TTK4190 Guidance and Control of Vehicles

Assignment 3, Part 3

Written Fall 2020 By Magnus Dyre-Moe, Patrick Nitschke and Siawash Naqibi

1 Propeller Revolution and Speed Control

1.1 Problem a)

We find K_T and K_Q with the *wageningen*() function from the MSS-Toolbox. We assume bollard pull, such that $J_a = 0$. Furthermore, pitch/diameter ratio is known to be $PD = 1.5$, blade area ratio is $AEAO = 0.65$, and number of propeller blades is $z = 4$. Inserting this into Matlab, **wageningen(Ja,PD,AEAO,z)** yields $K_T = 0.6367$ and $K_Q = 0.1390$.

1.2 Problem b)

The formulas for thrust T and torque Q assuming bollard pull are,

$$\begin{aligned} T &= \rho D^4 K_T |n| n \\ Q &= \rho D^5 K_Q |n| n \end{aligned} \tag{1}$$

Where ρ is the density of water, D is the diameter of the propeller, K_T and K_Q are obtained through section 1.1 and n is the propeller speed. These formulas are listed as equation 9.7 and 9.8 in Fossen.

Equations 9.33 and 9.35 in Fossen are,

$$\begin{aligned} I_m \dot{n} &= Q_m - Q - Q_f \\ H(s) &= \frac{Q_m}{Y}(s) \approx \frac{K}{Ts+1} e^{-\tau s} \end{aligned} \tag{2}$$

Where for our case, Y is defined as $Y = \frac{1}{K_m} Q_d(n_d)$. Furthermore, Q_m is the torque developed by the main engine, Q_f is frictional torque and Q is defined in equation 1. We are also given values for I_m, K_m, T_m and τ.

1.3 Problem c)

Following equation 1, the desired moment is defined as,

$$Q_d = \rho D^5 K_Q |n_d| n_d \tag{3}$$

Where n_d is the desired propeller speed. See appendix, line 282.

1.4 Problem d)

Equation 6.136 in Fossen is given as

$$(m - X_{\dot{u}})\dot{u}_r - X_u u_r = -X_{\delta\delta}\delta^2 + (1-t)T \tag{4}$$

This equation reduces to the cruise speed equation below, when *velocity is constant* and *rudder angle is zero.*

$$U = \frac{(t-1)T}{X_u} \tag{5}$$

Hence, $\dot{u}_r = 0$ and $\delta^2 = 0$.

1.5 Problem e)

Given a desired cruise speed U_d we can find desired torque through equation, 5

$$T_d = \frac{U_d X_u}{t-1} \tag{6}$$

Furthermore, we can find the desired propeller speed through equation 1,

$$n_d = \text{sign}(T_d)\sqrt{\frac{|T_d|}{\rho D^4 K_T}} \tag{7}$$

Now, we are able to find desired torque through equation 3. Furthermore, by using 2 we are able to calculate Q_m. Q is known by the actual propeller speed and Q_f is assumed zero.

The result of the controller can be viewed in figure 1. As observed, the desired speed is obtained when heading angle is constant.

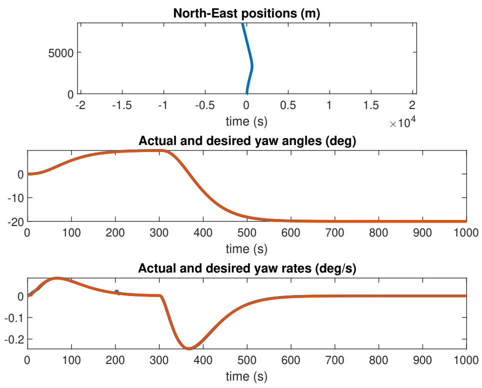
1.6 Problem f)

In our case, we have a controller for surge velocity by propeller speed. We also have heading control. By using heading control we are able to control sideslip. Hence, we will only have surge velocity. By not using a heading control, sideslip will be induced. When we have sideslip the velocity in sway is non-zero. If so, the velocity of the ship can be modelled into surge and sway velocities. For non-zero sway velocity there will be added cross-flow-damping, reducing the velocity of the vessel.

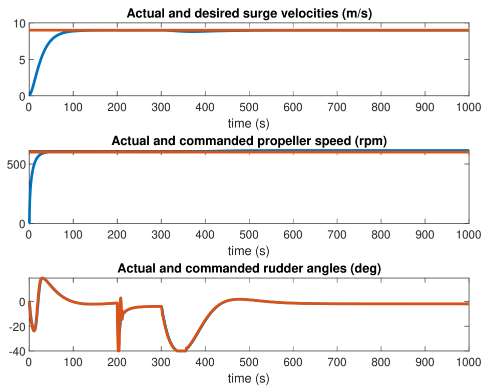
Hence, the speed will drop for cross-flow-damping, meaning we do not have an autopilot in heading. However, the velocity will remain constant when sideslip is zero, meaning an autopilot in heading is used.

1.7 Plots with speed autopilot

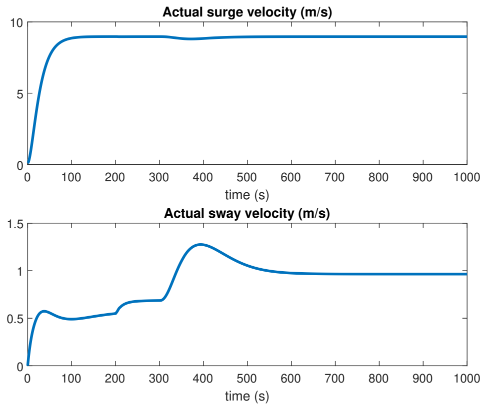
Plots can be found in figure 1.



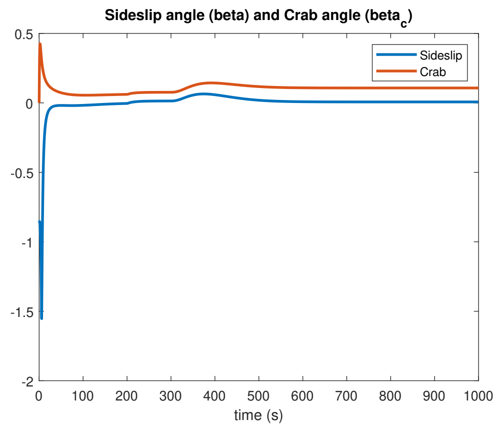
(a)



(b)



(c)



(d)

Figure 1: Figure 1a shows position of the vessel along with actual and desired yaw angles and yaw rates. Figure 1b displays actual and desired surge velocity and actual and commanded propeller speed and rudder angle. Figure 1c shows actual velocities in surge and sway. Figure 1d shows sideslip angle and crab angle.

A Matlab code

```
% Project in TTK4190 Guidance and Control of Vehicles
%
% Author:          Magnus Dyre-Moe, Patrick Nitschke and Siawash Naqibi
% Study program:   Cybernetics and Robotics

clear;
clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% USER INPUTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h  = 0.1;    % sampling time [s]
Ns = 10000;  % no. of samples

psi_ref = 10 * pi/180;  % desired yaw angle (rad)
U_d = 9;                % desired cruise speed (m/s)

% ship parameters
m = 17.0677e6;          % mass (kg)
Iz = 2.1732e10;         % yaw moment of inertia about CO (kg m^3)
xg = -3.7;              % CG x-ccordinate (m)
L = 161;                % length (m)
B = 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 (kg/m^3)
visc = 1e-6;            % kinematic viscousity at 20 degrees (m/s^2)
eps = 0.001;            % a small number added to ensure that the denominator of ...
    Cf is well defined at u=0
k = 0.1;                % form factor giving a viscous correction
t_thr = 0.05;           % thrust deduction number


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

% added mass matrix about CO
Xudot = -8.9830e5;
Yvdot = -5.1996e6;
Yrdot =  9.3677e5;
Nvdot =  Yrdot;
Nrdot = -2.4283e10;
MA = -[ Xudot 0     0
        0 Yvdot Yrdot
        0 Nvdot Nrdot ];

% rigid-body mass matrix
MRB = [ m 0    0
        0 m    m*xg
        0 m*xg Iz ];

Minv = inv(MRB + MA); % Added mass is included to give the total inertia

% ocean current in NED
Vc = 1;                             % current speed (m/s)
betaVc = deg2rad(45);               % current direction (rad)

% wind expressed in NED
Vw = 10;                   % wind speed (m/s)
betaVw = deg2rad(135);     % wind direction (rad)
rho_a = 1.247;             % air density at 10 deg celsius
cy = 0.95;                 % wind coefficient in sway
cn = 0.15;                 % wind coefficient in yaw
A_Lw = 10 * L;             % projected lateral area

```

```
% linear damping matrix (only valid for zero speed)
T1 = 20; T2 = 20; T6 = 10;

Xu = -(m - Xudot) / T1;
Yv = -(m - Yvdot) / T2;
Nr = -(Iz - Nrdot)/ T6;
D = diag([-Xu -Yv -Nr]); % zero speed linear damping


% rudder coefficients (Section 9.5)
b = 2;
AR = 8;
CB = 0.8;

lambda = b^2 / AR;
tR = 0.45 - 0.28*CB;
CN = 6.13*lambda / (lambda + 2.25);
aH = 0.75;
xH = -0.4 * L;
xR = -0.5 * L;

X_Δ2 = 0.5 * (1 - tR) * rho * AR * CN;
Y_Δ = 0.25 * (1 + aH) * rho * AR * CN;
N_Δ = 0.25 * (xR + aH*xH) * rho * AR * CN;

% input matrix
Bu = @(u_r,Δ) [ (1-t_thr)  -u_r^2 * X_Δ2 * Δ
                        0       -u_r^2 * Y_Δ
                        0       -u_r^2 * N_Δ            ];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Heading Controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Linearlized coriolis matrices
CRBstar = [ 0 0 0
        0 0 m*U_d
        0 0 m*xg*U_d];
CRBstar = CRBstar(2:3,2:3); % reduced to sway and yaw

CAstar = [   0    0                    0
        0    0              -Xudot*U_d
        0    (Xudot-Yvdot)*U_d    -Yrdot*U_d];
CAstar = CAstar(2:3,2:3); % reduced to sway and yaw

% Reduced D matrix
D_reduced = D(2:3,2:3);

% Reduced M
Minv_reduced = Minv(2:3,2:3); % 2 by 2


% linearized sway-yaw model (see (7.15)-(7.19) in Fossen (2021)) used
% for controller design. The code below should be modified.
N_lin = CRBstar + CAstar + D_reduced;
b_lin = [-2*U_d*Y_Δ -2*U_d*N_Δ]';

% initial states
eta = [0 0 0]';
nu  = [0.1 0 0]';
Δ = 0;
n = 0;
z = 0;
xd = [0; 0; 0];

% Tranfer function from Δ to r
[num,den] = ss2tf(-Minv_reduced * N_lin, Minv_reduced * b_lin, [0 1], 0);
root = roots(den);
T1 = -1/root(1);
```

```
T2 = -1/root(2);
T3 = num(2)/num(3);
T_nomoto = T1 + T2 - T3;
K_nomoto = num(3)/(root(1)*root(2));


% rudder control law
wb = 0.06;
zeta = 1;
wn = 1 / sqrt( 1 - 2*zeta^2 + sqrt( 4*zeta^4 - 4*zeta^2 + 2) ) * wb;

m_nomoto = T_nomoto / K_nomoto;
d = 1 / K_nomoto;
Kp = wn^2 * m_nomoto;
Kd = ( 2 * zeta * wn * T_nomoto - 1 ) / K_nomoto;
Ki = wn^3 / 10 * m_nomoto;
w_ref = 0.03;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PART 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Propeller coefficients
num_blades = 4;         % number of propeller blades
AEAO = 0.65;            % area of blade
PD = 1.5;               % pitch/diameter ratio
Ja = 0;                 % bollard pull
[KT, KQ] = wageningen(Ja, PD, AEAO, num_blades); %Propeller coefficients

Qm = 0;
t_T = 0.05;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MAIN LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
simdata = zeros(Ns+1,16);                % table of simulation data

for i=1:Ns+1
    t = (i-1) * h;                      % time (s)

    % Reference model
    if (t > 300)
        psi_ref = deg2rad(-20);
    else
        psi_ref = deg2rad(10);
    end
    Ad = [0     1                   0;
          0     0                   1;
          -w_ref^3 -(2*zeta+1)*w_ref^2 -(2*zeta+1)*w_ref];

    Bd = [0; 0; w_ref^3];

    xd_dot = Ad * xd + Bd * psi_ref;

    % Rotation from body to NED
    R = Rzyx(0,0,eta(3));

    % current (should be added here)
    nu_r = nu - [Vc*cos(betaVc - eta(3)), Vc*sin(betaVc - eta(3)), 0]';
    u_c = Vc*cos(betaVc - eta(3));

    % wind (should be added here)
    if t > 200
        u_rw = nu(1) - Vw * cos(betaVw - eta(3));
        v_rw = nu(2) - Vw * sin(betaVw - eta(3));
        V_rw = sqrt(u_rw^2 + v_rw^2);
        gamma_w = -atan2(v_rw, u_rw);
        Cy = cy * sin( gamma_w );
```

```
        Cn = cn * sin( 2*gamma_w );
        Ywind = 0.5 * rho_a * V_rw^2 * Cy * A_Lw; % expression for wind moment in ...
            sway should be added.
        Nwind = 0.5 * rho_a * V_rw^2 * Cn * A_Lw * L; % expression for wind moment...
             in yaw should be added.
    else
        Ywind = 0;
        Nwind = 0;
    end
    tau_env = [0 Ywind Nwind]';

    % state-dependent time-varying matrices
    CRB = m * nu(3) * [ 0 -1 -xg
                        1  0  0
                        xg 0  0  ];

    % coriolis due to added mass
    CA = [  0    0   Yvdot * nu_r(2) + Yrdot * nu_r(3)
            0    0   -Xudot * nu_r(1)
          -Yvdot * nu_r(2) - Yrdot * nu_r(3)     Xudot * nu_r(1)    0];
    N = CRB + CA + D;

    % nonlinear surge damping
    Rn = L/visc * abs(nu_r(1));
    Cf = 0.075 / ( (log(Rn) - 2)^2 + eps);
    Xns = -0.5 * rho * (B*L) * (1 + k) * Cf * abs(nu_r(1)) * nu_r(1);

    % cross-flow drag
    Ycf = 0;
    Ncf = 0;
    dx = L/10;
    Cd_2D = Hoerner(B,T);
    for xL = -L/2:dx:L/2
        vr = nu_r(2);
        r = nu_r(3);
        Ucf = abs(vr + xL * r) * (vr + xL * r);
        Ycf = Ycf - 0.5 * rho * T * Cd_2D * Ucf * dx;
        Ncf = Ncf - 0.5 * rho * T * Cd_2D * xL * Ucf * dx;
    end
    d = -[Xns Ycf Ncf]';

    % reference models
    psi_d = xd(1);
    r_d = xd(2);
    u_d = U_d;

    % thrust
    thr = rho * Dia^4 * KT * abs(n) * n;     % thrust command (N) Equation 9.7

    % control law
    z_dot = eta(3) - xd(1);
    Δ_c = - ( Kp * (eta(3) - xd(1)) + Kd * (nu(3) - xd(2)) + Ki * z ); ...
                    % rudder angle command (rad)

    % ship dynamics
    u = [ thr Δ ]';
    tau = Bu(nu_r(1),Δ) * u;
    nu_dot = Minv * (tau_env + tau - N * nu_r - d);
    eta_dot = R * nu;

    % Rudder saturation and dynamics (Sections 9.5.2)
    if abs(Δ_c) ≥ Δ_max
        Δ_c = sign(Δ_c)*Δ_max;
    end

    Δ_dot = Δ_c - Δ;
    if abs(Δ_dot) ≥ DΔ_max
        Δ_dot = sign(Δ_dot)*DΔ_max;
    end

```

```matlab
    % propeller dynamics
    Im = 100000; Tm = 10; Km = 0.6;         % propulsion parameters
    n_c = 10;                               % propeller speed (rps)

    T_prop = rho * Dia^4 * KT * abs(n) * n;
    Q_prop = rho * Dia^5 * KQ * abs(n) * n;
    T_d = (U_d - u_c)*Xu / (t_T - 1);
    n_d = sign(T_d) * sqrt( abs(T_d) / (rho*Dia^4*KT) );
    Q_d = rho * Dia^5 * KQ * abs(n_d) * n_d;
    Y = (1 / Km) * Q_d;
    Qm_dot = (1 / Tm) * (-Qm + Y*Km);
    Qf = 0;

    n_dot = (1/Im) * (Qm - Q_prop - Qf);            % should be changed in Part 3

    % Crab and sideslip to be stored in simdata
    sideslip_angle = asin( nu_r(2) / sqrt(nu_r(1)^2 + nu_r(2)^2) );
    crab_angle = atan( nu(2) / nu(1) );

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

    % Euler integration
    eta = euler2(eta_dot,eta,h);
    nu  = euler2(nu_dot,nu,h);
    Δ = euler2(Δ_dot,Δ,h);
    n  = euler2(n_dot,n,h);
    z = euler2(z_dot,z,h);
    xd = euler2(xd_dot,xd,h);
    Qm = euler2(Qm_dot,Qm,h);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLOTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t       = simdata(:,1);                 % s
n_c     = 60 * simdata(:,2);            % rpm
Δ_c = (180/pi) * simdata(:,3);      % deg
n       = 60 * simdata(:,4);            % rpm
Δ   = (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
sideslip = simdata(:,15);
crab    = simdata(:,16);

figure(1)
figure(gcf)
subplot(311)
plot(y,x,'linewidth',2); axis('equal')
title('North-East positions (m)'); xlabel('time (s)');
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,Δ,t,Δ_c,'linewidth',2);
title('Actual and commanded rudder angles (deg)'); xlabel('time (s)');

figure(3)
figure(gcf)
subplot(211)
plot(t,u,'linewidth',2);
title('Actual surge velocity (m/s)'); xlabel('time (s)');
subplot(212)
plot(t,v,'linewidth',2);
title('Actual sway velocity (m/s)'); xlabel('time (s)');

figure(4)
figure(gcf)
subplot(111)
plot(t, sideslip, t, crab, 'linewidth', 2)
title('Sideslip angle (beta) and Crab angle (beta_c)'); xlabel('time (s)')
legend('Sideslip', 'Crab')
```

References