Hello Everyone !

Autonomous Mobile Robotics Course
Final Presentation

4 thruster-Boat drive control using Gazebo and ROS2

By: Walid Shaker

Content

1. Boat Dynamic Model
2. Feedback Linearization with Proportional-Derivative (PD) Control
3. odeint / RK4
4. Recover Thrust Forces and Rudder Angles
5. ROS2/ Gazebo

Boat Dynamic Model

1 Boat Dynamic Model

Four-thrusters boat configuration model combines four movable thrust forces oriented w.r.t. x-axis by the rudder angle δ, as shown in Fig. 1.

where

- System state $X = [x, y, \psi, u, v, r]^T$
- x, y – position in global coordinate system.
- ψ - rotation around Z-axis.
- u, v - longitude and latitude velocities in boat local frar
- r – angular velocity w.r.t. Z-axis.
- δ - rudder angle w.r.t. x-axis.

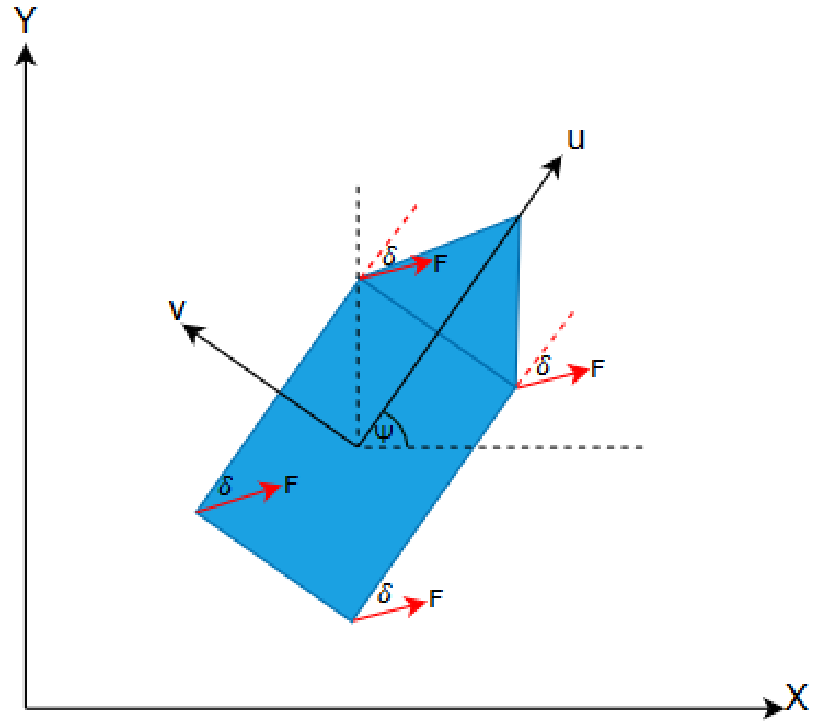


Figure 1: Four-thrusters boat configuration model.

Boat Dynamic Model

The system dynamics has the form of

$$
\begin{aligned}
&[\dot{u}, \dot{v}, \dot{r}]^T = M^{-1}\left(\tau - N * [u, v, r]^T\right) \\
&\dot{x} = \cos(\psi)u - \sin(\psi)v \\
&\dot{y} = \sin(\psi)u + \cos(\psi)v \\
&\dot{\psi} = r
\end{aligned}
\tag{1}
$$

where

$$
\begin{aligned}
M &= 10^3 * \begin{bmatrix} 2.131 & 0 & 0 \\ 0 & 2.93 & 0.34 \\ 0 & 0.34 & 3.05 \end{bmatrix} \\
N &= 10^5 * \begin{bmatrix} 4.74 & 0 & 0 \\ 0 & 0.0053 & -0.0008 \\ 0 & 0.0083 & 0.0109 \end{bmatrix} * U \\
U &= \sqrt{u^2 + v^2}
\end{aligned}
\tag{2}
$$

where U is the speed of the horizontal plane, it could be linearized as $U \approx u$ if the boat moves at constant speed or or at least slowly varying) forward speed [1].

Boat Dynamic Model

Let

- $\eta = [x, y, \psi]^T$ boat pose in global frame.
- $\nu = [u, v, r]^T$ represents longitude, latitude, and angular velocities in local frame.

From dynamic Eq. 1, it can be formulated as follows:

$$M\dot{\nu} + N(\nu)\nu = \tau \tag{3}$$

where M is the inertial matrix defined in Eq. 2 and N is the added mass coriolis and centripetal terms together with hydrodynamic damping terms are collected into the matrix $N(\nu) = C(\nu) + D(\nu)$ [1].

Boat Dynamic Model

From the dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{4}$$

This can be written as follows:

$$\dot{\eta} = R(\psi)\nu \tag{5}$$

From Eq. 5, global velocity vector (ν) can be described as:

$$\nu = R^T(\psi)\dot{\eta} \tag{6}$$

Detour:

Identity matrix I can be written as $I = R(\psi)R(\psi)^T$.

Differentiating this expression leads to: $R(\dot{\psi})R(\psi)^T + R(\psi)[\dot{R(\psi)^T}] = 0$

Thus $[\dot{R(\psi)^T}] = -R(\psi)^T \dot{(R(\psi))} R(\psi)^T$

In Eq. 6:

$$\dot{\nu} = [\dot{R(\psi)^T}]\dot{\eta} + R^T(\psi)\ddot{\eta} \tag{7}$$

Substitute Eq. 6 and Eq. 7 in Eq. 3:

$$MR^T(\psi)\ddot{\eta} + [MR(\dot{\psi})^T + N(\nu)R^T(\psi)]\dot{\eta} = \tau \tag{8}$$

Let

- $MR^T(\psi) = M_{inertia}$: new inertia matrix.
- $M[\dot{R(\psi)^T}] + N(\nu)R^T(\psi) = C_{coriolis}$: new coriolis matrix.

Then Eq. 8 can be formulated as:

$$M_{inertia}\ddot{\eta} + C_{coriolis}\dot{\eta} = \tau \tag{9}$$

Feedback Linearization- PD control

Let $\eta_d = [x_d, y_d, \psi_d]^T$ represents the desired pose. Then τ should be selected to perform the feedback linearization with PD controller as follows:

$$\tau = M_{inertia}\ddot{\eta}_d + C_{coriolis}\dot{\eta} - M_{inertia}[k_d(\dot{\eta} - \dot{\eta}_d) + k_p(\eta - \eta_d)] \tag{10}$$

Substitute Eq. 10 in Eq. 9:

$$(\ddot{\eta} - \ddot{\eta}_d) + k_d(\dot{\eta} - \dot{\eta}_d) + k_p(\eta - \eta_d) = 0 \tag{11}$$

Let error $(e) = \eta - \eta_d$, then Eq. 11 can be written as follows:

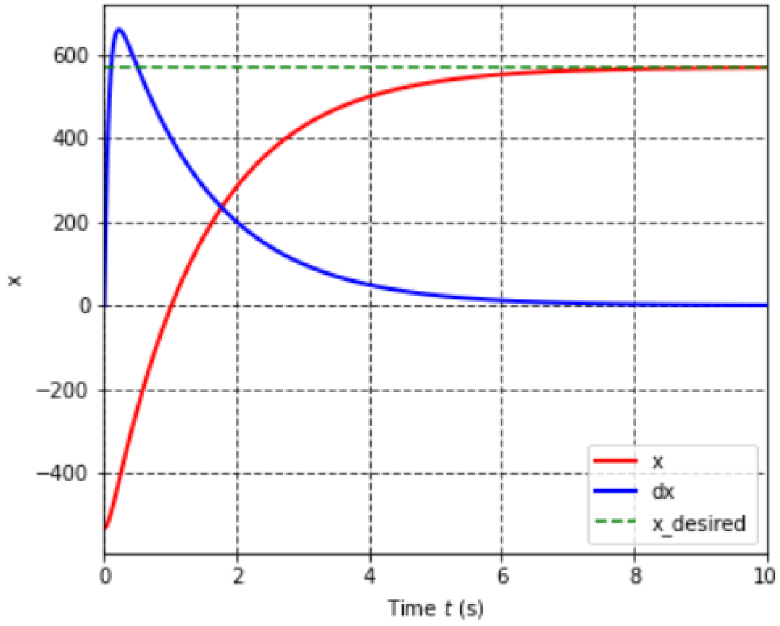
$$\ddot{e} + k_d\dot{e} + k_p e = 0 \tag{12}$$

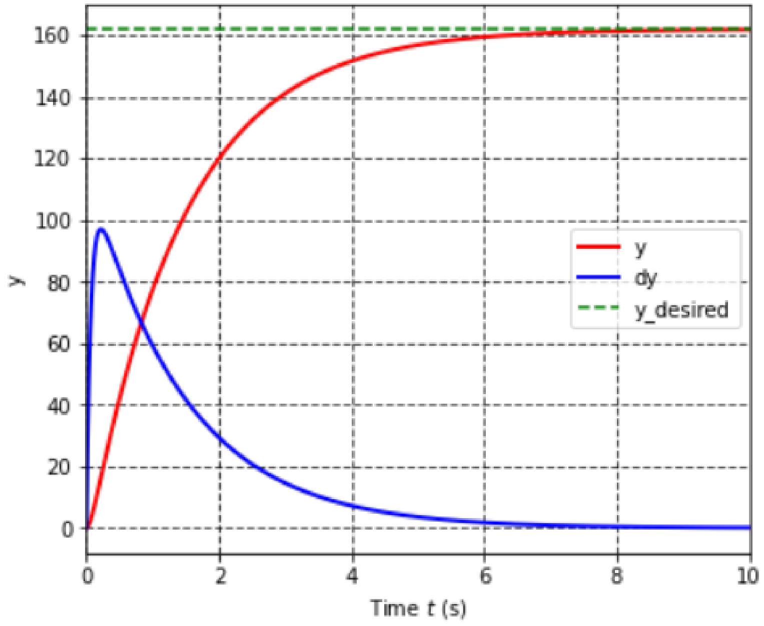
Odeint/RK4

```
def rungeKutta(self):
    if(self.state is not None):

        _, dX = self.control(self.state)
        k1 = self.Ts*dX

        _, dX = self.control(self.state+(k1/2))
        k2 = self.Ts*dX

        _, dX = self.control(self.state+(k2/2))
        k3 = self.Ts*dX

        _, dX = self.control(self.state+k3)
        k4 = self.Ts*dX

        state_n = self.state  + (1.0/6.0)*(k1 + 2*k2 + 2*k3 + k4)
        self.state  = state_n

        Tau, _ = self.control(self.state)
```

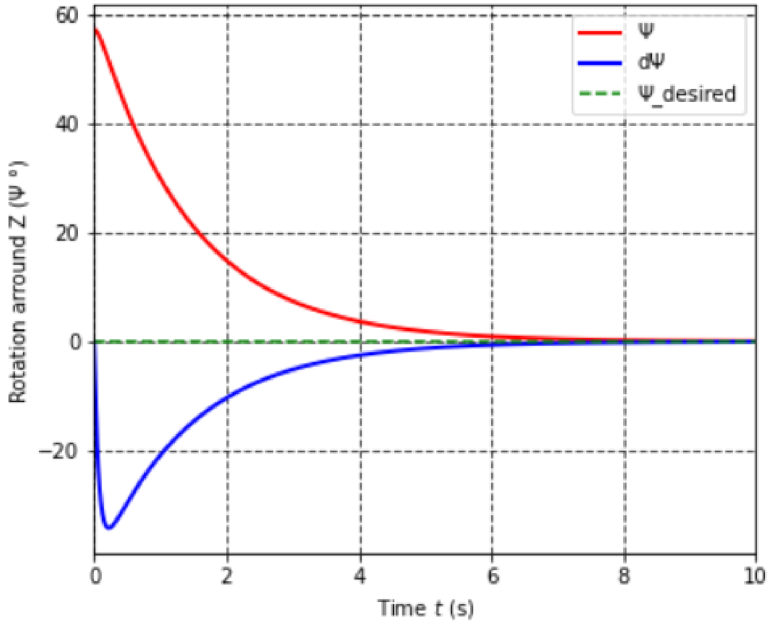
```
def solve(self):
    sol = odeint(self.sys_ode, self.init_state, self.t)
    self.x, self.y, self.psi, self.dx, self.dy, self.dpsi = sol[:,0], sol[:,1], sol[:,2], sol[:,3], sol[:,4], sol[:,5]
```

Odeint/RK4 Regulation



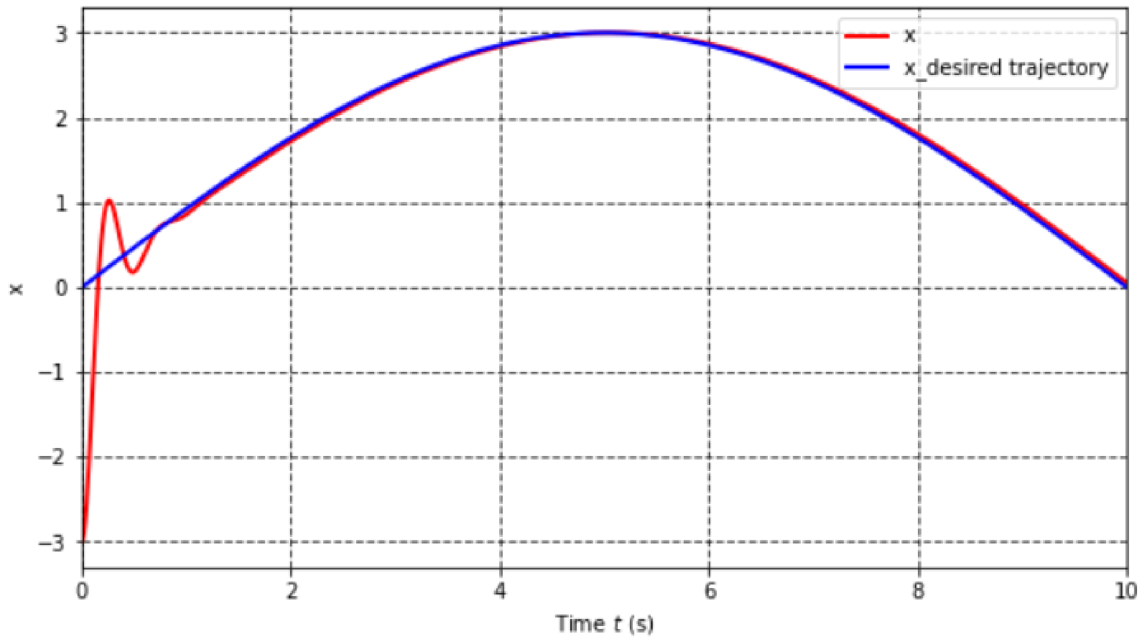


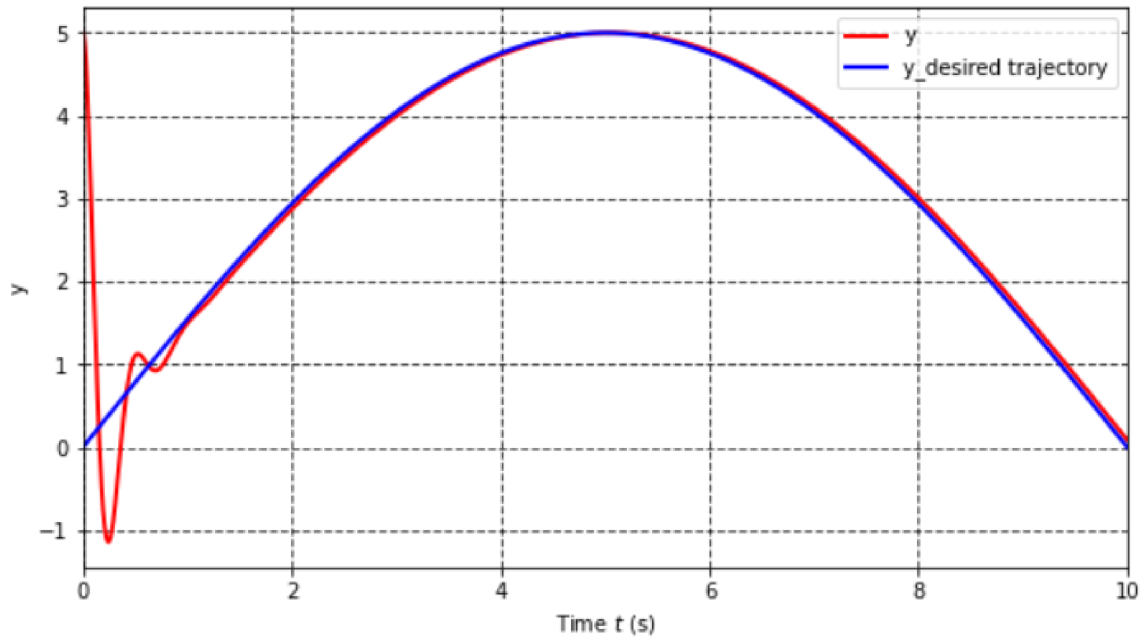
as shown in graph x and y are following x,y desired



this Ψ plot is at dΨ = 0.0 degree/second

Odeint/RK4 Tracking





Notebook

Thrust forces – Rudder Angles

The calculated τ in Eq. 10 can also be formulated as:

$$\tau = \begin{bmatrix} \Sigma f_{ix} \\ \Sigma f_{iy} \\ \Sigma r_i \times f_i \end{bmatrix} \tag{13}$$

where r_i is the location of the thruster w.r.t. to local frame, consequently:

$$\tau = \begin{bmatrix} c(\delta_1) & c(\delta_2) & c(\delta_3) & c(\delta_4) \\ s(\delta_1) & s(\delta_2) & s(\delta_3) & s(\delta_4) \\ x_1 s(\delta_1) - y_1 c(\delta_1) & x_2 s(\delta_1) - y_2 c(\delta_2) & x_3 s(\delta_1) - y_3 c(\delta_3) & x_4 s(\delta_1) - y_4 c(\delta_4) \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{14}$$

where c and s are cos and sin trigonometric functions.

Thrust forces – Rudder Angles

Eq. 14 can be formulated as:

$$\tau = B(r, \delta)F \tag{15}$$

where B is the input mapping matrix and f is the control input.

Another representation for Eq. 14 can combine rudder angles in thrust forces. Hence B does not depend on rudder angles any more.

$$\tau = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ -y_1 & x_1 & -y_2 & x_2 & -y_3 & x_3 & -y_4 & x_4 \end{bmatrix} \begin{bmatrix} f_{1x} \\ f_{1y} \\ f_{2x} \\ f_{2y} \\ f_{3x} \\ f_{3y} \\ f_{4x} \\ f_{4y} \end{bmatrix} \tag{16}$$

Consequently, Eq. 15 can be written as:

$$\tau = B(r)F \tag{17}$$

In order to solve such a system and obtain F, we need to find the smallest 2-norm F which at the same time provides the least residual e.

$$e = BF - \tau \tag{18}$$

As the minimum of $||e||_2$ coincides with the minimum of $(BF-\tau)^T(BF-\tau)$, then the solution to this least squares problem is given by a pseudoinverse similar to finding the extremum as follows:

$$\begin{gathered} 2B^T(BF-\tau) = 0 \\ B^TBF = B^T\tau \\ F = (B^TB)^{-1}B^T\tau \end{gathered}$$

then F can be calculated as:

$$F = B^+\tau \tag{19}$$

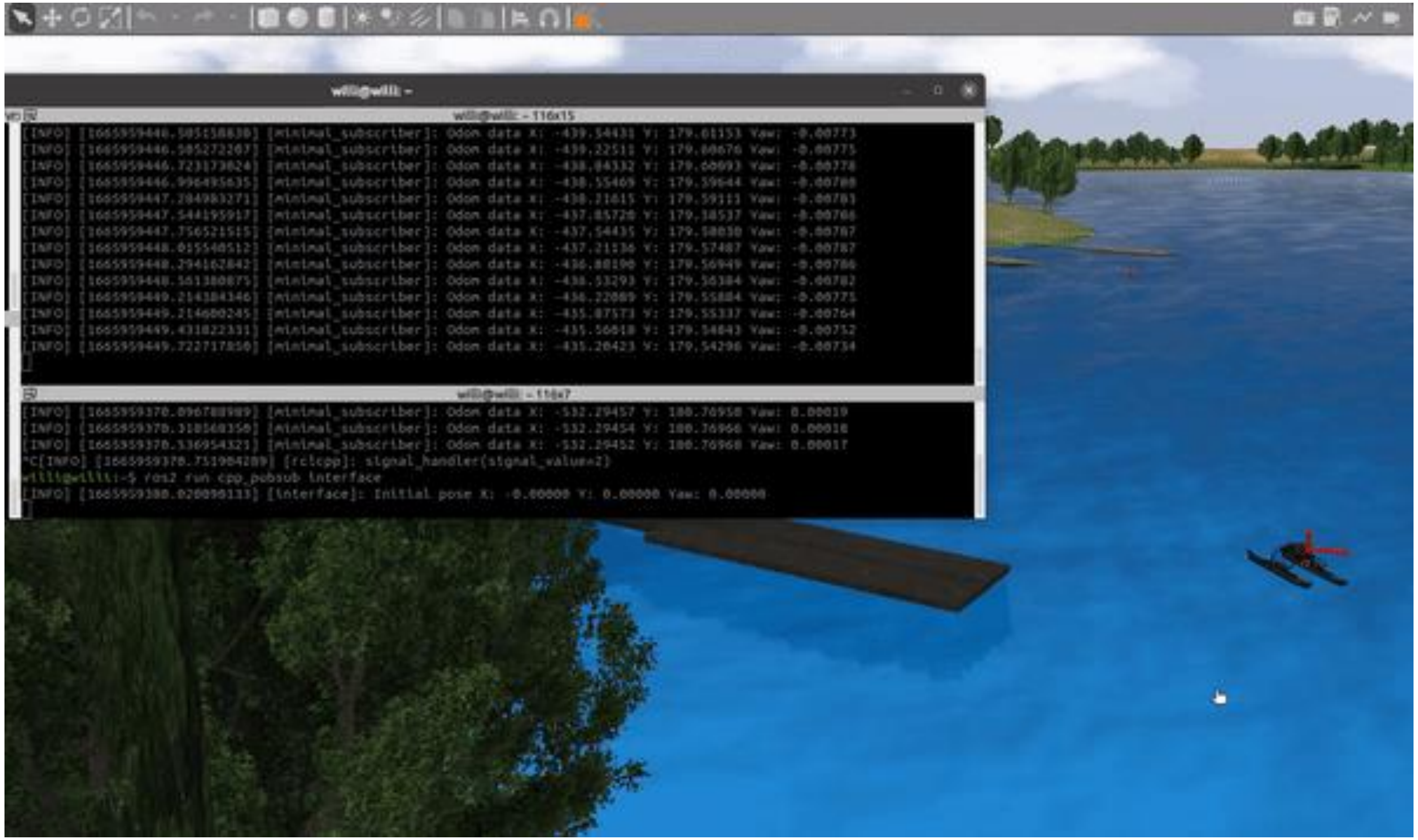
Then the force for each thruster f_i and the corresponding rudder angle δ_i can be obtained as:

$$f_i = \sqrt{f_{ix}^2 + f_{iy}^2} \tag{20}$$

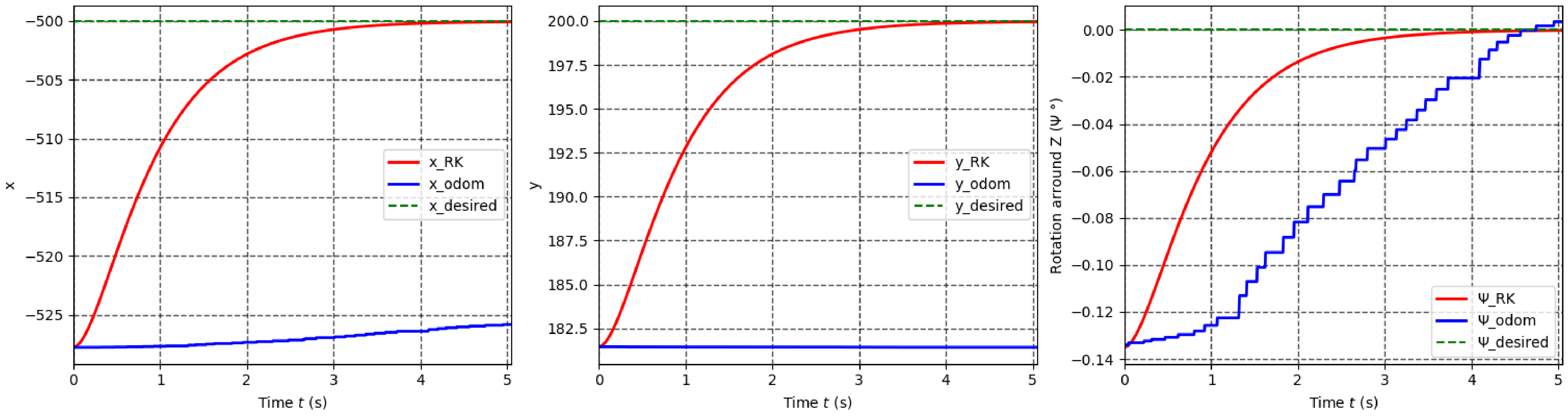
$$\delta_i = \arctan2(f_{iy}, f_{ix}) \tag{21}$$

ROS2- Gazebo

ROS2- Gazebo



ROS2- Gazebo



Parameters estimation

References

[1] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons, 2011.

Thank You !