetinsoy, E., and Unel, M.: ”Dynamic Model and Control of a New Quadrotor Unmanned Aerial Vehicle with Tilt-Wing Mechanism,” International Conference on Control, Automation, Robotics and Vision, 2008.

[6] Kaufman, E., Caldwell, K., Lee, D. and Lee, T.: ”Design and development of a free-floating hexrotor UAV for 6-DOF maneuvers,” Proceedings of the 2014 IEEE Aerospace Conference, pp. 1-10，2014.

[7] Brescianini，D.，D’Andrea，R., and Unel, M.: ”Design, Modeling and Control of an Omni-Directional Aerial Vehicle,” IEEE International Conference on Robotics and Automation, 2016.

[8] 清水 拓，上野 光，村上 弘記: ”6 自由度独立制御可能な飛行体の提案とその設計手法,” 第 32 回日本ロボット学会学術講演会講演論文集，3M1-05，2014.

[9] 安田 真大，伊吹 竜也，鈴木 洋史，三平 満司: ”ヘキサロータの動的可操作性に基づく切替位置・姿勢制御,” 計測自動制御学会論文集，Vol.52，No.9，pp.507-515，2016.

[10] Jiang, G. and Voyles, R.: ”Hexrotor UAV platform enabling dexterous interaction with structures-flight test,” Proceedings of the 2013 IEEE International Symposium on Sefety, Security, and Rescue Robotics, pp. 1-6，2013.

[11] Markus, R., Giuseppe, M., Francesco, P., Elisabetta, C., Gianluca, A., Fabrizio, C. and Antonio, F.: ”6D Physical Interaction with a Fully Actuated Aerial Robot,” IEEE International Conference on Robotics and Automation, 2017.

[12] 伊吹 竜也，木曽 勝之，安田 真大，三平 満司: ”動的可操作度と最大並進加速度を考慮したヘキサロータの構造最適化,” 日本機械学会論文集，Vol.83, No.846, p. 16-00206，2017.

[13] 龍野 雅裕，大橋 遼平，長谷川 直輝，鈴木 智，河村 隆，清水 拓，上野 光，村上 弘記: ”6 自由度独立に運動可能なマルチロータヘリコプタの構造最適化に関する研究,” ロボティクス・メカトロニクス講演会講演論文集，1P1-H03，2017.

[14] Kennedy, J., Eberhart, E.: ”Particle swarm optimization,” Proceedings of IEEE International Conference on Neural Networks, pp.1942-1948, 1995.

[15] ”PSoC,” http://japan.cypress.com/products/programmablesystem- chip-psoc

[16] ”Futaba,” http://www.futaba.co.jp/
[17] ”SBUS protcol,” https://developer.mbed.org/users/Digixx/notebook/futaba-s-bus-controlled-by-mbed/
[18] ”HOBBYWING,” http://www.hobbywing.com/
[19] ”TKK CSM-MG200,” https://www.gnas.jp/products/
[20] ”Prime13,” http://www.mocap.jp/optitrack/products/prime-13/
[21] 田原 誠，野波 健蔵: ”マルチロータ型ヘリコプタの機体設計と 6 発ロータ機の開発,” 第 54 回自動制御連合講演会，2A402，2011.
[22] 長谷川 直輝，鈴木 智，河村 隆，清水 拓，上野 光，村上 弘記:” 6 自由度独立制御可能な非平面マルチロータヘリコプタの開発,” 第 34 回日本ロボット学会学術講演会講演論文集，1F3-01，2016
[23] 豊橋技術科学大学高等専門学校制御工学教育連携プロジェクト: ”制御工学—技術者のための，理論・設計から実装まで,” 実教出版.
[24] Sugie, T., Yoshikawa, T.: ”Basic Structure of Two-Degree-of-Freedom Control Systems with Its Application to Servo Problem,” Transactions of the Society of Instrument and Control Engineers，Vol.22，Issue2，pp.156-161，1986.
[25] 川田 章弘: ”ロー・パス・フィルタの設計,” http://www.cqpub.co.jp/toragi/TRBN/contents/2006/tr0601/0601sp5.pdf
[26] Tayebi，A.，McGilvray，S.: ”Attitude Stabilization of a Four-Rotor Aerial Robot,” In proceedings of 43rd IEEE Conference on Decision and Control，pp14-17，2004.
[27] 小池 雄史，横山 誠: ”4 ロータ小型ヘリコプターの制御,” 第 48 回自動制御連合講演会，G2-13，2005
[28] 川田 昌克: ”MATLAB/Simulink による現代制御入門,” 森北出版株式会社.
[29] 野波 健蔵，水野 毅，他: ”制御の事典,” 朝倉書店，p100-107，p52.
[30] 鈴木 智，辛 振玉，田原 誠，小出 義郎，中澤 大輔，野波 健蔵: ”ホビークラス約 5kg 無人ヘリコプタの解析的モデリングと自律ホバリング制御,” 日本機械学会論文集 (C 編)，Vol.73，No.726，2007.
[31] 野波 健蔵，西村 秀和: ”MATLAB による制御理論の基礎,” 東京電機大学出版，p151-155.
[32] 高木 章二: ”メカトロニクスのための制御工学,” コロナ社，p210-219.
[33] ”DJI GS PRO,” https://www.dji.com/ground-station-pro
[34] ”DJI,” https://www.dji.com/
[35] 田畑 淨治，萩原 強: ”人工衛星打上げ用ロケットの誘導制御システム,” 計測と制御，17 巻 4 号，1978.
[36] 西村 敏充，川口 淳一郎: ”宇宙探査のための誘導制,” 計測と制御，30 巻 10 号，1991.
[37] 石井 崇大，鈴木 智，藤澤 陽平，岡田 伸也，飯塚 浩二郎，河村 隆: ”複数機飛行に向けた小型ヘリコプタの誘導制御系の構築,” ロボティクス・メカトロニクス講演会，1A2-G04，2013.
[38] 柴田 真充，笹岡 岳，鈴木 智，河村 隆: ”衝突回避を考慮した複数ヘリコプタの誘導制御,” 第 15 回「運動と振動の制御」シンポジウム，C24，2017.
[39] Zaki, H., Unel, M., Yildiz, Y.: ”Trajectory Control of a Quadrotor Using a Con-

trol Allocation Approach," International Conference on Unmanned Aircraft Systems (ICUAS), 2017.
[40] 野波 健蔵，赤坂 剛史: "飛躍するドローン -マルチ回転翼型無人航空機の開発と応用研究，海外動向，リスク対策まで-," NTS，p72-74.
[41] 吉岡 弘人，伊吹 竜也，三平 満司: "全駆動ヘキサロータの切り替え位置・姿勢制御実験," 第 15 回「運動と振動の制御」シンポジウム，B19，2017.