TTK4190 Guidance and Control of Vehicles

Assignment 3, Part 2

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

1 Environmental disturbances

1.1 Problem a)

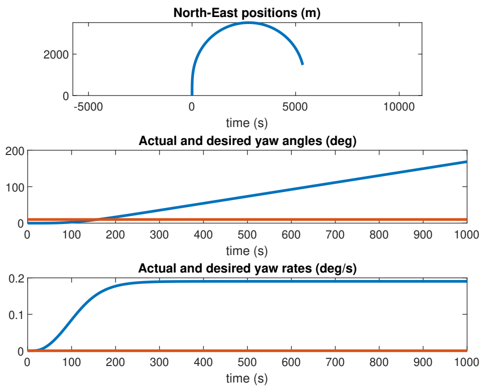
Added ocean currents $\nu_r = \nu - \nu_c$ through equation (10.149) in Fossen

$$\begin{aligned} u_c &= V_c cos(\beta_{V_c} - \psi) \\ v_c &= V_c sin(\beta_{V_c} - \psi) \end{aligned} \tag{1}$$

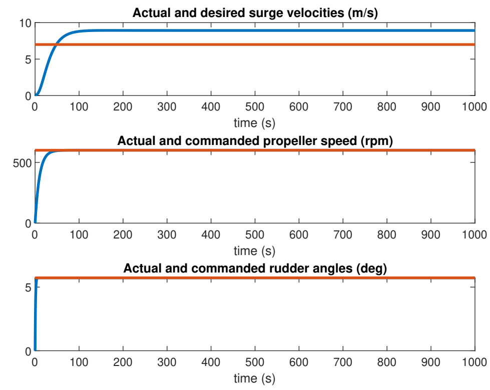
Where V_c is the ocean current, β_{V_c} is the direction of the current and ψ is the heading of the vessel. This is a transform from NED to body.

1.2 Problem b)

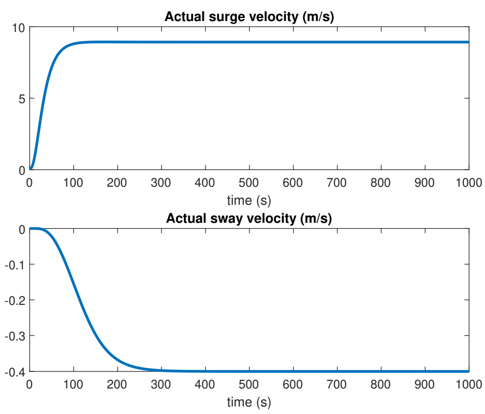
Simulating without ocean current in figure 1



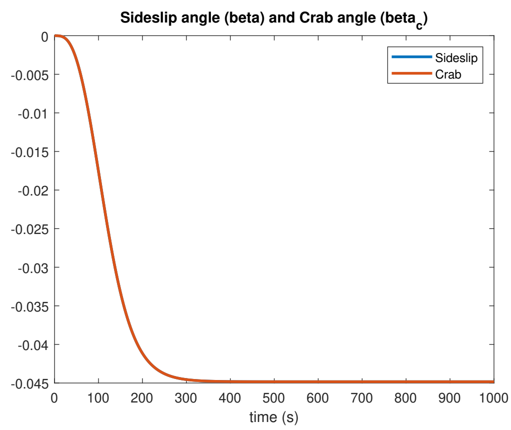
(a)



(b)



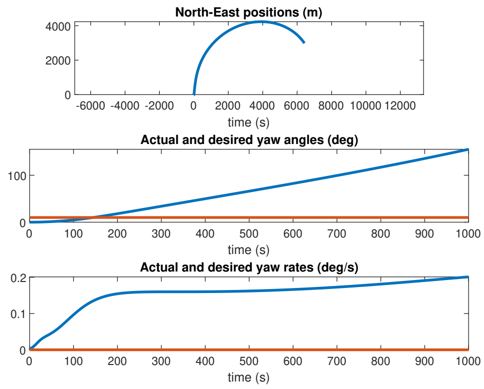
(c)



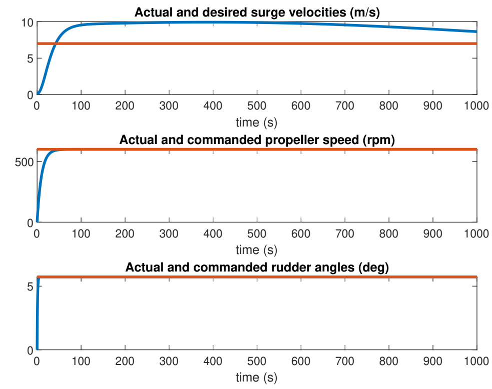
(d)

Figure 1

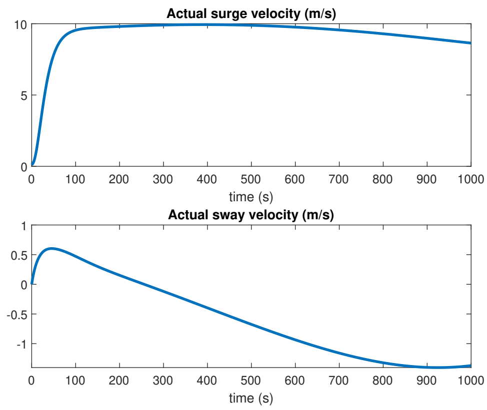
Simulating with ocean currents in figure 2.



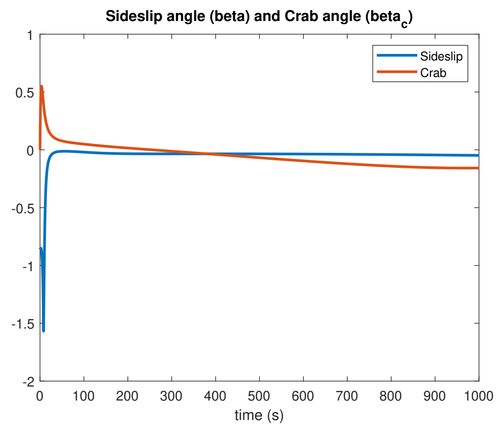
(a)



(b)

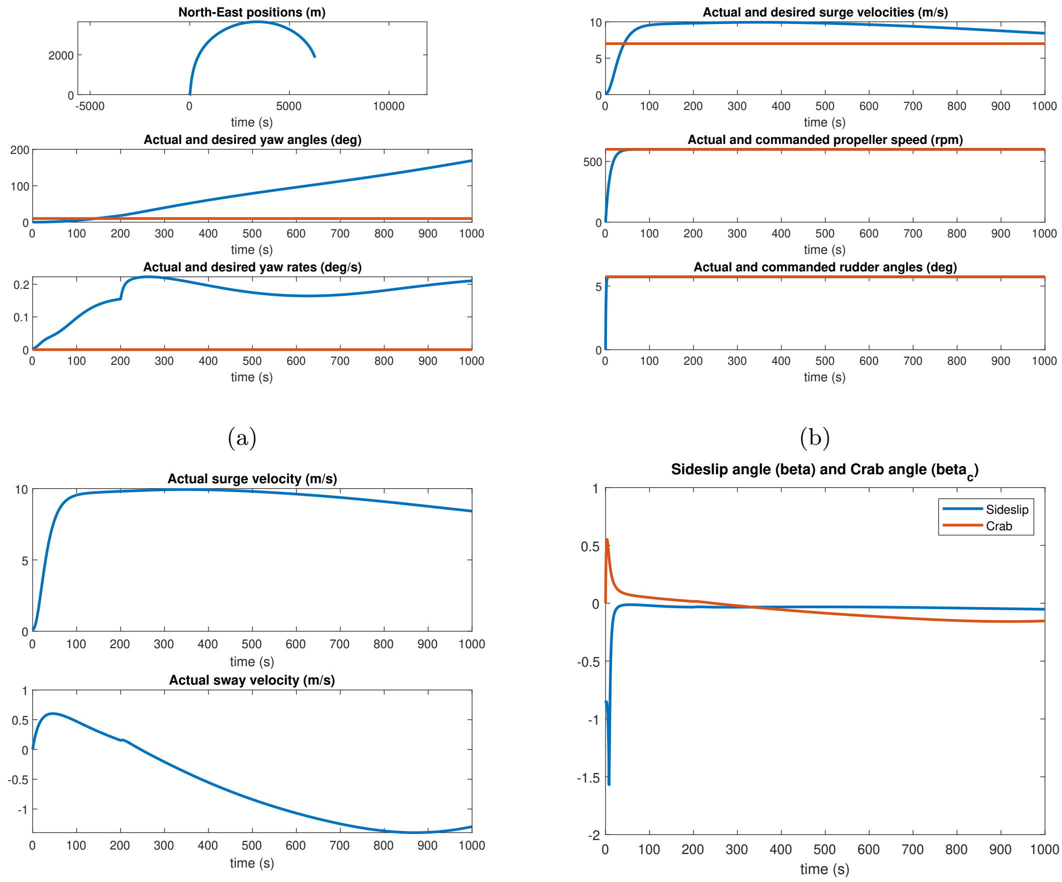


(c)



(d)

Figure 2



(a) (b) (c) (d)

Figure 3

1.3 Problem c)

Added wind in figure 3

2 Heading autopilot

2.1 Problem a)

When linearizing our system we already have C_{RB}, C_A and D from the previous part. Assuming $v = r = 0$ and unknown water currents we get

$$\begin{aligned} C_{RB}^* &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & mU_d \\ 0 & 0 & mx_g U_d \end{bmatrix} \\ C_A^* &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -X_{\dot{u}} U_d \\ 0 & (X_{\dot{u}} - Y_{\dot{v}})U_d & -Y_{\dot{r}} U_d \end{bmatrix} \end{aligned} \tag{2}$$

From equation (6.129) and (6.130) in Fossen chapter 6.

2.2 Problem b)

Linearizing for sway-yaw yields matrices

$$
\begin{aligned}
C_{RB}^* &= \begin{bmatrix} 0 & mU_d \\ 0 & mx_gU_d \end{bmatrix} \\
C_A^* &= \begin{bmatrix} 0 & -X_{\dot{u}}U_d \\ (X_{\dot{u}} - Y_{\dot{v}})U_d & -Y_{\dot{r}}U_d \end{bmatrix} \\
D &= -\begin{bmatrix} Y_v & Y_r \\ N_v & N_r \end{bmatrix}
\end{aligned} \tag{3}
$$

With a state space model where $N = C_{RB}^* + C_A^* + D$

$$
M\dot{\nu} + N\nu = b\delta \tag{4}
$$

Using the MATLAB function $ss2tf$ with input $A = -M^{-1}N$, $B = M^{-1}b$, $C = [01]$ and $D = 0$ gives the following transfer function from δ to r

$$
H(s) = \frac{r}{\delta}(s) = \frac{0.8638s + 0.0615}{s^2 + 0.1506s + 0.0008} \cdot 10^{-4} \tag{5}
$$

2.3 Problem c

To find the first-order Nomoto model, we first found the second-order Nomoto model and then approximated it. By factorising the roots, we obtained values for T_1 and T_2, while for T_3 a simple scaling was taken. This is easier seen as:

$$
\frac{r}{\delta}(s) = \frac{K'(s + n_1)}{(s + \lambda_1)(s + \lambda_2)} = \frac{K'n_1}{\lambda_1\lambda_2} \cdot \frac{(\frac{1}{n_1}s + 1)}{(\frac{1}{\lambda_1}s + 1)(\frac{1}{\lambda_2}s + 1)} = \frac{K(T_3s + 1)}{(T_1s + 1)(T_2s + 1)} \tag{6}
$$

The second order model was found to be:

$$
\frac{r}{\delta}(s) = 7.4976 \cdot 10^{-3} \cdot \frac{14.0455s + 1}{(176.7041s + 1)(6.8992s + 1)} \tag{7}
$$

Where we read our parameter values as: $K = 7.4976 \cdot 10^{-3}$, $T_1 = 176.7041$, $T_2 = 6.8992$ and $T_3 = 14.0455$. Approximating the first order time constant gives $T = T_1 + T_2 - T_3 = 169.5578$. K remains the same.

$$
\frac{r}{\delta}(s) = \frac{K}{Ts + 1} = \frac{0.0615 \cdot 10^{-4}}{169.5578s + 1} \tag{8}
$$

2.4 Problem d)

Based on Example 15.7, we could reverse engineer the parameter values K_p, K_d and K_i based on our linearised first-order Nomoto model:

$$
K_p = w_n^2 \frac{T}{K}, \qquad K_d = \frac{2\zeta w_n T - 1}{K}, \qquad K_i = w_n^3 \frac{T}{10K} \tag{9}
$$

where the values for T and K were found in c). With this, we defined the SISO linear PID control (with Nomoto, $k = 0$):

$$
\tau = -\left(K_p\tilde{x} + K_d\dot{x} + K_i \int_0^t \tilde{x}(\tau)d\tau \right) \tag{10}
$$

with $\tilde{x} = x - x_d$.

Next, we used a 3rd order reference model given by equation 12.12, where we defined $\Omega = w_{ref} = 0.006$ and $\Delta = \zeta = 1$.

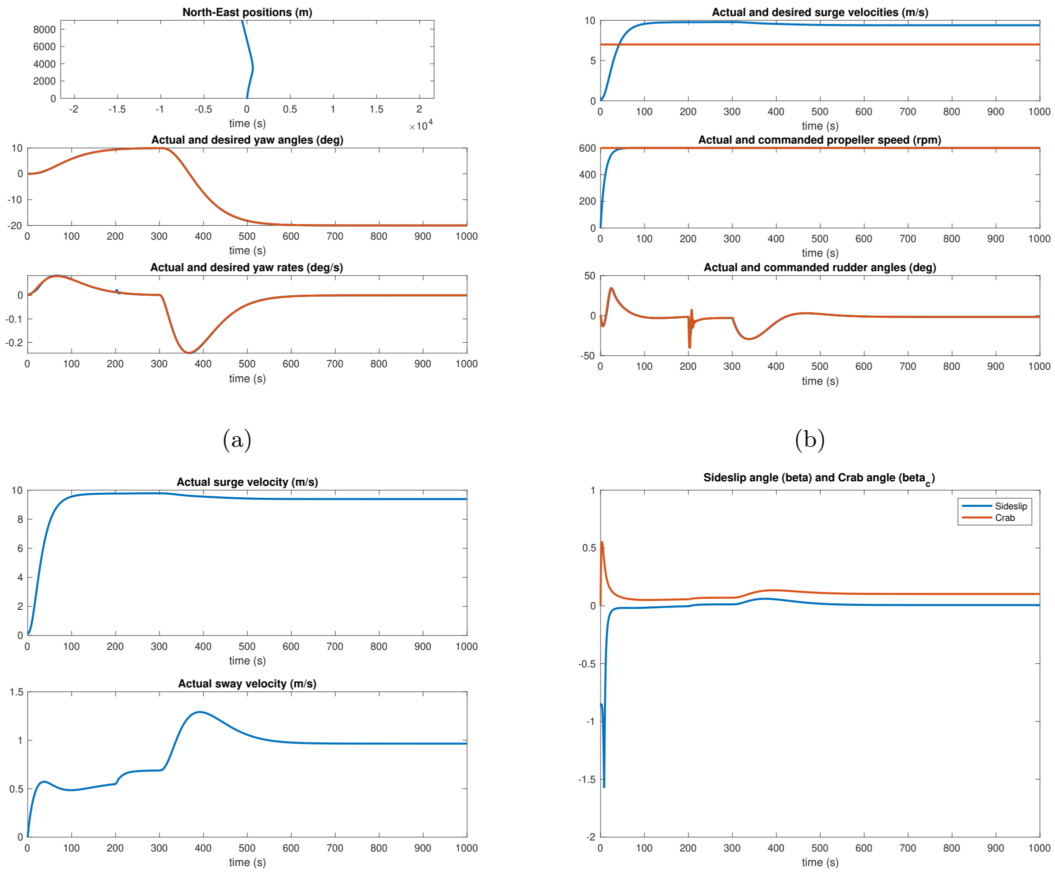
$$
\dot{x}_d = A_d x_d + B_d r^n \tag{11}
$$

$$A_d = \begin{bmatrix} 0_{nxn} & I_n & 0_{nxn} \\ 0_{nxn} & 0_{nxn} & I_n \\ -\Omega^3 & -(2\Delta + I_n)\Omega^2 & -(2\Delta + I_n)\Omega \end{bmatrix}, \qquad B_d = \begin{bmatrix} 0_{nxn} \\ 0_{nxn} \\ -\Omega^3 \end{bmatrix} \tag{12}$$

Setting the input into this reference model produces a nice tracking reference x_d for the control law in Eq. (10).

2.5 Problem 2e)

Boat dynamics for a 10° to −20° heading setpoint change in figure 4.



(a) (b) (c) (d)

Figure 4: Controller performance for a 10° to −20° maneuver.

From Figure 4, we see that integrator wind up is not a problem in our simulations, as the ship is quickly able to adjust to the new setpoint.

A Matlab code

A.1 Problem 4 - project.m

```
% 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 = 7;                % 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);
```

```matlab
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;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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]';

    % 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)

% 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

% propeller dynamics
Im = 100000; Tm = 10; Km = 0.6;         % propulsion parameters
n_c = 10;                               % propeller speed (rps)
n_dot = (1/10) * (n_c - n);             % 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);

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