# TTK4190 Guidance and Control of Vehicles: Assignment 3

Edvard Grødem, Jae Hyeong Hwang and Michael Hoyer

October 2020

# 1 Problem 1

## 1.1 (a)

(1)

$$\rho_m = m/LBH; \tag{2}$$

$Iz_{fun} = @(x, y, z)(x.^2 + y.^2) * rho_m$; $Iz = integral3(Iz_{fun}, -L/2, L/2, -B/2, B/2, -H/2, H/2)$; $I_z$ about center of gravity is 3.7544e+10

Products of inertia is a kind of a moment of inertia but considers both axis. Because the x axis and y axis of prism is in symmetry, $I_{xy} = I_{yx} = 0$

## 1.2 (b)

According to the Huygens-Steiner's Parallel-Axis Theorem,

$$I_b^b = m(((r_g^b)^T r_{bg}^b I_3 - r_{bg}^b (r_{bg}^b)^T) \tag{3}$$

So $I_z$ respect to CO is $I_z^{CG} + m(x^2 + y^2)$, which is about 3.7777e+10

ratio between the moments of inertia for the prism and the real ship is about 0.5753

## 1.3 (c)

## 1.4 (d)

Skew-symmetric property is useful when designing a nonlinear motion control system since the quadratic form $\nu^{top} C_{RB}(\nu)\nu \equiv 0$ This is exploited in energy-based designs where Lyapunov functions play a key role.[1]

## 1.5 (e)

Since ocean current is assumed to be irrotational and $C_{RB}$ does not depend on linear velocity Coroiolis matrix is useful.

# 2 Problem 2

## 2.1 a)

Using Archimedes law:

$$mg = \rho g \Delta$$

Solving for $\Delta$

$$\Delta = \frac{m}{\rho}$$

This gives us a numeric value of

$$\Delta = \frac{17.0677 * 10^6 kg}{1025 kg/m^3} = 16651.41 m^3$$

## 2.2 b)

Assuming roll and pitch angle low we can use

$$A_{wp} = LB$$

This gives us the numeric value

$$A_{wp} = 161m * 21.8m = 3509.8m^2$$

Since the $A_{wp}$ is constant along heave we have that

$$Z_{ns} = -\rho g A_{wp} z^n$$

This gives the numeric value

$$Z_{ns} = -1025 \frac{kg}{m^3} * 9.81 \frac{m}{s^2} * 3509.8m^2 * z^n = 35.283 z^n MN$$

## 2.3 c)

$$T_3 = 2\pi \sqrt{\frac{2T}{g}}$$

Inserting numeric values gives us

$$T_3 = 2\pi \sqrt{\frac{2 * 8.9}{9.81}} s = 8.46s$$

## 2.4 d)

$$GM_T = \frac{I_T}{\Delta}$$

$$GM_L = \frac{I_L}{\Delta}$$

Since we are calculating on a prism we have that

$$I_L = \frac{BL^3}{12}$$

$$I_T = \frac{LB^3}{12}$$

This gives us

$$GM_T = 455.30m$$

$$GM_L = 8.34m$$

## 2.5 e)

The ship is metacentrically stable because

$$GM_T >> 0$$

and

$$GM_L >> 0$$

# 3 Problem 3

## 3.1 Task a

By assuming assuming small values for heave-roll-pitch as well as no coupling between them and the rest of the system, we can rewrite M as

$$M_A = \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix} \tag{4}$$

By further assuming no coupling between surge and sway-yaw, as well as a symmetric added mass matrix we can simplify $M_A$ to

$$M_A = \begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & Y_{\dot{r}} & N_{\dot{r}} \end{bmatrix} \tag{5}$$

where $X_{\dot{u}}, Y_{\dot{v}}, Y_{\dot{r}} and N_{\dot{r}}$ are coefficients describing the coupling between acceleration of the vessel and the force applied to the water surrounding it.

## 3.2 Task b

By using property 6.2 to define $C_A$ in terms of $M_A$ we can write it as

$$\mathbf{C}_A(\nu_r) = \begin{bmatrix} 0 & 0 & a2 \\ 0 & 0 & -a1 \\ -a2 & a1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} v_r + Y_{\dot{r}} r \\ 0 & 0 & -X_{\dot{u}} u_r \\ -Y_{\dot{v}} v_r - Y_{\dot{r}} r & X_{\dot{u}} u_r & 0 \end{bmatrix} \tag{6}$$

# 4 Problem 4

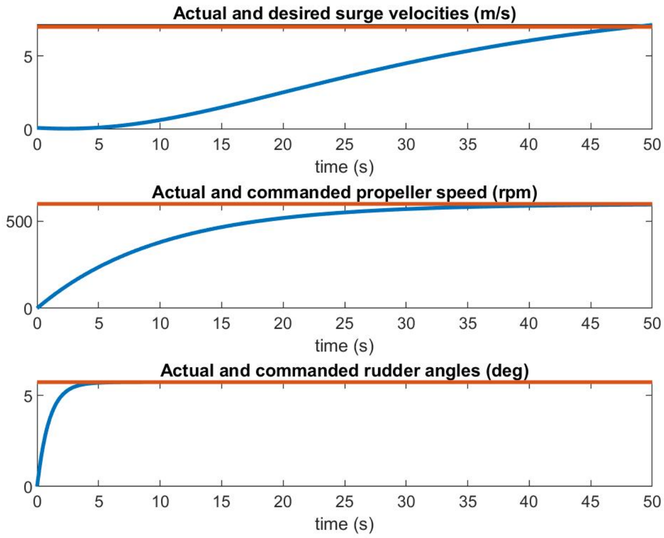


Figure 1: Final plots part 1

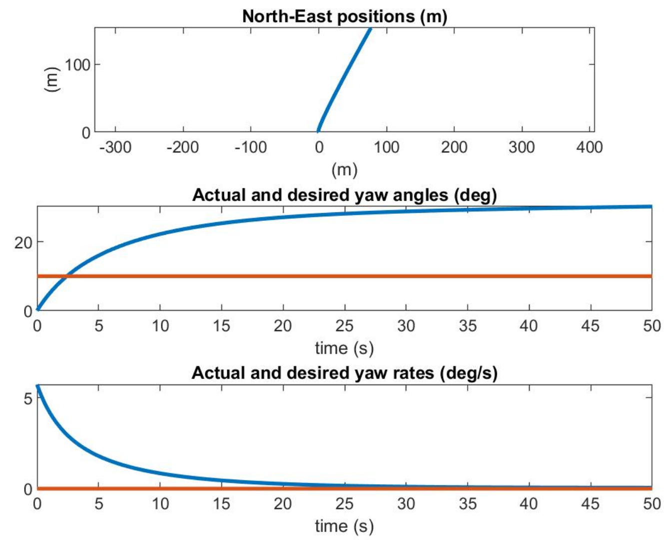


Figure 2: Final plots part 1

```
% Project in TTK4190 Guidance and Control of Vehicles
%
% Author:           My name
% Study program:    My study program
addMass = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% USER INPUTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h  = 0.05;    % sampling time [s]
Ns = 1000;    % no. of samples
```

```matlab
psi_ref = 10 * pi/180;  % desired yaw angle (rad)
u_ref   = 7;            % desired surge speed (m/s)

% ship parameters
m = 17.0677e6;          % mass (kg)
Iz = 2.1732e10;         % yaw moment of inertia (kg m^3)
xg = -3.7;              % CG x-ccordinate (m)
L = 161;                % length (m)
Beam = 21.8;               % beam (m)
T = 8.9;                % draft (m)
KT = 0.7;               % propeller coefficient (-)
Dia = 3.3;              % propeller diameter (m)
rho = 1025;             % density of water (m/s^3)
kin_visc = 1e-6;        % kinematic viscosity (m/s^2)

% damping
T1 = 20;                                          % (s)
T2 = 20;                                          % (s)
T6 = 10;                                          % (s)
S = L * Beam;      % (m^2)
k = 0.1;
Cr = 0;
epsilon = 0.001;

% rudder limitations
delta_max  = 40 * pi/180;        % max rudder angle      (rad)
Ddelta_max = 5  * pi/180;        % max rudder derivative (rad/s)

% added mass matrix
Xudot = -8.9830e5;
Yvdot = -5.1996e6;
Yrdot =  9.3677e5;
Nvdot =  Yrdot;
Nrdot = -2.4283e10;

MA = - [  Xudot    0       0;
        0        Yvdot   Yrdot;
        0        Yrdot   Nrdot];

CA = @(u, v, r) [           0              0          Yvdot*v + Yrdot*r;
                            0              0              -Xudot*u;
                 -Yvdot*v-Yrdot*r       Xudot*u              0];



% rigid-body mass matrix
MRB = [ m 0      0
        0 m     m*xg
        0 m*xg  Iz  ];
if addMass == 1
    Minv = inv(MRB + MA);
else
    Minv = inv(MRB);
end

% linear damping matrix
```

```
Xu = - (m - Xudot) / T1;
Yv = - (m - Yvdot) / T2;
Nr = - (Iz - Nrdot) / T6;
D = - diag([Xu Yv Nr]);

% nonlinear damping param
Cf = @(ur) 0.075 / (((log(L*abs(ur)/kin_visc)) - 2)^2 + epsilon);
Cd = Hoerner(Beam, T);

% input matrix
t_thr = 0.05;          % thrust deduction number
X_delta2 = 0;          % rudder coefficients (Section 9.5)
Y_delta = 0;
N_delta = 1;
B = [ (1-t_thr)  X_delta2
        0        Y_delta
        0        N_delta  ];

% initial states
eta = [0 0 0]';
nu  = [0.1 0 0.1]';
delta = 0;
n = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MAIN LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
simdata = zeros(Ns+1,14);                  % table of simulation data

for i=1:Ns+1

    t = (i-1) * h;                         % time (s)

    % state-dependent time-varying matrices
    CRB = m * nu(3) * [ 0 -1 -xg
                        1  0  0
                        xg 0  0  ];
    if addMass == 1
        C = CRB + CA(nu(1), nu(2), nu(3));
    else
        C = CRB;
    end
    R = Rzyx(0,0,eta(3));

    % reference models
    psi_d = psi_ref;
    r_d = 0;
    u_d = u_ref;

    % thrust
    thr = rho * Dia^4 * KT * abs(n) * n;    % thrust command (N)

    % control law
    delta_c = 0.1;             % rudder angle command (rad)
    n_c = 10;                  % propeller speed (rps)
```

```
    % damping
    u_r = nu(1);
    v_r = nu(2);
    r = nu(3);
    X = - 0.5 * rho * S * (1+k) * Cf(u_r) * u_r * abs(u_r);

    Yh = 0;                    % init
    Nh = 0;                    % init
    dx = L/10;                 % 10 strips
    for xL = -L/2:dx:L/2
        Ucf = abs(v_r + xL*r)*(v_r + xL*r);
        Yh = Yh - 0.5*rho*T*Cd*Ucf*dx;          % sway force
        Nh = Nh - 0.5*rho*T*Cd*xL*Ucf*dx;       % yaw moment
    end

    t_damp = - [X Yh Nh]';

    % ship dynamics
    u = [ thr delta ]';
    tau = B * u;
    nu_dot = Minv * (tau - C * nu - D * nu - t_damp);     % added linear damping
    eta_dot = R * nu;

    % Rudder saturation and dynamics (Sections 9.5.2)
    if abs(delta_c) >= delta_max
        delta_c = sign(delta_c)*delta_max;
    end

    delta_dot = delta_c - delta;
    if abs(delta_dot) >= Ddelta_max
        delta_dot = sign(delta_dot)*Ddelta_max;
    end

    % propeller dynamics
    n_dot = (1/10) * (n_c - n);

    % store simulation data in a table (for testing)
    simdata(i,:) = [t n_c delta_c n delta eta' nu' u_d psi_d r_d];

    % Euler integration
    eta = euler2(eta_dot,eta,h);
    nu  = euler2(nu_dot,nu,h);
    delta  = euler2(delta_dot,delta,h);
    n  = euler2(n_dot,n,h);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PLOTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t       = simdata(:,1);                 % s
n_c     = 60 * simdata(:,2);            % rpm
delta_c = (180/pi) * simdata(:,3);      % deg
n       = 60 * simdata(:,4);            % rpm
delta   = (180/pi) * simdata(:,5);      % deg
x       = simdata(:,6);                 % m
```

```
y        = simdata(:,7);                    % m
psi      = (180/pi) * simdata(:,8);         % deg
u        = simdata(:,9);                    % m/s
v        = simdata(:,10);                   % m/s
r        = (180/pi) * simdata(:,11);        % deg/s
u_d      = simdata(:,12);                   % m/s
psi_d    = (180/pi) * simdata(:,13);        % deg
r_d      = (180/pi) * simdata(:,14);        % deg/s

figure(1)
figure(gcf)
subplot(311)
plot(y,x,'linewidth',2); axis('equal')
title('North-East positions (m)'); xlabel('(m)'); ylabel('(m)');
subplot(312)
plot(t,psi,t,psi_d,'linewidth',2);
title('Actual and desired yaw angles (deg)'); xlabel('time (s)');
subplot(313)
plot(t,r,t,r_d,'linewidth',2);
title('Actual and desired yaw rates (deg/s)'); xlabel('time (s)');


figure(2)
figure(gcf)
subplot(311)
plot(t,u,t,u_d,'linewidth',2);
title('Actual and desired surge velocities (m/s)'); xlabel('time (s)');
subplot(312)
plot(t,n,t,n_c,'linewidth',2);
title('Actual and commanded propeller speed (rpm)'); xlabel('time (s)');
subplot(313)
plot(t,delta,t,delta_c,'linewidth',2);
title('Actual and commanded rudder angles (deg)'); xlabel('time (s)');
```

# References

[1] Thor.I.Fossen, *HANDBOOK OF MARINE CRAFT HYDRODYNAMICS AND MOTION CONTROL*. John Wiley Sons, 2011, 2021.