TTK4190 Guidance and Control of Vehicles

Assignment 2, Pt. 2

Written Fall 2023

Seknaya, Sacit Ali
sacitas@stud.ntnu.no

Kopperstad, Håvard Olai
haavarok@stud.ntnu.no

Almenningen, Elias Olsen
eliasoa@stud.ntnu.no

October 31, 2023

Problem 1 - Environmental Disturbances

a)

The code is modified to include 2-D irrotational ocean currtent in surge and sway:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 2, 1a) Add current disturbance here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
beta_vs = deg2rad(45);                 % current direction in rad
Vc = 1;                                % Current speed m/s
uc = Vc*cos(beta_vs - x(6));
vc = Vc*sin(beta_vs - x(6));
nu_c = [ uc vc 0 ]';
```

b)

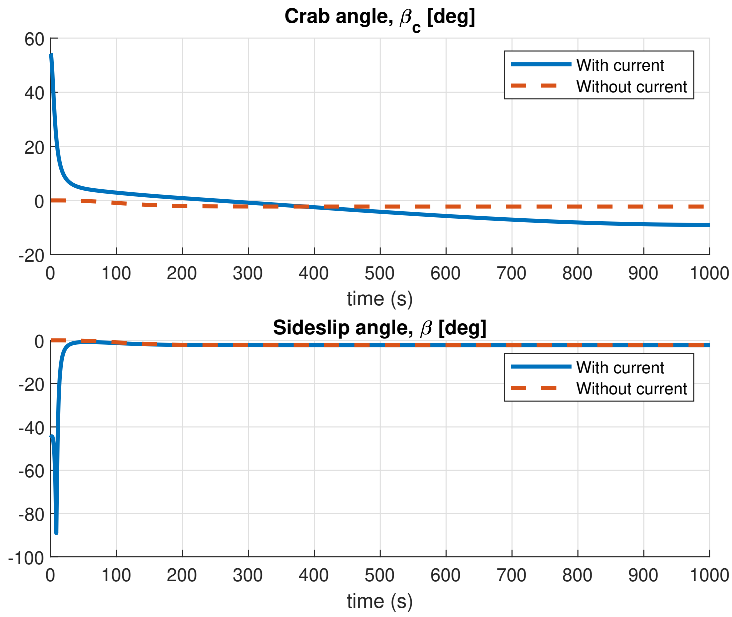


Figure 1: Comparing sideslip and crab angles when there is no current vs current

If considering our ship when there are no ocean currents, we expect the sideslip angle β and the crab ange β_c to be equal [1]:

$$\beta_c = \beta = sin^{-1}\left(\frac{v}{U}\right) \tag{1}$$

From Figure 1 it is possible to see that the sideslip and crab angles are equal when there are no currents.

When introducing a constant irrotational ocean current, the angles are expected to be different from each other. By looking at Figure 1 it can be verified that is indeed the case.

c)

The code is updated to include wind moments Y_{wind} and N_{wind} occuring after 200 seconds:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 2, 1c) Add wind disturbance here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vw = 10;                               % wind speed m/s
```

```
    beta_Vw = deg2rad(135);              % wind direction
    gamma_w = x(6) - beta_Vw - pi;       % wind angle of attack

    u_w = Vw*cos(beta_Vw - x(6));        % wind x speed
    v_w = Vw*sin(beta_Vw - x(6));        % wind y speed
    u_rw = x(1) - u_w;                   % relatice x speed
    v_rw = x(2) - v_w;                   % relatice y speed
    gamma_rw = atan2(v_rw,u_rw);         % relatice angle of attack angle

    cy = 0.95;                           % wind coefficients
    cn = 0.15;                           % wind coefficients
    ALw = 10*L_oa;                       % latteral projected area

    C_Y = cy * sin(gamma_rw);            % wind coefficient
    C_N = cn * sin(2*gamma_rw);          % wind coefficient

    V_rw = sqrt(u_rw^2 + v_rw^2);        % relative wind speed
    if t > 200
        Ywind = 0.5*rho_a*V_rw^2*C_Y*ALw;       %
        Nwind = 0.5*rho_a*V_rw^2*C_N*ALw*L_oa; %
    else
        Ywind = 0;
        Nwind = 0;
    end
    tau_wind = [0 Ywind Nwind]';
```

Problem 2 - Heading Autopilot

a)

Linearizing the nonlinear Coriolis forces $\boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}$ about $r = u = 0$ and $u = u_d$:

$$\begin{aligned}
\boldsymbol{C}_{RB}\boldsymbol{\nu} &= \begin{bmatrix} 0 & -mr & -mrx_g \\ mr & 0 & 0 \\ mrx_g & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \\
&= \begin{bmatrix} -vmr - mr^2x_g \\ mru \\ mrux_x \end{bmatrix} := \begin{bmatrix} a \\ b \\ c \end{bmatrix}
\end{aligned}$$

The linerized forces are defined as

$$\begin{aligned}
\boldsymbol{C}^*_{RB} &= \begin{bmatrix} \frac{\partial a}{\partial u} & \frac{\partial a}{\partial v} & \frac{\partial a}{\partial r} \\ \frac{\partial b}{\partial u} & \frac{\partial b}{\partial v} & \frac{\partial b}{\partial r} \\ \frac{\partial c}{\partial u} & \frac{\partial c}{\partial v} & \frac{\partial c}{\partial r} \end{bmatrix} |u = r = 0,\, u = u_d \qquad (2) \\
&= \begin{bmatrix} 0 & -mr & -vm - 2mrx_g \\ mr & mru & mu \\ mrx_g & 0 & mux_g \end{bmatrix} |u = r = 0,\, u = u_d \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & mu_d \\ 0 & 0 & mu_dx_g \end{bmatrix}
\end{aligned}$$

Linearizing the nonlinear Coriolis forces $\boldsymbol{C}_A(\boldsymbol{\nu})\boldsymbol{\nu}$ about $r = u = 0$ and $u = u_d$:

$$\begin{aligned}
\boldsymbol{C}_A(\boldsymbol{\nu})\boldsymbol{\nu} &= \begin{bmatrix} 0 & 0 & Y_{\dot v}v + Y_{\dot r}r \\ 0 & 0 & -X_{\dot u}u \\ -Y_{\dot v}v - Y_{\dot r}r & X_{\dot u}u & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \\
&= \begin{bmatrix} Y_{\dot v}vr + Y_{\dot r}r^2 \\ -X_{\dot u}ur \\ -Y_{\dot v}vu - Y_{\dot r}ru + X_{\dot u}uv \end{bmatrix}
\end{aligned}$$

Performing the same operations as in Equation 2 gives

$$\boldsymbol{C}^*_A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -X_{\dot u}u_d \\ 0 & -Y_{\dot v}u_d + X_{\dot u}u_d & Y_{\dot r}u_d \end{bmatrix} \qquad (3)$$

b)

The linearized sway-yaw maneuvering model can be written as

$$\begin{aligned}
M &:= \boldsymbol{M}_{RB} + \boldsymbol{M}_A \\
N &:= \boldsymbol{C}^*_{RB} + \boldsymbol{C}^*_A + \boldsymbol{D} \\
\dot\nu_r &= \underbrace{-\boldsymbol{M}^{-1}\boldsymbol{N}}_{\boldsymbol{A}}\,\boldsymbol{\nu}_r + \underbrace{\boldsymbol{M}^{-1}\boldsymbol{b}}_{\boldsymbol{B}}\,\delta \\
y &= \underbrace{[0,\, 1]}_{\boldsymbol{C}}\,\boldsymbol{\nu}_r + \underbrace{0}_{\boldsymbol{D}}\,\delta
\end{aligned}$$

which again can be transformed into a transfer function using matlab as:

```
%% Problem 2 b
% Maa hente ut sway og yaw
Minv = Minv(2:3,2:3);
N = C(2:3,2:3) + D(2:3,2:3);
```

```
b = 2*ud*[-Y_∆;-N_∆];
% Make the state space
A = Minv*(-N);
B = Minv*b;
C = [0 1];
D = 0;

[num,den] = ss2tf(A,B,C,D)
```

giving us the numeric transfer function

$$\frac{r}{\delta} = \frac{8.683e-5s + 0.615e-5}{s^2 + 0.1506s + 0.0008} \tag{4}$$

c)

The second-order Laplace transformated Nomoto model yields

$$\frac{r}{\delta}(s) = \frac{K(T_3 s + 1)}{(T_1 s + 1)(T_2 s + 1)} \tag{5}$$

The numerical values for the time constant T_1, T_2, T_3 and the gain K can be found by comparing the numerator and denominator in 4 with 5. By defining the equivalent time constant as $T := T_1 + T_2 - T_3$, the second-order Nomoto model can be approximated as a first-order Nomoto model:

$$\frac{r}{\delta}(s) = \frac{K}{Ts + 1} \tag{6}$$

```
%% Problem 2c
poles = roots(den);

T1 = -1/poles(1);
T2 = -1/poles(2);

K = num(3) * T1 * T2;

T3 = num(2) * (T1 * T2)/K;

T = T1 + T2 - T3;
```

The numerical values for T and K corresponding to a first-order Nomoto model are $T = 169.54$ and $K = 0.0075$.

d)

The PID controller for the heading autopilot can be found using Algorithm 15.1 in [1]. A third order reference model to generate the desired yaw angle and yaw rate is also used. The derivation is done in the code below:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 2, 2d) Add the heading controller here
e_psi       = ssa(x(6) - psi_d);            % error heading
e_r         = x(3) - r_d;                   % error yaw rate
e_int       = e_int + h * e_psi;            % integral of heading error
∆_c         = PID_heading(e_psi,e_r,e_int); % rudder angle command (rad)
% ∆_c = 0.1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function ∆_c = PID_heading(e_psi,e_r,e_int)
```

```matlab

w_b      = 0.06;              % bandwith
zeta     = 1;                 % relative damping ratio

K        = 0.0075;            % NOMOTO gain
T        = 169.549327910636;  % NOMOTO time constant

m        = T/K;               %
d        = 1/K;               %
k        = 0;                 %

% Compute the natural frequency
w_n      = 1/sqrt(1 - 2*zeta^2 + sqrt(4*zeta^4 - 4*zeta^2 +2)) * w_b;

Kp       = m*w_n^2-k;         % Compute the P gain
Kd       = 2*zeta*w_n*m - d;  % Compute the D gain;
Ki       = w_n/10 * Kp;       % Compute the I gain;

% PID control law
Δ_c = -( Kp*e_psi + Kd * e_r + Ki * e_int );
end
```

```matlab
function xd_dot = ref_model(xd,psi_ref)
w_ref      = 0.03;
zeta     = 1;


a1 = w_ref +  2*zeta*w_ref;
a2 = 2*zeta*w_ref^2 + w_ref^2;
a3 = w_ref^3;
b3 = a3;
A = [ 0 1 0; 0 0 1; -a3 -a2 -a1];
B = [0 0 b3]';

xd_dot = A*xd + B*psi_ref;
end
```

The performance of the PID controller with regards to the environmental disturbances is shown in Figure 3. Since we have made a heading autopilot, the yaw angle is controlled to 0, handling both the ocean current and the wind disturbances. This is the expected behaviour of the PID controller. The North-East-plot shows the distance traveled, however. In this plot it is possible to see the ship moving constantly to the North-East with the $\psi = 0$, $\beta \neq 0$ and $\beta_c \neq 0$. It is beacuse of β and β_c we get behaviour of Figure 2a. If we want the ship to follow a desired course, a course autopilot needs to be implemented.

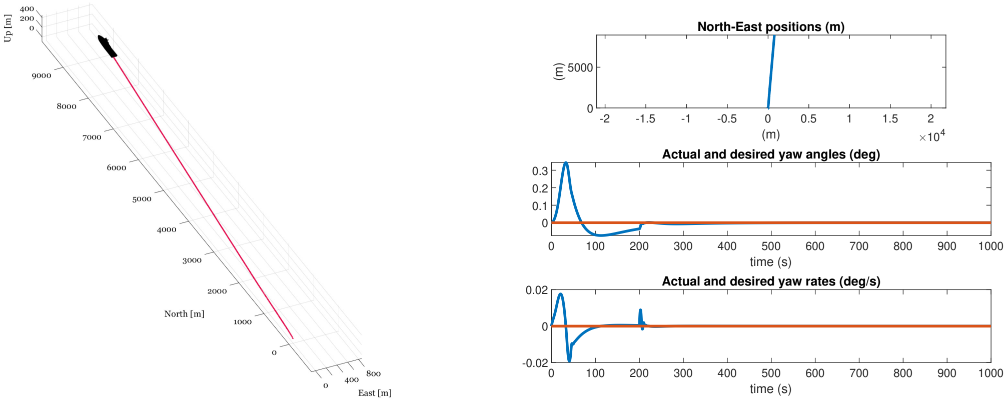
To summarize, the PID controller for the heading autopilot does indeed compensate for environmental disturbances.

e)

Based on the simulation results from Figure 3c, it does not seem like integrator windup is a problem for the heading angle. The actual and the commanded yaw angles are close to identical when the controller changes the value of ψ_c.

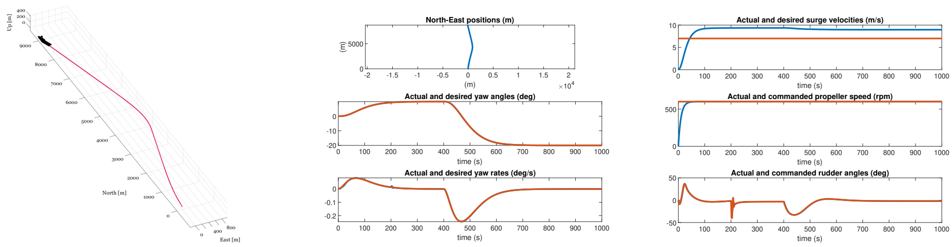
```matlab
psi_ref = [10 * pi/180; - 20 * pi/180];  % desired yaw angle (rad)
```

```matlab
if t > 200
    xd_dot = ref_model(xd,psi_ref(2));
else
    xd_dot = ref_model(xd,psi_ref(1));
    end
```



(a) The course χ of the Boat (b) The heading ψ of the Boat

Figure 2: The performance of the PID heading controller



(a) The course χ of the Boat (b) The heading ψ of the Boat (c) The speed of the Boat

Figure 3: Controller performance for a 10°heading setpoint followed by a -20°heading setpoint

References

[1] T. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011.