MAE 3260

Fall 2025
Final Group Work: Exploring a System of Interest

Ship Happens: A Guide to Keeping Things Under Control

Topic of Interest: Mathematical Model and Analysis of Sailboat Dynamics

Students/Roles

Student	Task/Role
Chinmay Mangalgi	Created MATLAB animation script modeling a sailboat with a PD controller navigating to a waypoint with a heading input.
Olivia Santiago	Develop a MATLAB script to linearize equations of motion using Jacobian to attain a state space model. Also discuss controllability.
Nitya Shukla	Identified simplifying assumptions and limitations of model. Contributed to write-up for discussion of simulation results.
Poon Jeerapat	Built a mathematical model of the sail boat and its state space. In cooperate the system dynamic, along with the sail and rudder forces into equations.

Key Concepts: State-Space Modelling, ODEs, Steady-State Behavior, Parameter Estimation, PD Control

Abstract

This project develops a simplified dynamic model of a sailboat using a 2 dimensional representation, neglecting rolling and pitching movement. This model would allow us to focus solely on planar motion, yawing and linear position of the boat. This system is chosen because it captures feedback control as a result of the interaction between aerodynamics, hydrodynamics, and control systems. The model would include the rotational degree caused by yaw motion from the rudder and sail input. The state variables include linear position (x,y), yaw angle (ψ), forward velocity (v), and yaw rate (w). The control inputs are the rudder angle and maximum sail angle. Moreover, we intend to account for disturbances such as wind speed and friction (aerodynamics coefficient).

1. Introduction

Sailboats are nonlinear systems where aerodynamic forces on the sail and hydrodynamic forces on the hull and rudder interact with control inputs. Accurately modeling these complex interactions is essential to understanding the boat's stability and control. In this project, we develop a two-dimensional model that represents only the planar degrees of freedom: surge, sway, and yaw, while neglecting heave, roll, and pitch. The main points of this project are: deriving the nonlinear equations of motion that describe horizontal motion and yaw rotation; assuming steady sailing conditions, linearizing those equations with Jacobian matrices; formulating a state-space model around the linearized equations; and designing controllers that stabilize yaw. Finally, we simulate the nonlinear dynamics in MATLAB and analyze the system response.

2. System Modeling

2.1 Coordinate Frames and Notation

In the system we are exploring, we set two frames: inertial frame and body frame. The inertial frame has its origin fixed on the water surface with x pointing east and y pointing in north direction ($J = \{X, Y\}$). The body frame, $B = \{x_b, y_b\}$, has its origin at the boat's center of mass and x_b points along the boat's centerline toward the bow while y_b points to port.

For the state variables, we only consider 2D planar motion: $x = [x; y; \psi; v; \omega]$
x, y is the position of boat center of mass in inertial frame; ψ is the yaw angle of the boat; v is the speed of the boat; ω is the yaw rate

For the control inputs, we set: $u = \left[\delta_r; \delta_s\right]$
δ_r is the rudder angle in relation to the boat axis; δ_s is the sail angle in relation to the boat axis
Due to the complexity introduced by simultaneously modeling both control inputs, only the rudder angle is varied in the model. Further justification for this choice is provided in simplifying assumption (7).

Moreover, in this system, we take account of the disturbances, which we choose to be:
a: true wind speed; ψ_w: wind direction in the inertial frame

Using the tabulated values from *Control Algorithms for a Sailboat Robot with a Sea Experiment* by Clement [1], we set our parameters to the following: M (mass) = 200 kg; J (rotational inertia) = 1000 kg $\cdot\, m^2$; α_d (drift coefficient) = 0.5; α_v (longitudinal friction coefficient) = 100; α_ω (rotational friction coefficient) = 6000; α_s (lift coefficient for the sail) = 1750; α_r (lift coefficient for the rudder) = 250
Geometric moment arms of sail and rudder: $p_6 = 2m, p_7 = 2m, p_8 = 4m$

2.2 Nonlinear Equations of Motion

First, we address the kinematic equations, consisting of the boat translation motion in the inertial frame and yaw kinematics. Boat translation in the inertial frame models the boat's velocity and the drift motion caused by the wind:

$x' = v\cos(\psi) + \alpha_d a\cos(\psi_\omega)$

$y' = v\sin(\psi) + \alpha_d a\sin(\psi_\omega)$

The yaw kinematics can be described by $\psi' = \omega$.

We define the forces and moments impacting the sailboat's motion. The effective sail angle δ_v is the angle between the sail and the boat axis: $\delta_v = sign(\sin(\psi - \psi_\omega))\delta_s$

The sail force and rudder force are accounted for in the following equations:

$f_s = \alpha_s a\sin(\psi - \psi_\omega + \delta_v)$ – force perpendicular to sail

$f_r = \alpha_r v^2 \sin(\delta_r)$ – force perpendicular to rudder

We identify the relationship between the two and utilize the force and moment balance:

1. Balance along the x_b axis: $v' = \frac{1}{M}(f_s\sin(\delta_v) - f_r\sin(\delta_r) - \alpha_v v^2)$

   $f_s\sin(\delta_v)$ is the component of sail force the solely contribute to the forward motion

   $f_r\sin(\delta_r)$ is the drag against this particular motion and $\alpha_v v^2$ is the hull drag.

2. Moment balance around the z-axis: $w' = \frac{1}{J}(f_s(p_6 - p_7\cos(\delta_v)) - p_8 f_r\cos(\delta_r) - \alpha_\omega \omega$

   $f_s(p_6 - p_7\cos(\delta_v)$ is the sail moment about the center of mass

   $p_8 f_r\cos(\delta_r)$ is the rudder moment about center of mass and $\alpha_\omega \omega$ is rotational damping.

Combining all equations, yields the state-space model:

$x' = v\cos(\psi) + \alpha_d a\cos(\psi_\omega)$

$y' = v\sin(\psi) + \alpha_d a\sin(\psi_\omega)$

$\psi' = \omega$

$v' = \frac{1}{M}(\alpha_s a\sin(\psi - \psi_\omega + \delta_v)\sin(\delta_v) - \alpha_s a\sin(\psi - \psi_\omega + \delta_v)\sin(\delta_r) - \alpha_v v^2)$

$w' = \frac{1}{J}(\alpha_s a\sin(\psi - \psi_\omega + \delta_v)(p_6 - p_7\cos(\delta_v)) - p_8\alpha_s a\sin(\psi - \psi_\omega + \delta_v)\cos(\delta_r) - \alpha_\omega \omega$

2.3 Simplifying Assumptions

1. We applied the small angle approximation to simplify the translational and rotational position calculation of the boat. This assumption indicates that the angle between the vertical or horizontal axis and the front of the boat is minimal. No external force is assumed to cause a strong rotational change in the boat's overall trajectory, meaning that the yaw coordinate does not vary substantially for each iterated time increment. The small-angle approximation allows us to define the velocity coordinate components and express the model in state-space form. Without applying this approximation, the model contains trigonometric functions of velocity, which makes it more difficult to express the model equations in state-space form.
2. The boat is assumed to have constant rotational inertia J, meaning that the body's mass is distributed relatively uniform with respect to the axis of rotation. This simplifies calculations corresponding to the yaw rate of the boat. A single parameter can be used to represent the rotational inertia of the boat, rather than a summation of point masses and distances to account for complexities and non-uniform distributions in mass.
3. The boat is assumed to have a constant mass throughout the simulation and is treated as a vertically and rotationally stable body. No water or external mass is expected to enter the hull, and no mass is removed from the boat during operation. As a result, the model does not include effects such as flooding, ballast changes, cargo status, or fuel consumption, and both the total mass and the center-of-gravity of the boat remain fixed.
4. The modelled boat is thin-shelled and rectangular in shape. This removes any added complexity caused by nonuniformities in the boat hull geometry. The assumption also simplifies the calculation of the drag force acting on the boat, as the drag acting on the hull is directly proportional to the boat's cross-sectional area. Since the cross-sectional area remains constant when the boat is modeled rectangular in shape, the drag force acting on the hull is a function of a constant and the squared velocity acting at any given time value.
5. The boat's flow is assumed to be turbulent, as opposed to laminar; fluid moves around the boat hull at a relatively high Reynolds number. Regions of separation and vortices on the hull are inherently turbulent due to the boat's geometric condition. The turbulent flow around the boat results in the drag force to be proportional to velocity squared. Because The model requires
6. Because the system is modelled in a horizontal plane, no pitching or heeling motion of the boat is considered. In other words, the boat's rotation with respect to the two axes within the horizontal plane of motion is negligible; the only rotation accounted for in this model is that with respect to the axis perpendicular to the horizontal plane. Consequently, the model is composed of three degrees of freedom: two translational coordinates and one rotational coordinate $(x,\ y,\ \psi)$.
7. A typical sailboat relies on two primary internal control systems: rudder control and sail trim. Modeling both systems together can make the dynamical model of the sailboat convoluted and difficult to analyze; therefore, it is essential to introduce simplifying assumptions about these control systems so that the model's analysis becomes easier to understand. In the model described in this report, sail trim angle is assumed to remain constant over the course of the boat's motion. Rudder angle, on the other hand, is modeled to vary, causing prominent changes in the overall dynamics of the sailboat. These are reasonable assumptions to make, as the inputted sail trim is highly dependent on external variables such as wind direction and speed, which would highly complicate the model. The rudder angle more directly influences the trajectory of the boat and is less reliant upon external forces and conditions to yield predictable results.

3. Linearization

3.1 Jacobian Derivation

Nonlinear state vector $\vec{f}$ is :

$$\vec{f} = \begin{bmatrix} v \\ v\omega \\ \omega \\ -\frac{250}{m} vu^2 - \frac{100}{m} v^2 \\ -\frac{1000}{J} vu - \frac{6000}{J} \omega \end{bmatrix}$$

Finding $\Rightarrow \dot{x} = Ax + Bu$

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \theta \\ v \\ w \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \omega} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \omega} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \omega} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial \theta} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \omega} \\ \frac{\partial f_5}{\partial x} & \frac{\partial f_5}{\partial y} & \frac{\partial f_5}{\partial \theta} & \frac{\partial f_5}{\partial v} & \frac{\partial f_5}{\partial \omega} \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v \\ w \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial u_1} \\ \frac{\partial f_2}{\partial u_1} \\ \frac{\partial f_3}{\partial u_1} \\ \frac{\partial f_4}{\partial u_1} \\ \frac{\partial f_5}{\partial u_1} \end{bmatrix} \begin{bmatrix} \delta_r \end{bmatrix}$$

Linear output map :

$$\vec{g} = \begin{bmatrix} \theta \\ w \end{bmatrix}$$

Finding $\Rightarrow \dot{y} = Cx + Du$

$$\frac{d}{dt}\begin{bmatrix} \theta \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_1}{\partial y} & \frac{\partial g_1}{\partial \theta} & \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial \omega} \\ \frac{\partial g_2}{\partial x} & \frac{\partial g_2}{\partial y} & \frac{\partial g_2}{\partial \theta} & \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial \omega} \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v \\ w \end{bmatrix} + \begin{bmatrix} \frac{\partial g_1}{\partial u_1} \\ \frac{\partial g_2}{\partial u_1} \end{bmatrix} \begin{bmatrix} \delta_r \end{bmatrix}$$

The Jacobian is important because it makes nonlinear systems easier to analyze and design controllers for. Most real physical systems are inherently nonlinear. For example, drag relies on a velocity squared component, the rudder force requires the velocity times the input and of course trig functions exist. Using the Jacobian, we can develop a linear approximation of a nonlinear system without sacrificing accuracy and precision.

3.2 Linearized State-Space Model

Using the Jacobian to linearize the equations of motion and get a space space model, we get the state space matrices:

```
A =

  1.0e+04 *

         0         0         0    0.0001         0
         0         0         0    0.0002    0.0005
         0         0         0         0    0.0001
         0         0         0   -0.0005         0
         0         0         0   -0.6545   -7.5000

B =

  1.0e+04 *

         0
         0
         0
  -0.0007
  -6.2500

C =

     0     0     1     0     0
     0     0     0     0     1

D =

     0
     0
```

Where $A = \frac{\partial f}{\partial x}$, $B = \frac{\partial f}{\partial u}$, $C = \frac{\partial g}{\partial x}$, $D = \frac{\partial g}{\partial u}$

Matrix A tells us how the boat evolves on its own, assuming no rudder input change. Each nonzero term shows a physical coupling. For example, the 1 in

- $A_{1,4}$ means forward position integrates forward speed

- $A_{2,4} = 2$ means that increasing forward speed increases the lateral motion if yawing
- $A_{2,5} = 5$ means that increasing the yaw rate increases the lateral motion if moving forward
- $A_{3,5} = 1$ means yaw rate integrates heading
- $A_{4,4} =- 5$ indicates strong surge damping.
- $A_{5,4} =- 6545$ means that the forward speed strongly affects yaw acceleration
- $A_{5,5} =- 75000$ means that there is significant yaw damping

What we can take from this is that fast boats amplify rudder-induced yaw and the small polar moment of inertia value causes fast yaw decay. This makes sense physically.

Matrix B tells us how the rudder instantly accelerates the states.

- B_1 to B_3 tells us that there is no direct effect on position or heading. Rather position and heading respond indirectly via velocity and yaw rate
- $B_4 =- 6.54$ tells us physically that the rudder deflection greatly increases drag and therefore the forward speed decreases when the rudder is applied
- $B_5 =- 62500$ tells us that the rudder is extremely effective at producing yaw acceleration

Matrix C tells us which states are observable outputs. This is influenced by the fact that we are designing a state space model designed for the implementation of a yaw stabilization controller.

- $C_{1,3} = 1$ tells us that yaw angle is an observable output
- $C_{2,5} = 1$ tells us that yaw rate is an observable output

Matrix D is a zero matrix which in this context tells us that the rudder angle does not instantaneously change heading or yaw rate and that these effects occur by boat dynamics.

In conclusion, matrix A reveals the natural motion and damping of the boat, matrix B reveals how strongly rudder angle accelerates the boat, matrix C reveals what variables matter for designing a controller, and matrix D verifies the coupling of state variables and inputs.

3.3 Controllability Analysis

Before designing a controller, it is important to check if the system is controllable given the state space model. A system is controllable if using the control input u, one can move the system from any initial state to any desired state in a finite time. This means answering the question of can the rudder influence all the states, and can it affect that state indirectly through dynamics coupling.

To check the controllability, we compute the controllability matrix using a built-in MATLAB function. Then we determine the rank of this matrix. If the rank equals our number of states, then the system is controllable. If not, then we must alter our state space model or control input to make our system controllable.

In our case, we got a rank of 1 for our controllability matrix meaning that our system is currently not controllable. Given the scope of this course, we would need to reduce the order of our state space and apply some initial conditions that make our system controllable with just the rudder angle.

For example, if we reduce our state variables to just those that are affected by rudder angle directly, we can develop a controller and set requirements and desired pole locations. This is especially useful if looking solely at yaw stabilization over other outputs or observations.

4. Simulation Output

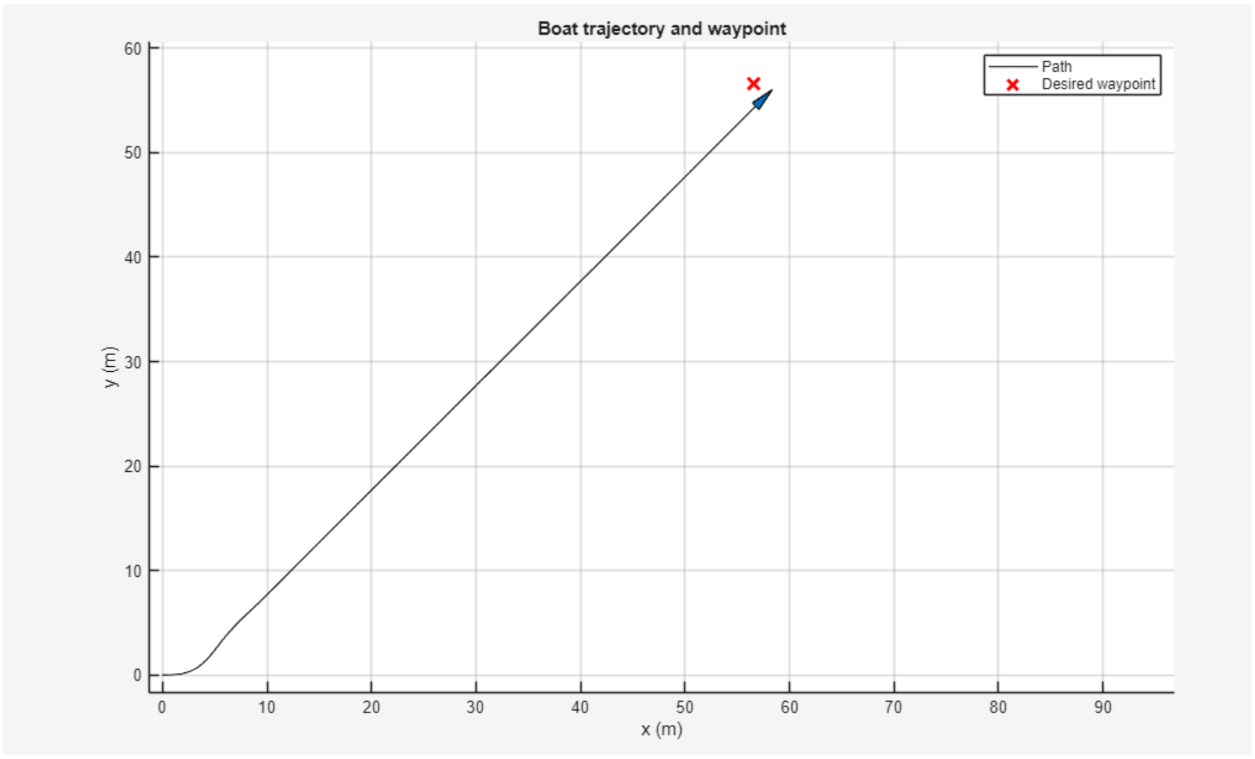

Fig. 1: The simulated sailboat begins at the origin and performs a heading change maneuver to align with a reference 45° input. After an initial turn, the boat tracks a nearly straight-line trajectory toward the target located approximately 55 m east and 55 m north, demonstrating closed-loop heading control behavior. The boat does not reach the exact location of the waypoint due to no integral control, but is close enough to it to neglect errors.

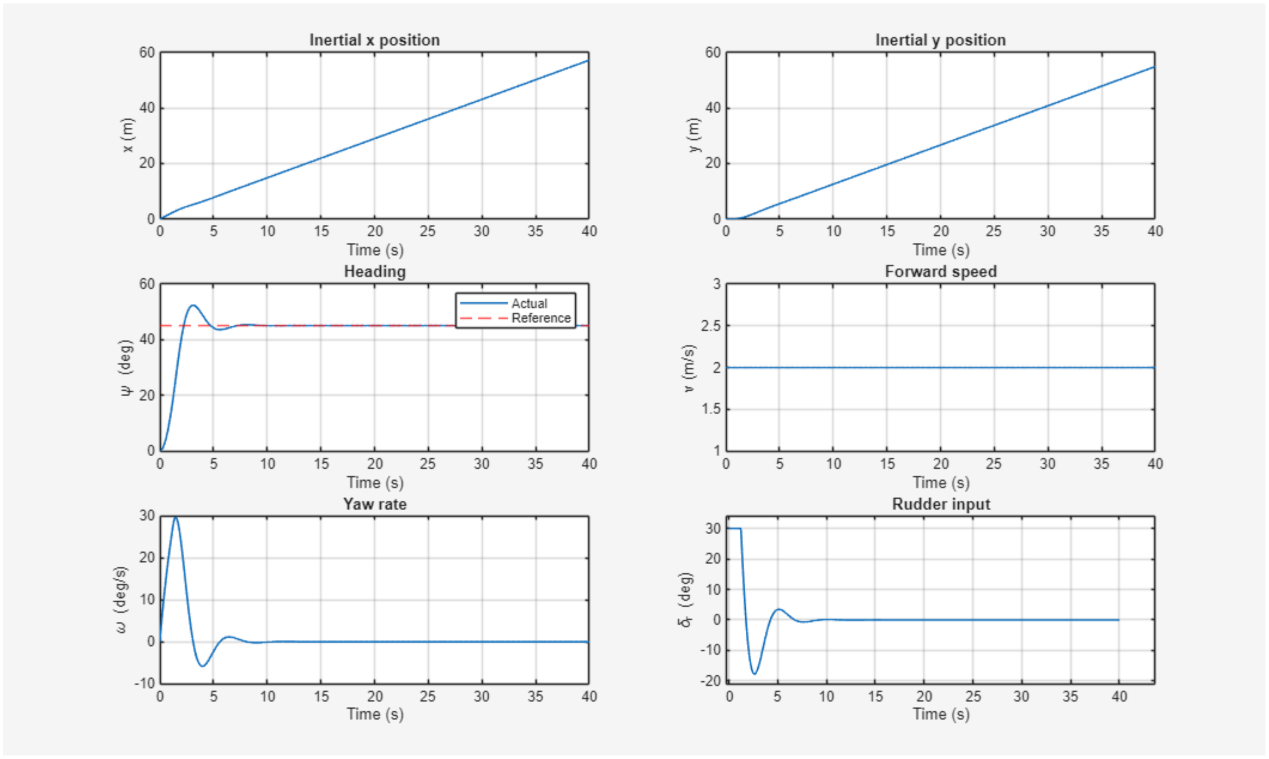

Fig. 2: Time history of the system states and control input under PD heading control, with K_p=2.0 and K_d=0.8 (arbitrarily chosen).
Top row: inertial x and y positions increase linearly as the boat moves at constant surge speed.
Middle row: heading (ψ) responds with about at 13% overshoot before converging near the 45° reference, while forward speed (v) remains nearly constant as modeled.
Bottom row: yaw rate (ω) exhibits typical underdamped behavior and decays to zero, and rudder input (δ_r) saturates briefly at 30 degrees before returning near neutral as the boat aligns with the reference heading.

5. Final Takeaways

All in all, the model captures the motion of the boat reasonably well. The boat's path does not reach the final wavepoint, though it is close, as suggested by the boat's path line in Fig. 1. This is likely due to the fact that the rudder input saturates and overshoots past the nominal value. To reduce the error between the desired wavepoint and the boat's final position, a different controller could be implemented in the model. The current script uses a PD controller, which is both simple to implement and yields improved stability and response. However, the controller is unable to eliminate steady-state error and would be more applicable for systems that respond to sudden disturbances or external changes. The addition of an integral control would allow the boat to reach the desired waveform, as it would completely eliminate any steady-state error. Such an alteration in control method would come with the tradeoff of increased maximum control and potential saturation of the control variable. Thus, further analysis is needed to determine the optimal combination of controller types to yield the best results.

The primary limitation of this model is that it does not include sail trim control in addition to rudder angle control due to simplifying assumption (7) highlighted above. For future applications, it may be necessary to create two separate models, one integrating rudder control and the other considering sail angle variations to understand the impact that these control systems have separately on sailboat dynamics. These two control systems are assumed to be independent of one another, meaning that sail trim control has a minimal impact on rudder angle control, and vice versa. From a physical sense, this can be explained by the fact that the sail trim is largely affected by factors within an aerodynamic system, whereas rudder angle control is dependent upon the hydrodynamic system around it. Such an analysis could validate the assumption made in the construction of this model: rudder angle variations affect sailboat motion more strongly than sail trim adjustments. Additionally, this model can be extended to consider more complicated dynamics of a sailboat. A more thorough model would mathematically describe the boat's rotational motion in other planes, other than the horizontal plane of motion that the boat interacts with translationally. Thus, the model outputs would include information about the heeling and pitching motion of the boat and more accurately describe the boat's angular position at each given time value. Greater complexity could also be added to the model by considering geometric non-uniformities of the hull, calling for passive design changes in the model. The modelled geometry of the boat is not realistic to that of a typical hull body. For more accurate results, it is necessary for the model to consider the varying cross-sectional area of a hull along its length, which, in turn, would inform the drag acting on the boat over time.

Another important consideration moving forward is the effect of environmental disturbances that were not accounted for in the simplified model. In reality, the wind would not be constant, but there would be gusts, blowing at unsteady angles, turbulence along the sail surface and water disturbances (i.e., waves). These factors could potentially impact the direction where the boat is heading and its steady state velocity. In future models, we could look forward to capturing and integrating these variables into the model to observe the extent the controller could perform to maintain stable heading and path-following performance.

6. Appendix

```
clear; clc; close all;
% Linearize nonlinear system using Jacobians
J = 0.08; % Temporary polar moment of inertia for rectangular, thin-shelled
boat
m = 200; % mass (kg)
% Define the nonlinear state equations in terms of state variables and u
syms xs ys theta v omega u
x = [xs; ys; theta; v; omega];
```

```
% Define the nonlinear state equations
f1 = v;
f2 = v*omega;
f3 = omega;
f4 = -250*v*u^2 / m - 100*v^2 / m;
f5 = -1000*v*u / J - 6000 * omega / J;
f = [f1; f2; f3; f4; f5];
% Define output equations ydot = g(x, u)
g = [theta; omega]; % Attitude/Yaw controller (rudder stabilization)
% Compute Jacobians A, B, C, D
A_sym = jacobian(f, x); % Jacobian of the state equations with respect to state
variables
B_sym = jacobian(f, u); % Jacobian of the state equations with respect to input
C_sym = jacobian(g, x); % Jacobian of the output equations with respect to
state variables
D_sym = jacobian(g, u); % Jacobian of the output equations with respect to
input
% Substitute trim/equilibrium values
x_trim = [0; 0; 0; 5; 2];   % example
u_trim = pi/6;              % initial rudder angle
A = double(subs(A_sym, [x; u], [x_trim; u_trim]));
B = double(subs(B_sym, [x; u], [x_trim; u_trim]));
C = double(subs(C_sym, [x; u], [x_trim; u_trim]));
D = double(subs(D_sym, [x; u], [x_trim; u_trim]));

% sailboat_simulation.m
%
% This script linearizes a simplified 2-D sailboat model using symbolic
% Jacobians, designs a heading controller, simulates the nonlinear dynamics
% in closed loop, plots key variables over time and animates the boat
% trajectory in the plane.  It can be adapted by changing the parameter
% values, trim conditions, controller gains and reference heading.
%
% The nonlinear model retains only the horizontal plane motion:
%   x  – inertial east position
%   y  – inertial north position
%   psi (theta) – yaw angle (heading)
%   v  – forward (surge) speed
%   omega – yaw rate
% The control input is the rudder angle delta_r (called u in the
% original symbolic model).  Disturbances such as wind are not
% explicitly modelled here; their effects are embedded implicitly in
% the damping coefficients.
function sailboat_simulation
   clear; clc; close all;
   %% Physical parameters
   J = 0.08;      % polar moment of inertia (kg*m^2) for thin-shelled boat
   m = 200;       % mass (kg)
   %% Symbolic linearization using Jacobians
   % Define symbolic variables
   syms xs ys theta v omega delta_r
   x_vec = [xs; ys; theta; v; omega];
   % Nonlinear state equations f(x,u) as defined in the report.  These
   % expressions correspond to a simplified model where surge speed v
   % drives the positions and the input delta_r influences acceleration
   % through quadratic and linear terms.
   f1 = v * cos(theta);          % x dot - forward speed projected on inertial
x
   f2 = v * sin(theta);          % y dot - forward speed projected on inertial
y
   f3 = omega;                   % psi dot - yaw rate
   f4 = sym(0);
   a_r = 0.5;    % yaw damping
   b_r = 1.0;    % rudder effectiveness
   f5 = -a_r * omega + b_r * delta_r;
   f = [f1; f2; f3; f4; f5];
   % Output equations - choose yaw angle and yaw rate as outputs
   g = [theta; omega];
   % Compute Jacobians
   A_sym = jacobian(f, x_vec);
   B_sym = jacobian(f, delta_r);
   C_sym = jacobian(g, x_vec);
   D_sym = jacobian(g, delta_r);
   % Substitute trim/equilibrium values.  Here we choose a sample trim
   % state: the boat is travelling straight (psi=0) at constant forward
   % speed v0 and yaw rate omega0.  The rudder input delta_r0 is chosen
   % such that accelerations vanish (approximate).
   v0 = 2;              % forward speed at trim (m/s)
   omega0 = 0;         % zero yaw rate at trim
   psi0 = 0;           % heading aligned with inertial x
   x_trim = [0; 0; psi0; v0; omega0];
   delta_r_trim = 0;    % trim rudder angle (rad)
   % Evaluate numeric A,B,C,D matrices
   A = double(subs(A_sym, [x_vec; delta_r], [x_trim; delta_r_trim]));
   B = double(subs(B_sym, [x_vec; delta_r], [x_trim; delta_r_trim]));
   C = double(subs(C_sym, [x_vec; delta_r], [x_trim; delta_r_trim]));
   D = double(subs(D_sym, [x_vec; delta_r], [x_trim; delta_r_trim]));
   %% Control design
   % We design a heading controller to track a reference psi_ref.  A
   % simple choice is proportional-derivative (PD) control on the
   % heading error with derivative action on yaw rate.  For example,
   Kp = 2.0;    % proportional gain on heading error
   Kd = 0.8;    % derivative gain on yaw rate
   delta_max = deg2rad(30);  % rudder saturation (±30 deg)
   %% Simulation parameters
   t_final = 40;            % simulation time (seconds)
   dt = 0.05;               % sample interval for animation
   t_span = 0:dt:t_final;   % time vector
   % Initial conditions (x, y, psi, v, omega)
   x0 = [0; 0; 0; 2; 0];    % starting at origin, heading 0, forward speed 2
m/s
   % Reference heading (step input).
   psi_ref = deg2rad(45);
   %% Run simulation using ode45
   % Define function that wraps the nonlinear dynamics and
   % control law
   dynamics = @(t, x) sailboat_dynamics(t, x, psi_ref, Kp, Kd, m, J,
delta_max);

   % Integrate
   [t_sol, x_sol] = ode45(dynamics, t_span, x0);
   % Compute control input history for plotting
   delta_hist = zeros(size(t_sol));
   for k = 1:length(t_sol)
       xs_k   = x_sol(k,1);
       ys_k   = x_sol(k,2);
       psi_k  = x_sol(k,3);
       v_k    = x_sol(k,4);
       omega_k= x_sol(k,5);
       % heading error
       epsi = wrapToPi(psi_ref - psi_k);
       % PD control law
       delta_k = Kp * epsi - Kd * omega_k;
       delta_hist(k) = max(min(delta_k, delta_max), -delta_max);
   end
   %% Plot results
   figure('Name','State Variables vs Time');
   subplot(3,2,1); plot(t_sol, x_sol(:,1), 'LineWidth',1.2); grid on;
   xlabel('Time (s)'); ylabel('x (m)'); title('Inertial x position');
   subplot(3,2,2); plot(t_sol, x_sol(:,2), 'LineWidth',1.2); grid on;
   xlabel('Time (s)'); ylabel('y (m)'); title('Inertial y position');
   subplot(3,2,3);
   plot(t_sol, rad2deg(x_sol(:,3)), 'LineWidth',1.2); hold on; grid on;
   yline(rad2deg(psi_ref),'r--','LineWidth',1.0);
   xlabel('Time (s)'); ylabel('\psi (deg)');
   title('Heading');
   legend('Actual','Reference');
   subplot(3,2,4); plot(t_sol, x_sol(:,4), 'LineWidth',1.2); grid on;
   xlabel('Time (s)'); ylabel('v (m/s)'); title('Forward speed');
   subplot(3,2,5); plot(t_sol, rad2deg(x_sol(:,5)), 'LineWidth',1.2); grid on;
   xlabel('Time (s)'); ylabel('\omega (deg/s)'); title('Yaw rate');
   subplot(3,2,6); plot(t_sol, rad2deg(delta_hist), 'LineWidth',1.2); grid on;
   xlabel('Time (s)'); ylabel('\delta_r (deg)'); title('Rudder input');
   %% Animate trajectory
   figure('Name','Sailboat Animation');
   axis equal; hold on; grid on;
   % Plot path
   plot(x_sol(:,1), x_sol(:,2), 'k-', 'LineWidth',0.8);
   % Desired waypoint for reference
   waypoint = [t_final * v0 * cos(psi_ref), t_final * v0 * sin(psi_ref)];
   plot(waypoint(1), waypoint(2), 'rx', 'LineWidth',2, 'MarkerSize',10);
   legend('Path','Desired waypoint');
   xlabel('x (m)'); ylabel('y (m)'); title('Boat trajectory and waypoint');
   % Predefine boat shape (triangle) in body frame for drawing
   L = 1.5; % length scale of boat drawing
   boatShape = L * [  1, -0.5, -0.5,  1;   % x vertices
                      0,  0.3, -0.3,  0 ]; % y vertices
   boatPatch = patch('XData',[], 'YData',[], ...
                 'FaceColor',[0 0.4470 0.7410], ...
                 'EdgeColor','k', ...
                 'HandleVisibility','off');
   legend('Path','Desired waypoint');
   % Animation loop
   for k = 1:20:length(t_sol)
       psi_k = x_sol(k,3);
       x_k   = x_sol(k,1);
       y_k   = x_sol(k,2);
       % rotate and translate boat shape
       R = [cos(psi_k), -sin(psi_k); sin(psi_k), cos(psi_k)];
       boatWorld = R * boatShape + [x_k; y_k];
       set(boatPatch, 'XData', boatWorld(1,:), 'YData', boatWorld(2,:));
       % draw now
       drawnow;
   end
   disp('Simulation complete.');
   disp('A ='); disp(A);
   disp('B ='); disp(B);
end
%% Dynamics function for ode45
function dx = sailboat_dynamics(~, x, psi_ref, Kp, Kd, m, J, delta_max)
   % Unpack state variables
   xs    = x(1);
   ys    = x(2);
   psi   = x(3);
   v     = x(4);
   omega = x(5);
   % Heading error
   epsi  = wrapToPi(psi_ref - psi);
   % PD control law on heading error and yaw rate
   delta_r = Kp * epsi - Kd * omega;
   % Saturate rudder
   delta_r = max(min(delta_r, delta_max), -delta_max);
   % Nonlinear state equations (matching those defined in the symbolic section)
   dx = zeros(5,1);
   dx(1) = v * cos(psi);                         % x dot
   dx(2) = v * sin(psi);                         % y dot
   dx(3) = omega;                              % psi dot
   dx(4) = 0;    % keep v constant
   a_r = 0.5;
   b_r = 1.0;
   dx(5) = -a_r * omega + b_r * delta_r;
End

A =
        0         0         0    1.0000         0
        0         0    2.0000         0         0
        0         0         0         0    1.0000
        0         0         0         0         0
        0         0         0         0   -0.5000

(nonlinearized jacobian matrices)
B =
    0
    0
    0
    0
    1
```

References:

[1] B. Clement, "Control algorithms for a sailboat robot with a sea experiment," in Proceedings of the 9th IFAC Conference on Control Applications in Marine Systems (CAMS), Osaka, Japan, Sept. 17–20, 2013, pp. 19–24, doi: 10.3182/20130918-4-JP-3022.00061.