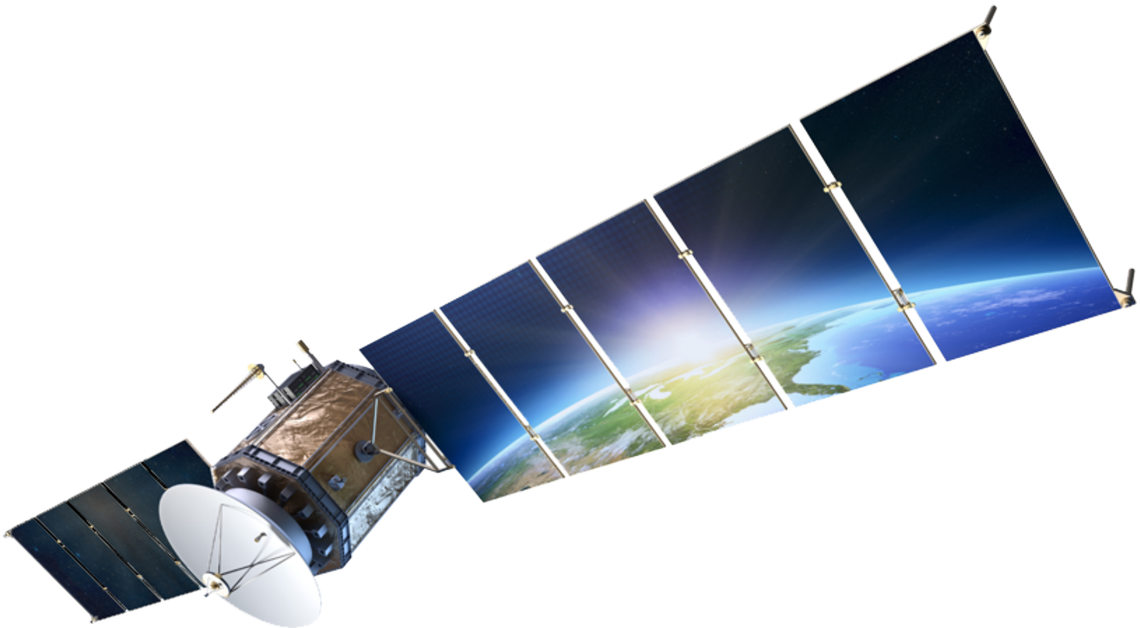
TTK4190 Guidance and Control of Vehicles

Assignment 1

Written Fall 2018 By
Anders Haver Vagle,
Ada Skarsholt Larsen,
Sondre Aleksander Bergum,
Martin Madsen



NTNU

Department of Engineering Cybernetics

Problem 1 - Attitude Control of Satellite

Problem 1.1

The equilibrium point $\boldsymbol{x}_0$ of the closed-loop $\boldsymbol{x} = [\boldsymbol{\epsilon}^T, \boldsymbol{\omega}^T]^T$ can be found by looking at $\dot{\boldsymbol{q}} = \mathbf{0}$ to find $\boldsymbol{\omega}$.

$$\begin{aligned}
\dot{\boldsymbol{q}} &= \boldsymbol{T}_q(\boldsymbol{q})\boldsymbol{\omega} && (1)\\
&= \frac{1}{2}\begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & \epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix}\boldsymbol{\omega} && (2)\\
&= \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\boldsymbol{\omega} = \mathbf{0} && (3)\\
& && (4)
\end{aligned}$$

This gives

$$\boldsymbol{\omega} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{5}$$

Therefore, when $\boldsymbol{\epsilon} = [0, 0, 0]^T$ we have

$$\boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{\epsilon}^T \\ \boldsymbol{\omega}^T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

Linearizing about $\boldsymbol{x} = \boldsymbol{x}_0$ gives

$$\begin{aligned}
\boldsymbol{I}_{CG}\dot{\boldsymbol{\omega}} &= \boldsymbol{S}(\boldsymbol{I}_{CG}\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{\tau} && (7)\\
\dot{\boldsymbol{\omega}} &= \boldsymbol{I}_{CG}^{-1}\boldsymbol{S}(\boldsymbol{I}_{CG}\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{I}_{CG}^{-1}\boldsymbol{\tau} && (8)\\
& && (9)
\end{aligned}$$

and

$$\begin{aligned}
\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{\epsilon}}^T \\ \dot{\boldsymbol{\omega}}^T \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\omega}\boldsymbol{T} \\ \boldsymbol{I}_{CG}^{-1}\boldsymbol{S}(\boldsymbol{I}_{CG}\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{I}_{CG}^{-1}\boldsymbol{\tau} \end{bmatrix} = \mathbf{h}(\mathbf{x}, \mathbf{u}) && (10)\\
&& (11)
\end{aligned}$$

Where we have

$$\begin{aligned}
\boldsymbol{I}_{CG}^{-1}\boldsymbol{S}(\boldsymbol{I}_{CG}\boldsymbol{\omega})\boldsymbol{\omega} &= \frac{1}{mr^2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & -mr^2\omega & mr^2\omega \\ mr^2\omega & 0 & -mr^2\omega \\ -mr^2\omega & mr^2\omega & 0 \end{bmatrix} && (12)\\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} && (13)
\end{aligned}$$

We then linearize:

$$\mathbf{A} = \frac{\delta \mathbf{h}(\mathbf{x}, \mathbf{u})}{\delta \mathbf{x}}|_{\mathbf{x}^*, \mathbf{u}^*}, \quad \mathbf{B} = \frac{\delta \mathbf{h}(\mathbf{x}, \mathbf{u})}{\delta \mathbf{u}}|_{\mathbf{x}^*, \mathbf{u}^*}, \tag{14}$$

This gives A matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{15}$$

And B matrix given by the second part of $\dot{\boldsymbol{\omega}}$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{mr^2} & 0 & 0 \\ 0 & \frac{1}{mr^2} & 0 \\ 0 & 0 & \frac{1}{mr^2} \end{bmatrix} \tag{16}$$

Problem 1.2

With our state space expression from 1.2 and $\boldsymbol{\tau} = -\boldsymbol{K}_d \boldsymbol{\omega} - k_p \boldsymbol{\epsilon}$ we have

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(-\boldsymbol{K}_d \boldsymbol{\omega} - k_p \boldsymbol{\epsilon}) \tag{17}$$

This gives

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ -\frac{k_p}{mr^2} & 0 & 0 & -\frac{k_d}{mr^2} & 0 & 0 \\ 0 & -\frac{k_p}{mr^2} & 0 & 0 & -\frac{k_d}{mr^2} & 0 \\ 0 & 0 & -\frac{k_p}{mr^2} & 0 & 0 & -\frac{k_d}{mr^2} \end{bmatrix} \tag{18}$$

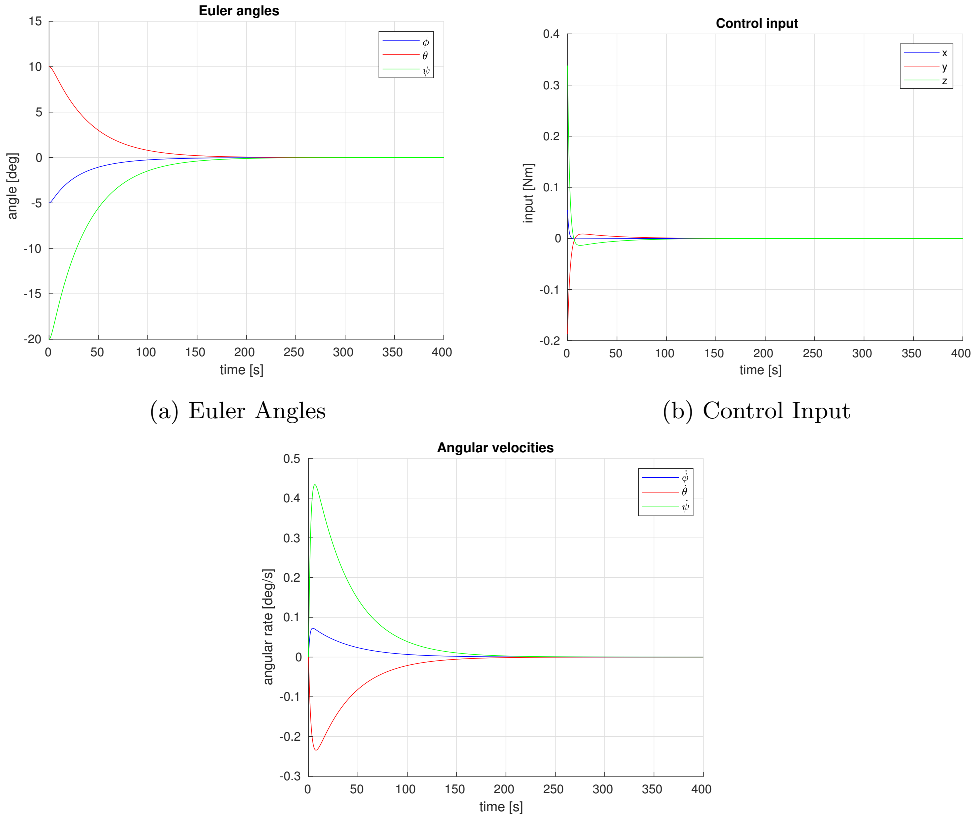
Inserted into MATLAB we calculate the eigenvalues to be complex, all in the left half plane. The eigenvalues in the linearized system are therefore stable.

In this particular application we would like to have our poles result in a critically damped response, $\zeta = 1$, given by poles with negative real part and no complex part[1]. However, whether the poles are real or complex does not matter too much because the system is linearized meaning that the system response will not be accurate to the real system response anyway. Any inaccuracies in modeling will alter the damping ratio of the actual system.

Problem 1.3

Simulating with the given initial conditions gives the plots in Figure 1. The system response seems reasonable, all angles and velocities reach their desired reference of zero, and the control input approaches zero. This seems reasonable as the system is not affected by any external disturbances. We introduce a reference by changing the control law to equation 19, this allows us to follow nonzero references.

$$\boldsymbol{\tau} = \boldsymbol{K}_d(\boldsymbol{\omega}_r - \boldsymbol{\omega}) + k_p(\boldsymbol{\epsilon}_r - \boldsymbol{\epsilon}) \tag{19}$$



(a) Euler Angles

(b) Control Input

(c) Angular Velocities

Figure 1: Simulation of attitude dynamics

Problem 1.4

The quaternion error can be written as

$$\tilde{\mathbf{q}} := \begin{bmatrix} \tilde{\eta} \\ \tilde{\epsilon} \end{bmatrix} = \bar{\mathbf{q}}_d \otimes \mathbf{q} \tag{20}$$

Because we have

$$\bar{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{\eta} \\ -\boldsymbol{\epsilon}^T \end{bmatrix} \tag{21}$$

and

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \epsilon^T \end{bmatrix} \tag{22}$$

We get

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} \eta_d \eta + \epsilon_d^T \epsilon \\ \eta_d \epsilon - \eta \epsilon_d - S(\epsilon_d)\epsilon \end{bmatrix} \tag{23}$$

We find $S(\epsilon_d)\epsilon$ to be

$$S(\epsilon_d)\epsilon = \begin{bmatrix} 0 & -\epsilon_{3,d} & \epsilon_{2,d} \\ \epsilon_{3,d} & 0 & -\epsilon_{1,d} \\ -\epsilon_{2,d} & \epsilon_{1,d} & 0 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} -\epsilon_{3,d}\epsilon_2 + \epsilon_{2,d}\epsilon_3 \\ \epsilon_{3,d}\epsilon_1 - \epsilon_{1,d}\epsilon_3 \\ -\epsilon_{2,d}\epsilon_1 + \epsilon_{1,d}\epsilon_2 \end{bmatrix} \tag{24}$$

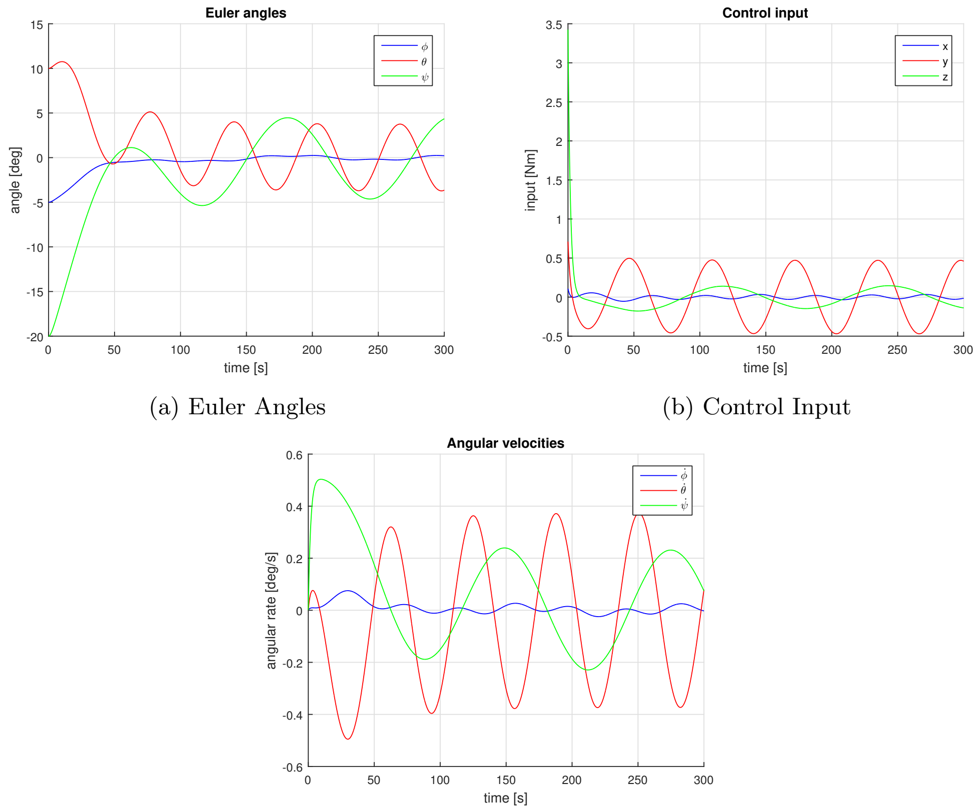
Which when $q = q_d$ yields

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} \eta^2 + \epsilon^2 \\ -\epsilon_3\epsilon_2 + \epsilon_2\epsilon_3 \\ \epsilon_3\epsilon_1 - \epsilon_1\epsilon_3 \\ -\epsilon_2\epsilon_1 + \epsilon_1\epsilon_2 \end{bmatrix} = \begin{bmatrix} 1 - \epsilon^2 + \epsilon^2 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{25}$$

Problem 1.5

Simulating the attitude dynamics with the given parameters we get the plots in Figure 2. We can see that ϕ goes to approximately zero and θ and ψ is approximating cosine and sine but a large reference tracking error.

Figure 3 shows the tracking error. We penalize any nonzero ω in our control law, while trying to follow a nonzero reference. This leads to a very damped system response and the large tracking error that we observe.



(a) Euler Angles

(b) Control Input

(c) Angular Velocities

Figure 2: Simulation of attitude dynamics

Problem 1.6

The control law in this problem can be written as

$$\boldsymbol{\tau} = -\mathbf{K}_d\tilde{\boldsymbol{\omega}} - k_p\tilde{\boldsymbol{\epsilon}} \tag{26}$$

and the desired angular velocity as

$$\boldsymbol{\omega}_d = \mathbf{T}^{-1}_{\Theta_d}(\Theta_d)\dot{\Theta}_d \tag{27}$$

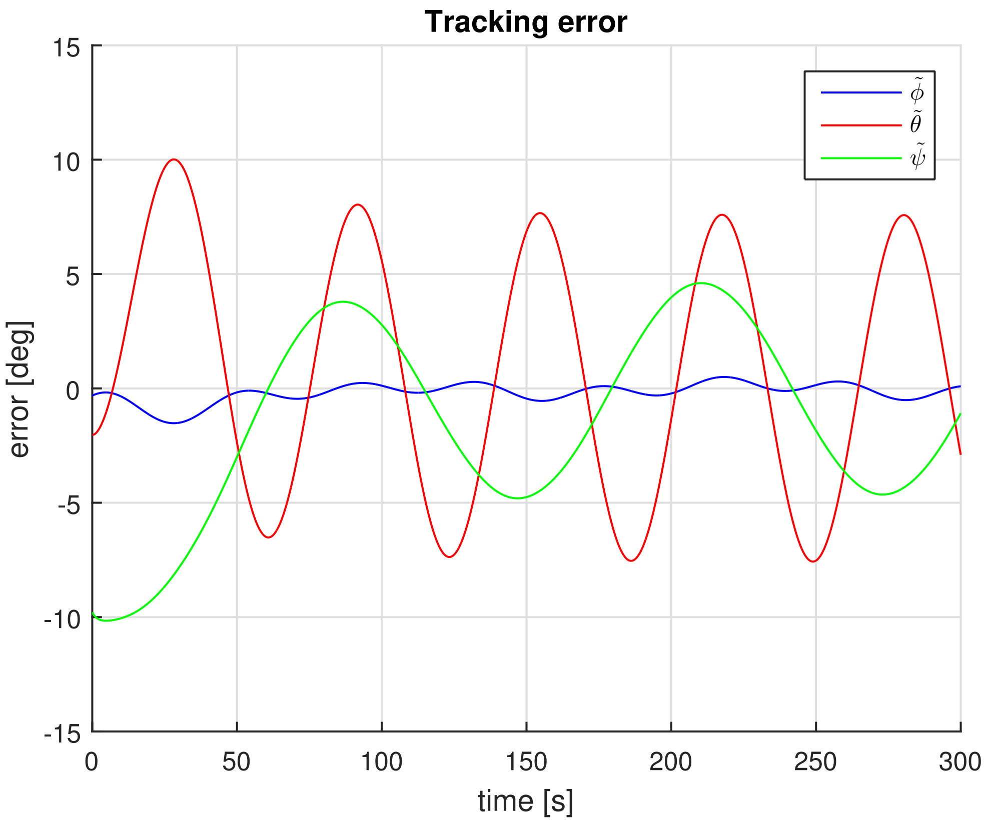


Figure 3: Tracking Error

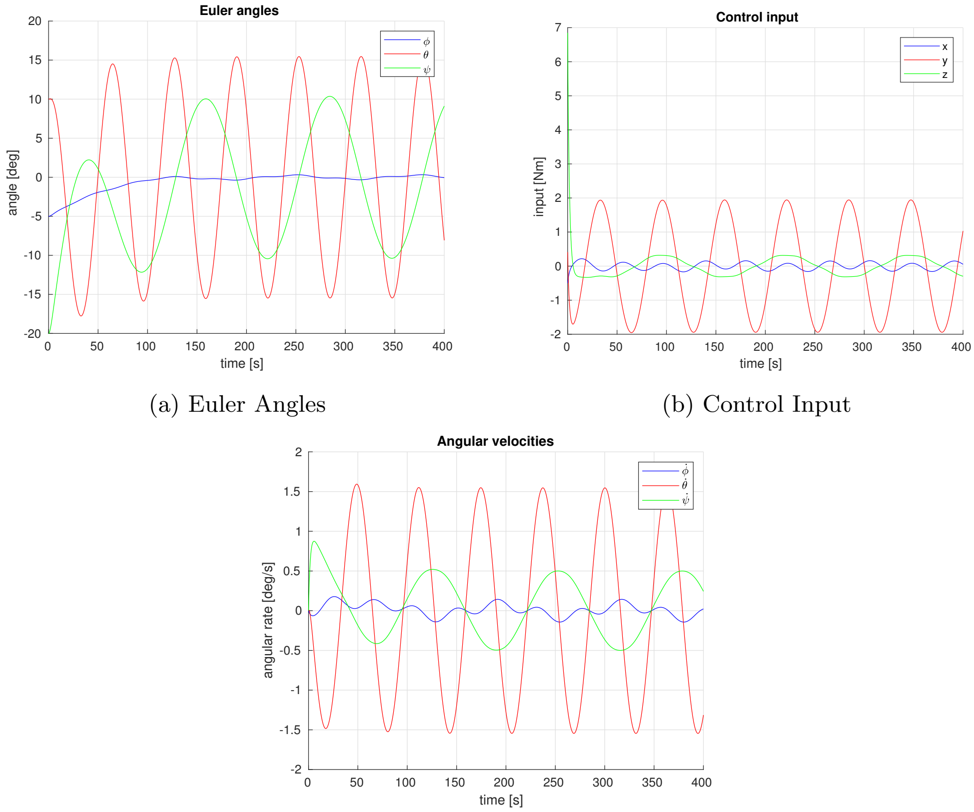
We have:

$$\phi(t) = 0 \tag{28}$$
$$\theta(t) = 15\cos(0.1t) \tag{29}$$
$$\psi(t) = 10\sin(0.05t) \tag{30}$$
$$\dot{\phi}(t) = 0 \tag{31}$$
$$\dot{\theta}(t) = -1.5\sin(0.1t) \tag{32}$$
$$\dot{\psi}(t) = 0.5\cos(0.05t) \tag{33}$$

Which gives

$$\boldsymbol{\omega}_d = \boldsymbol{T}_{\Theta_d}^{-1}(\boldsymbol{\Theta}_d) \tag{34}$$
$$= \begin{bmatrix} 1 & 0 & -\sin(\theta_d) \\ 0 & \cos(\phi_d) & \cos(\theta_d)\sin(\phi_d) \\ 0 & -\sin(\phi_d) & \cos(\theta_d)\cos(\phi_d) \end{bmatrix} \begin{bmatrix} 0 \\ -1.5\sin(0.1t) \\ 0.5\cos(0.05t) \end{bmatrix} \tag{35}$$
$$\Rightarrow = \begin{bmatrix} -0.5\sin(\theta_d)\cos(0.05t) \\ -1.5\sin(0.1t)\cos(\phi_d) + 0.5\cos(0.05t)\cos(\theta_d)\sin(\phi_d) \\ 1.5\sin(\phi_d)\sin(0.1t) + 0.5\cos(0.05t)\cos(\theta_d)\cos(\phi_d) \end{bmatrix} \tag{36}$$

By looking at the simulation plots in Figure 4 we can see that θ and ψ now actually follow the reference and ϕ is a little bit better than the previous task. This can be confirmed by looking at



(a) Euler Angles

(b) Control Input

(c) Angular Velocities

Figure 4: Simulation of attitude dynamics

the tracking error, figure 5, which is significantly better than the previous task, and is caused by the change from ω to $\tilde{\omega}$ in the control law.
A way to further improve this result can be to add an integral term or to find a control law with a candidate function which gives a negative definite derivation such that the equilibrium is globally exponentially stable[2].

Problem 1.7

The Lyapunov function can be written as

$$V = \frac{1}{2}\tilde{\boldsymbol{\omega}}^{\top}\mathbf{I}_{CG}\tilde{\boldsymbol{\omega}} + 2k_p(1-\tilde{\eta}) \tag{37}$$

We want to prove that V is positive:

$$\begin{aligned} \mathbf{I_{CG}} &> 0 && (38)\\ \Rightarrow \tilde{\omega}\mathbf{I_{CG}}\tilde{\omega} &> 0 && (39)\\ k_p &> 0 && (40)\\ \tilde{\omega} &= \omega - \omega_{\mathbf{d}} = \omega && (41)\\ & && (42) \end{aligned}$$

$\tilde{\eta}$ is a quaternion and thus $\tilde{\eta} \leq 1 \Rightarrow (1-\tilde{\eta}) \geq 0$. As such, V is positive.

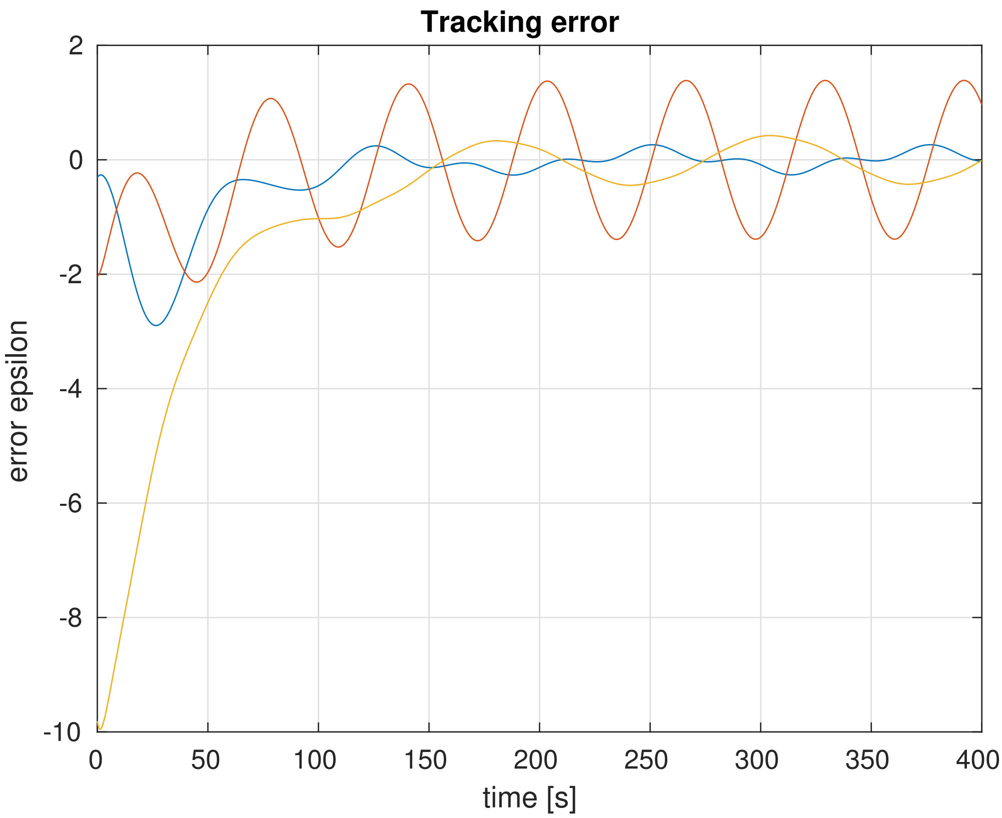


Figure 5: Tracking Error

We then also see that

$$\lim_{\omega \to \infty} V = \infty \tag{43}$$

We then want to show that

$$\dot{V} = -k_d \omega^T \omega \tag{44}$$

$$V = \frac{1}{2}\tilde{\boldsymbol{\omega}}^\top \mathbf{I}_{CG}\tilde{\boldsymbol{\omega}} + 2k_p(1-\tilde{\eta}) \tag{45}$$

$$I_{CG}\dot{\omega} - S(I_{CG}\omega)\omega = \tau \tag{46}$$

$$\Rightarrow \dot{\omega} = \frac{\tau + \omega S(I_{CG}\omega)}{I_{CG}} \tag{47}$$

$$\dot{\tilde{\eta}} = -\frac{1}{2}\tilde{\epsilon}^T\tilde{\omega} \tag{48}$$

$$\tau = -K_d\omega - k_p\tilde{\epsilon} \tag{49}$$

$$\tag{50}$$

$$S(I_{CG}\omega)\omega = \begin{bmatrix} 0 & -mr^2\omega_3 & mr^2\omega_2 \\ mr^2\omega_3 & 0 & -mr^2\omega_1 \\ -mr^2\omega_2 & mr^2\omega_1 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{51}$$

$$= \begin{bmatrix} -mr^2\omega_3\omega_2 + mr^2\omega_2\omega_3 \\ mr^2\omega_3\omega_1 - mr^2\omega_1\omega_3 \\ -mr^2\omega_2\omega_1 + mr^2\omega_1\omega_2 \end{bmatrix} \tag{52}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{53}$$

Which with the use of (51) gives

$$\dot{V} = \omega^T I_{CG}\dot{\omega} - 2k_p\dot{\tilde{\eta}} \tag{54}$$

$$= \omega^T(\tau + S(I_{CG}\omega)\omega) + k_p\tilde{\epsilon}^T\tilde{\omega} \tag{55}$$

$$= \omega^T(-K_d\omega - k_p\tilde{\epsilon} + S(I_{CG}\omega)\omega) + k_p\tilde{\epsilon}^T\tilde{\omega} \tag{56}$$

$$= \omega^T(-K_d\omega) + \omega^T S(I_{CG}\omega)\omega \tag{57}$$

$$\Rightarrow \dot{V} = -k_d\omega^T\omega \tag{58}$$

All criteria of Theorem 4.2 in Khalil[2] are satisfied, meaning the equilibrium is globally asymptotically stable.

Problem 2 - Straight-line path following in the horizontal plane

Problem 2.1

Using the kinematics in chapter 2 in book by Fossen [3] we have

$$
\begin{aligned}
\dot{\eta} &= R(\psi)v && (59)\\
\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\end{bmatrix} &= \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u\\ v\\ r\end{bmatrix} && (60)\\
\rightarrow \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} &= \begin{bmatrix}u\cos(\psi) - v\sin(\psi)\\ u\sin(\psi) + v\cos(\psi)\end{bmatrix} && (61)
\end{aligned}
$$

Here we assume that θ and ϕ are small enough so that $\cos(\theta) \approx 1$, $\sin(\theta) \approx 0$.

We have that:

$$
\begin{aligned}
\cos(\phi + \beta) &= \cos(\phi)\cos(\beta) - \sin(\phi)\sin(\beta) && (62)\\
\Rightarrow U\cos(\phi + \beta) &= U(\cos(\phi)\cos(\beta) - \sin(\phi)\sin(\beta) && (63)\\
& && (64)
\end{aligned}
$$

$$
\dot{x} = u\cos(\psi) - v\sin(\psi) \tag{65}
$$

To match the above expression, we can write:

$$
\begin{aligned}
u\cos(\phi) &= U\cos(\phi)\cos(\beta) && (66)\\
v\sin(\phi) &= U\sin(\phi)\sin(\beta) && (67)\\
\Rightarrow u &= U\cos(\beta), \quad v = U\sin(\beta) && (68)
\end{aligned}
$$

Which gives that:

$$
\begin{aligned}
\dot{x} &= u\cos(\psi) - v\sin(\psi) && (69)\\
&= U\cos(\psi)\cos(\beta) - U\sin(\psi)\sin(\beta) && (70)\\
&= U(\cos(\psi)\cos(\beta) - \sin(\psi)\sin(\beta)) && (71)\\
&= U\cdot\cos(\psi + \beta) && (72)\\
&= U\cos(\chi) && (73)
\end{aligned}
$$

Likewise,

$$
\begin{aligned}
\dot{y} &= v\cos(\psi) + u\sin(\psi) && (74)\\
&= U\sin(\psi)\cos(\beta) + U\cos(\psi)\sin(\beta) && (75)\\
&= U(\sin(\psi)\cos(\beta) + \cos(\psi)\sin(\beta)) && (76)\\
&= U\cdot\sin(\psi + \beta) && (77)\\
&= U\sin(\chi) && (78)
\end{aligned}
$$

Problem 2.2

If B is small, $\psi + B \approx \psi$. With ψ also being small, $\sin\psi \approx \psi$ and $\cos\psi \approx 1$. Y becomes the cross-track error when the path we are following is directly north or south from $y = 0$.

Problem 2.3

We have

$$\begin{aligned} T\dot{r} + r &= K\delta + b && (79)\\ \dot{\psi} &= r && (80)\\ & && (81) \end{aligned}$$

Following the Laplace transform we get

$$\begin{aligned} Ts \cdot r + r &= K\delta(s) + b(s) && (82)\\ \rightarrow r(1 + Ts) &= K\delta(s) + b(s) && (83)\\ \rightarrow r &= \frac{K\delta(s) + b(s)}{1 + Ts} && (84)\\ s\psi(s) &= r && (85)\\ \rightarrow \psi(s) &= \frac{r}{s} && (86) \end{aligned}$$

The cross-track error y can then be stated as

$$\begin{aligned} y(s) &= \frac{r}{s^2}U(s) && (87)\\ &= \frac{1}{s^2}U(s)\frac{K\delta(s) + b(s)}{1 + Ts} && (88)\\ &= \frac{U(s)K}{s^2(1 + Ts)}\delta(s) + \frac{U(s)}{s^2(1 + Ts)}b(s) && (89) \end{aligned}$$

The transfer functions $h_1(s)$ and $h_2(s)$ are:

$$\begin{aligned} h_1(s) &= K\frac{U(s)}{s^2(1 + Ts)} && (90)\\ h_2(s) &= \frac{U(s)}{s^2(1 + Ts)} && (91) \end{aligned}$$

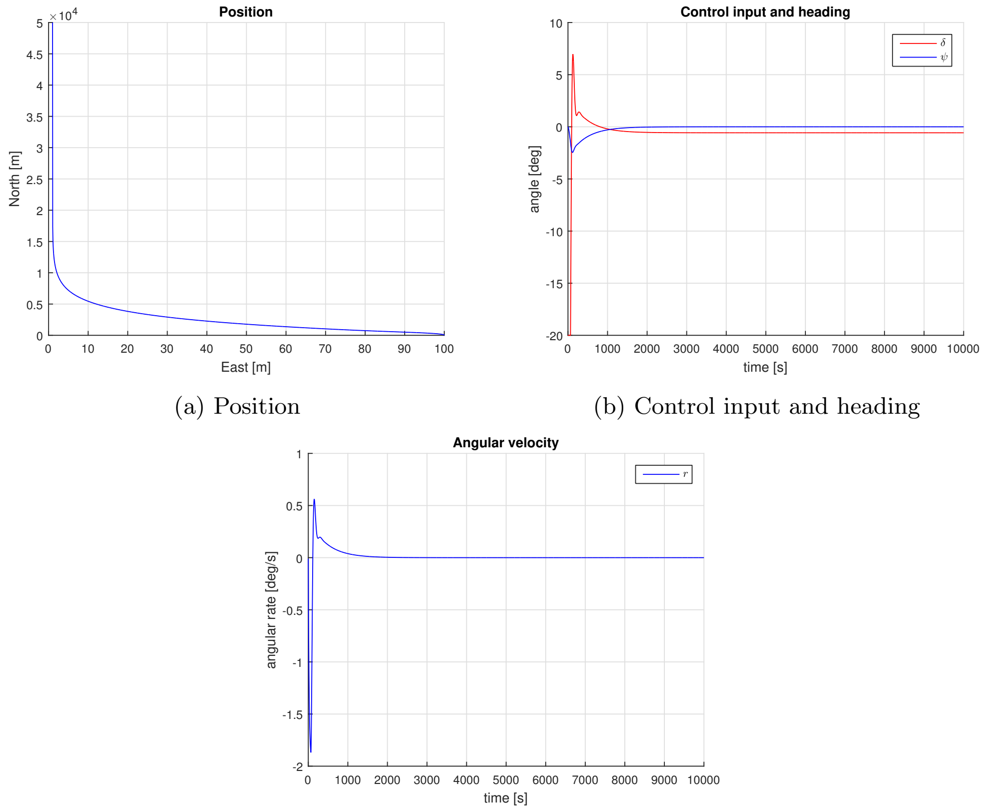
We need the I term in the controller because it eliminates the accumulating error and we need the D term because it counteracts the overshoot caused by I and P, and damps some oscillations in the system therefore avoiding damaging the actuators of the system.

Problem 2.4

The PID-controller is given by

$$\delta = -k_p y - k_d \dot{y} - k_i \int y \tag{92}$$

$$x(0) = 0 \ m \tag{93}$$
$$y(0) = 100 \ m \tag{94}$$
$$\psi(0) = 0^\circ \tag{95}$$
$$r(0) = 0 \ deg/s \tag{96}$$
$$U(0) = 5 \ m/s \tag{97}$$
$$\implies \quad u = 5 \ m/s \tag{98}$$
$$v = 0 \ m/s \tag{99}$$



(a) Position

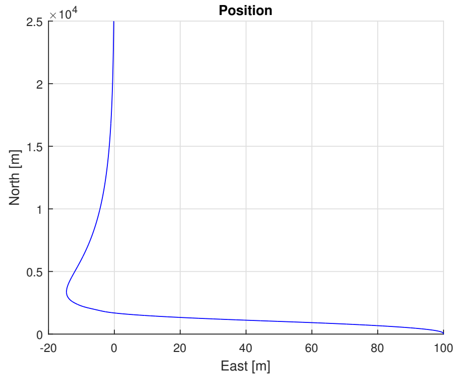
(b) Control input and heading

(c) Angular Velocity

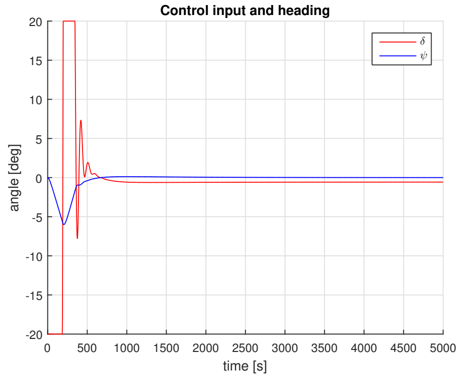
Figure 6: PD controller

With a PD controller, we get the response as shown in Figure 6. We see that we get a slight stationary error due to the bias on the rudder. Introducing an integral effect in the controller to combat this, we get the results shown in Figure 7.

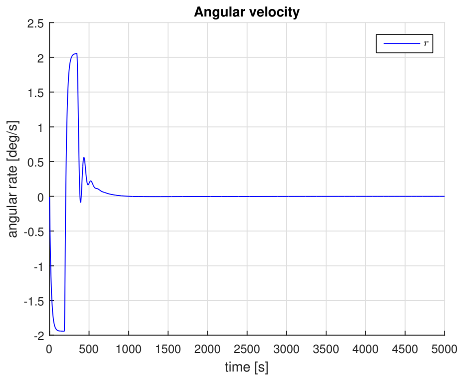
We now see that the stationary error is gone, but we now have a more complex controller to tune. We were unable to remove the overshoot while keeping the system stable. How critical this overshoot is would depend entirely on the specific use case of the system. The same is true for how aggressive the controller should be; the most efficient path to a point far ahead up north would be a straight line from the initial point. However, if it is more important to follow the path at



(a) Position



(b) Control input and heading



(c) Angular Velocity

Figure 7: PID controller

$y = 0$ at all times than to reach the destination as fast as possible, the controller should be more aggressively tuned.

Appendix

Problem1.m

```
% M-script for numerical integration of the attitude dynamics of a rigid
% body represented by unit quaternions. The MSS m-files must be on your
% Matlab path in order to run the script.
%
% System:                        .
%                                q = T(q)w
%                                .
%                              I w - S(Iw)w = tau
% Control law:
%                              tau = constant
%
% Definitions:
%                              I = inertia matrix (3x3)
%                              S(w) = skew-symmetric matrix (3x3)
%                              T(q) = transformation matrix (4x3)
%                              tau = control input (3x1)
%                              w = angular velocity vector (3x1)
%                              q = unit quaternion vector (4x1)
%
% Author:                     2018-08-15 Thor I. Fossen and Hkon H. Helgesen

%% USER INPUTS
h = 0.1;                       % sample time (s)
N  = 4000;                      % number of samples. Should be adjusted

kp = 20;
kd = 400;
Kd = kd*eye(3);
m = 180;
r = 2;

% model parameters
I = m*r^2*eye(3);         % inertia matrix
I_inv = inv(I);

% constants
deg2rad = pi/180;
rad2deg = 180/pi;

phi = -5*deg2rad;              % initial Euler angles
theta = 10*deg2rad;
psi = -20*deg2rad;
q = euler2q(phi,theta,psi);    % transform initial Euler angles to q

phi_d = 0;
theta_d = 15*deg2rad;
psi_d = 0;
q_d = euler2q(phi_d,theta_d,psi_d);
```

```
w = [0 0 0]';                    % initial angular rates
table = zeros(N+1,14);           % memory allocation

%% FOR-END LOOP
eps_tild = zeros(3,N);
for i = 1:N+1,
   t = (i-1)*h;                     % time

   eps_tild(:,i) = q_d(1)*q(2:4)-q(1)*q_d(2:4)-Smtrx(q_d(2:4))*q(2:4);
   w_d = deg2rad*[-0.5*sin(theta)*cos(0.05*t);
             -1.5*sin(0.1*t)*cos(phi)+cos(theta)*sin(phi)*0.5*cos(0.05*
                t);
             1.5*sin(phi)*sin(0.1*t)+cos(theta)*cos(phi)*0.5*cos(0.05*t
                )];

   w_tild = w-w_d;

   tau = -Kd*w_tild - kp*eps_tild(:,i);     % control law

   [phi,theta,psi] = q2euler(q); % transform q to Euler angles
   [J,J1,J2] = quatern(q);       % kinematic transformation matrices

   q_dot = J2*w;                            % quaternion kinematics
   w_dot = I_inv*(Smtrx(I*w)*w + tau);  % rigid-body kinetics

   table(i,:) = [t q' phi theta psi w' tau'];  % store data in table

   phi_d = 0;
   theta_d = 15*cos(0.1*t)*deg2rad;
   psi_d = 10*sin(0.05*t)*deg2rad;
   q_d = euler2q(phi_d, theta_d, psi_d);
   q = q + h*q_dot;                          % Euler integration
   w = w + h*w_dot;

   q  = q/norm(q);                   % unit quaternion normalization
end

%% PLOT FIGURES
t       = table(:,1);
q       = table(:,2:5);
phi     = rad2deg*table(:,6);
theta   = rad2deg*table(:,7);
psi     = rad2deg*table(:,8);
w       = rad2deg*table(:,9:11);
tau     = table(:,12:14);


figure (1); clf;
hold on;
plot(t, phi, 'b');
plot(t, theta, 'r');
plot(t, psi, 'g');
hold off;
grid on;
legend('\phi', '\theta', '\psi');
title('Euler angles');
```

```
xlabel('time [s]');
ylabel('angle [deg]');

figure (2); clf;
hold on;
plot(t, w(:,1), 'b');
plot(t, w(:,2), 'r');
plot(t, w(:,3), 'g');
hold off;
grid on;
legend({'$\dot{\phi}$', '$\dot{\theta}$', '$\dot{\psi}$'},'Interpreter','latex');
title('Angular velocities');
xlabel('time [s]');
ylabel('angular rate [deg/s]');

figure (3); clf;
hold on;
plot(t, tau(:,1), 'b');
plot(t, tau(:,2), 'r');
plot(t, tau(:,3), 'g');
hold off;
grid on;
legend('x', 'y', 'z');
title('Control input');
xlabel('time [s]');
ylabel('input [Nm]');

figure (4); clf;
hold on;
plot(t, rad2deg*eps_tild(1,:), 'b');
plot(t, rad2deg*eps_tild(2,:), 'r');
plot(t, rad2deg*eps_tild(3,:), 'g');
hold off;
grid on;
legend({'$\tilde{\phi}$', '$\tilde{\theta}$', '$\tilde{\psi}$'},'Interpreter','latex');
title('Tracking error');
xlabel('time [s]');
ylabel('error [deg]');
```

Problem2.m

```
% M-script for numerical integration of the attitude dynamics of a rigid
% body represented by unit quaternions. The MSS m-files must be on your
% Matlab path in order to run the script.
%
% System:                          .
%                                  q = T(q)w
%                                  .
%                                I w - S(Iw)w = tau
% Control law:
%                                tau = constant
%
% Definitions:
```

```
%                               I = inertia matrix (3x3)
%                               S(w) = skew-symmetric matrix (3x3)
%                               T(q) = transformation matrix (4x3)
%                               tau = control input (3x1)
%                               w = angular velocity vector (3x1)
%                               q = unit quaternion vector (4x1)
%
% Author:                      2018-08-15 Thor I. Fossen and Hkon H.
    Helgesen

%% USER INPUTS
h = 0.1;                        % sample time (s)
N  = 50000;                       % number of samples. Should be adjusted

% No integral effect
kp = 0.01;
kd = 0.08;
ki = 0;

% With integral effect
kp = 0.09;
kd = 0.22;
ki = 0.00008;

T = 20;
K = 0.1;
b = 0.001;
U = 5;

% constants
deg2rad = pi/180;
rad2deg = 180/pi;

% initial states
x = 0;
y = 100;
psi = 0;
r = 0;
z = 0;

X = [x y psi r z];

x_dot = U;
y_dot = 0;
y_dot_C = 0;
psi_dot = 0;
r_dot = 0;
z_dot = 0;

X_dot = [x_dot y_dot psi_dot r_dot z_dot];

y_int = 0;
u = 5;
v = 0;

table = zeros(N+1,6);           % memory allocation
```

```
%% FOR-END LOOP

for i = 1:N+1,
    t = (i-1)*h;                    % time

    delta = -kp*y-kd*y_dot_C-ki*z;
    if delta > deg2rad*20
        delta = deg2rad*20;
    elseif delta < deg2rad*-20
        delta = deg2rad*-20;
    end
    delta;

    x_dot = U*cos(deg2rad*psi);
    y_dot = U*sin(deg2rad*psi);
    y_dot_C = U*psi;
    psi_dot = r;
    r_dot = (K*delta+b-r)/T;
    z_dot = y;

    x = x + h*x_dot;
    y = y + h*y_dot;
    psi = psi + h*psi_dot;
    r = r + h*r_dot;
    z = z + h*z_dot;

    table(i,:) = [t x y psi r delta];  % store data in table

end

%% PLOT FIGURES
t        = table(:,1);
x        = table(:,2);
y        = table(:,3);
psi      = table(:,4);
r        = rad2deg*table(:,5);
tau      = rad2deg*table(:,6);


figure (1); clf;
hold on;
plot(y, x, 'b');
hold off;
grid on;
title('Position');
xlabel('East [m]');
ylabel('North [m]');

figure (2); clf;
hold on;
plot(t, r, 'b');
hold off;
grid on;
legend({'$r$'},'Interpreter','latex');
title('Angular velocity');
```

```
xlabel('time [s]');
ylabel('angular rate [deg/s]');

figure (3); clf;
hold on;
plot(t, tau(:,1), 'r');
plot(t, psi, 'b');
hold off;
grid on;
legend({'$\delta$','$\psi$'},'Interpreter','latex');
title('Control input and heading');
xlabel('time [s]');
ylabel('angle [deg]');
```

References

[1] B. F. J.G. Balchen, T. Andresen, *Reguleringsteknikk*. Institutt for teknisk kybernetikk, NTNU, 2016.

[2] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

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