

A TRIDENT SCHOLAR PROJECT REPORT

NO. 501

**Cyber-Physical System Security of Surface Ships
Using Intelligent Constraints**

by

Midshipman 1/C Daniel J. Butchko, USN



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

USNA-1531-2

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 7/12/21		2. REPORT TYPE		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Cyber-Physical System Security of Surface Ships Using Intelligent Constraints				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Butchko, Daniel J.				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Naval Academy Annapolis, MD 21402				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) Trident Scholar Report no. 501 (2021)	
12. DISTRIBUTION / AVAILABILITY STATEMENT This document has been approved for public release; its distribution is UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Cyber-physical systems are vulnerable to attacks that can cause them to reach undesirable states. This paper provides theoretical solutions for increasing the resiliency of control systems through the use of a high-authority supervisor that monitors and regulates control signals sent to the actuator. The supervisor aims to determine the control signal limits that provide maximum freedom of operation while protecting the system. Approaches for finding control signal limits include analytical, reachable set, and Monte Carlo methods. These methods are applied to linear models for autonomous surface vessels to determine the rudder constraints necessary to protect vessels from collisions in the plane of motion. Promising methods are evaluated through simulations that incorporate natural geography with coastline data and realistic traffic patterns produced from ships' automatic identification system data.					
15. SUBJECT TERMS cyber-physical system, supervisory control theory, propagation of uncertainty, predictive control; Nomoto model					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 66	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)

U.S.N.A. --- Trident Scholar project report; no. 501 (2021)

**CYBER-PHYSICAL SYSTEM SECURITY OF SURFACE SHIPS
USING INTELLIGENT CONTSTRAINTS**

by

Midshipman 1/C Daniel J. Butchko
United States Naval Academy
Annapolis, Maryland

Certification of Advisers Approval

Professor Kiriakos Kiriakidis
Weapons, Robotics, and Control Engineering Department

CDR Brien Croteau, USN
Cyber Science Department

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

Abstract

Cyber-physical systems are vulnerable to attacks that can cause them to reach undesirable states. This report provides theoretical solutions for increasing the resiliency of control systems through the use of a high-authority supervisor that monitors and regulates control signals sent to the actuator. The supervisor aims to determine the control signal limits that provide maximum freedom of operation while protecting the system. Approaches for finding control signal limits include analytical, reachable set, and Monte Carlo methods. These methods are applied to linear and nonlinear models for autonomous surface vessels to determine the rudder constraints necessary to protect vessels from collisions in the plane of motion. Promising methods are evaluated through simulations that incorporate natural geography with coastline data and realistic traffic patterns produced from ships' automatic identification system data.

Keywords— cyber-physical system; supervisory control theory; propagation of uncertainty; predictive control; Nomoto model

Contents

1	Introduction	4
1.1	Motivation	4
1.2	Contribution	5
1.3	Related Works	5
2	Background	7
2.1	Linear Time-Invariant Systems	7
2.2	Cyber Attack's Effect on the System	9
3	Approaches to Artificial Control Signal Limits	9
3.1	Defending against Cyber Attacks: the Concept of the Supervisor	9
3.2	Limits Calculated Analytically	12
3.2.1	Steady State Limits	12
3.2.2	Transient Limits	13
3.3	Limits Based on a Reachable Set	13
3.3.1	Assumptions	13
3.3.2	Approximating the Reachable Set	14
3.3.3	Determining Control Signal Limits	16
3.4	Limits from Monte Carlo Simulation	17
3.4.1	Random Control Signals	17
3.4.2	Trajectories in the State Space	18
3.4.3	Determining Control Signal Limits	19
3.5	An Illustrative Example	19
3.5.1	Problem Statement	21

	3
3.5.2 Analytical Approach	23
3.5.3 Reachable Set Approach	26
3.5.4 Monte Carlo Approach	27
4 Ship Application	29
4.1 Autonomous Surface Ships	29
4.2 Nomoto Model	29
4.3 Autopilot Controller	32
4.4 Analytical Approach	33
4.4.1 Steady State Limits	33
4.4.2 Transient Limits	34
4.5 Reachable Set Approach	37
4.5.1 Attack Model	37
4.5.2 Determining Control Signal Limits	38
4.6 Monte Carlo Approach	40
5 Synthetic Channel Simulation	42
5.1 Problem Statement	42
5.2 Solution	42
5.3 Results	44
6 San Diego Bay Channel Simulation	47
6.1 Coastline Data	47
6.2 Automatic Identification System Tracks	48
6.3 Advanced Features of the Supervisor	50
6.4 Problem Statement	52

6.5	Results	52
6.6	Kalman Filter	54
7	Conclusion	56
A	Appendix: San Diego Bay Channel Simulation Code	60

1 Introduction

1.1 Motivation

Cyber-physical systems (CPSs) are digital and mechanical devices that use sensors and actuators to detect and change their state respectively. Some examples of CPSs include ships [1], aircraft, submarines, spacecraft, automobiles, pipelines [2], railways [3], power plants, water plants [4], smart grids [5], and modern medical devices [6]. Cyber-physical systems often govern one or more automated or automatic processes that attempt to achieve a desired state through feedback loops with sensors and actuators. For example, a car's cruise control system drives a vehicle to a desired speed. In the same way, a nuclear power plant drives a nuclear reactor to a desired temperature. Cyber-physical systems rely on the accuracy and precision of sensors and actuators to reach a desired state. All digital and mechanical systems are prone to malfunctions or errors. Additionally, cyber-physical systems are vulnerable to deliberate attacks on both their cyber and physical components. Errors due to malfunctions or deliberate attacks on a CPS can cause the system to undergo an undesired, or possibly dangerous response. For example, in 2008, a software update to the Hatch nuclear power plant's computer reset the control system's data, convincing the system that it had an insufficient water supply for cooling the nuclear fuel rods [7]. The error triggered an automated safety system that shut down the plant for 48 hours. As for deliberate cyber attacks, in 2015 hackers demonstrated an exploitation of a Jeep Cherokee vehicle in which they overrode signals to the climate control system, brakes, steering,

transmission, and entertainment system while miles away from the vehicle [8].

Consider another example of a deliberate cyber attack. A ship's autopilot navigates between waypoints defined by GPS coordinates. An adversary spoofs the GPS signal being sent to the ship and tricks the autopilot into believing that the ship is located somewhere other than its actual location [9]. As a result, the autopilot directs the ship off course, risking a collision or running the ship aground; a sensor error produces an undesired response. However, errors are not just limited to eliciting undesired responses on two dimensional maps. A ship could reach an undesired roll, yaw, or pitch. For instance, in January of 2015, the *Heogh Osaka*, a car carrying cargo ship, reached a dangerous 40 degree starboard list after making too tight a turn out of port with improperly loaded ballast tanks [10]. Essentially, undesired responses can occur in any one of the degrees of freedom of ship motion. This report aims to prevent undesired responses in cyber-physical systems, specifically surface ships, in the event of a malfunction or cyber-attack.

1.2 Contribution

This report offers the following primary contributions:

1. Explores three options for determining control signal limits.
2. Employs a statistics-based approach for determining control signal limits that maximizes freedom of operation while subject to safety constraints.
3. Demonstrates, through simulation, the application of a high-authority supervisor for the rapid calculation and employment of control signal limits.

1.3 Related Works

Much of the research in the area of CPS security places an emphasis on the prevention and detection of cyber attacks [11] rather than outlining how a CPS should behave after its defenses have been penetrated. The topic

of survivability is an area of research that has yet to be fully explored. Survivability is defined as a CPS's ability to endure an attack and reach its goals at the same time [12]. Similarly, research in fault tolerant control (FTC) focuses on maintaining system stability and operation when failures occur [13]. A group of researchers at Purdue University focused their work on the safe operation of unmanned aircraft systems (UASs) under the influence of undetected cyber attacks. They developed a successful safety assessment algorithm for UASs using knowledge of an aircraft's current state and its flight dynamics [14].

This work builds upon the theories developed in the field of Kinodynamics [15] which seeks to determine motion-control inputs subject to dynamic constraints governed by differential equations of motion and kinematic constraints, such as obstacles, including the sampling-based algorithms of rapidly exploring random trees (RRTs) [16] and probabilistic roadmaps (PRM) [17].

Additionally, the concept of using a supervisor agent was inspired by research in supervisory control of discrete-event systems [18]. In this work, the discrete events that requires re-planning includes a change in the anticipated attack magnitude and the discovery of new obstacles along the planned trajectory.

Specifically, the concept of using actuator limits [19] has been extended in this work to dynamically determine safe limits for a limited time horizon on a periodic basis.

Reachable sets approximated from linear matrix inequalities have been employed to demonstrate the effects of sensor attacks on CPSs [20]. Computational costs are a challenge that come with calculating reachable sets, so a computationally efficient processes [21] is a key motivation for this research.

In regards to practical applications, maritime autonomous surface ships have a strong need for protection against attacks due to their heavy reliance on motion controllers [22] and the Automatic Identification System [23] (AIS) for navigation. Previous work by the authors presents an approach for protecting surface ships that relies on redundant actuators that allow for alternative actuation paths when the primary actuation path has been compromised by an attack [24].

2 Background

2.1 Linear Time-Invariant Systems

Physical systems, often called plants, can be represented by simplified mathematical models for the purpose of simulating the plant's behavior or designing a control system that drives the plant to a desired state. Models are often defined by a set of linear or nonlinear differential equations. In the Laplace, or frequency domain, the dynamics are represented with transfer function (1), where $U(s)$ is the input, and $Y(s)$ is the output. One can determine the steady state gain of any stable transfer function by substituting zero for s .

$$G(s) = \frac{Y(s)}{U(s)} \quad (1)$$

Transfer functions allow for dynamic systems to be modeled in the frequency domain. Linear dynamics can also be represented in matrix form in a state space. Linear Gaussian time-invariant continuous state spaces take the form of (2) and (3), in which vector $\mathbf{x}(t)$ contains the system's states, vector $\mathbf{u}(t)$ contains the system's inputs, vector $\mathbf{y}(t)$ contains the system's outputs, $\mathbf{w}(t)$ is the process noise, $\mathbf{v}(t)$ is the measurement noise, and matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are the state, input, output, feedforward, and noise matrices respectively.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \quad (2)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{v}(t) \quad (3)$$

When working with computers and digital controllers that perform calculations at time interval, T , it is necessary to convert (2) and (3) to the discrete form in (4) and (5) where $t = kT$, t_k is the time at step k , and t_{k+1} is the time at step $k + 1$.

$$\mathbf{x}(t_{k+1}) = \mathbf{\Phi}\mathbf{x}(t_k) + \mathbf{\Gamma}\mathbf{u}(t_k) + T\mathbf{w}(t_k) \quad (4)$$

$$\mathbf{y}(t_k) = \mathbf{C}\mathbf{x}(t_k) + \mathbf{v}(t_k) \quad (5)$$

Matrices $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are approximated with (6) and (7) respectively.

$$\mathbf{\Phi} = e^{\mathbf{A}T} \approx \mathbf{I} + \mathbf{A}T \quad (6)$$

$$\mathbf{\Gamma} = \int_0^T e^{\mathbf{A}\tau} \mathbf{B} d\tau \approx \mathbf{B}T \quad (7)$$

Physical plants, by themselves, operate as open loop systems. A plant requires a controller to drive a state to a desired value. Feedback loops, like the one shown in Fig. 1, use an actuator to correct the state based on feedback from a sensor.

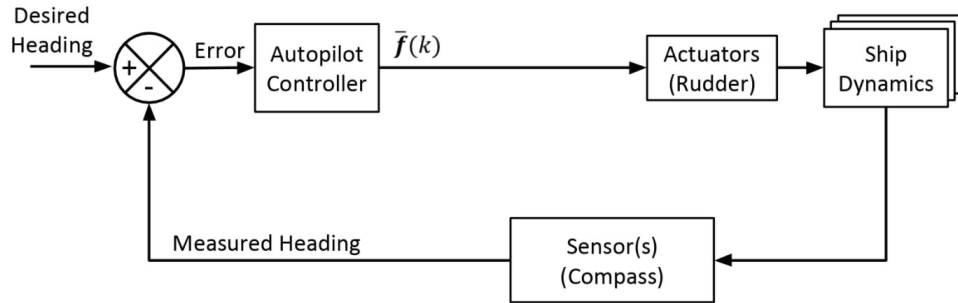


Figure 1: Control loop for a ship's autopilot system where $\bar{f}(k)$ is the control signal sent from the controller to the actuator. The plant or model is defined by the "Ship Dynamics" component.

2.2 Cyber Attack's Effect on the System

It is assumed that the adversary has the ability to directly or indirectly affect the value of the control signal sent to the actuator. The adversary can attack the system at any one of the components in the feedback loop, denoted in Fig. 1. In effect, the attack is propagated through the system and modifies the value of the control signal in some random manner. The effect can be modeled as overwriting the original control signal with a random value. The attacker may have some degree of control over the duration for which each control signal is held constant, as well as the duration of the attack period.

3 Approaches to Artificial Control Signal Limits

3.1 Defending against Cyber Attacks: the Concept of the Supervisor

Actuators have natural hard limits that prevent them from driving a system to extreme states. For example, most rudders on ships have a hard limit of 35 degrees which prevents the ship from rotating faster than a maximum yaw rate. However, there are times when it would be useful to impose artificial, or soft limits, on actuators to prevent the system from reaching undesired states. In theory, if limits can be defined on the system's response in the physical domain, then limits can be calculated for the actuator's state using what is known about the plant dynamics. Since the actuator's state is proportional to the control signal, then the limits to be imposed on the control signal can be calculated. In this report, for the sake of simplicity, it is assumed that the control signal is equivalent to the actuator's state. The supervisor only constrains control signals that risk the safety of the system; the supervisor does not interfere with corrupt, but safe control signals.

In a typical control loop, like the one shown in Fig. 1, the control signal, $\bar{f}(k)$, is passed directly from the controller to the actuator. This research proposes bounding the control signal with a device called the

supervisor. In Fig. 2, the unbounded control signal, $\bar{f}(k)$, is passed from the controller to the supervisor. The supervisor constrains the unbounded control signal and passes off the bounded control signal, $\bar{u}(k)$, from the supervisor to the actuator.

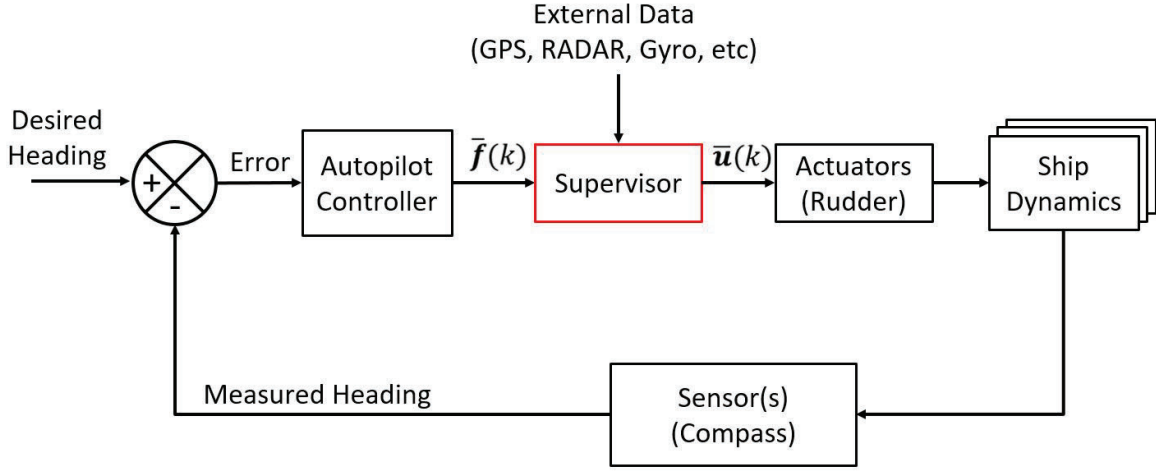


Figure 2: Control loop for a ship's autopilot system with the supervisor.

The supervisor's function can be defined mathematically in the form of piecewise function (8). The upper and lower limits placed on the unbounded control signal are given by $\bar{\alpha}(k)$ and $\bar{\beta}(k)$ respectively. Function (8) shows the example for when only absolute upper and lower limits are placed on the unbounded control signal. It possible to also have local upper and lower limits within the absolute limits.

$$\bar{u}(k) = \begin{cases} \bar{\alpha}(k) & \bar{f}(k) > \bar{\alpha}(k) \\ \bar{f}(k) & \bar{\beta}(k) \leq \bar{f}(k) \leq \bar{\alpha}(k) \\ \bar{\beta}(k) & \bar{f}(k) < \bar{\beta}(k) \end{cases} \quad (8)$$

The supervisor must accomplish two specific functions. It must first determine the limits to be placed on the system's response in the state space domain. Second, it must calculate the corresponding actuator limits and control signal limits. The second function, calculating actuator limits and control signal limits, is the focus of this research, but it is also important to understand how the supervisor determines state space limits. Essentially, the supervisor imposes limits on state variables based on environmental conditions. It is

assumed that the supervisor has access to a variety of sensors and data that describe the state of the system and its surrounding environment. From those observations, the supervisor can determine what states in the state space are safe or unsafe. The boundary between the safe and unsafe states is the limit to be placed on the system's response.

In the event of a cyber-attack, one might question why a ship or CPS should be able to trust GPS sensors or radar, especially when GPS spoofing is such a realistic threat. It is assumed that the supervisor and its sensors cannot be attacked directly. In a real-world implementation, the supervisor would be considered a third party device that is added on to a CPS after its production. The GPS, radar, and other sensors that the supervisor uses to collect data are internal to the supervisor's hardware package and are separate from the ship's built-in sensors. The supervisor would be the product of a trusted manufacturing process that negates any risks of back doors or built-in vulnerabilities. Additionally, since the supervisor is added to the CPS after the CPS has been produced, it can be assumed that the adversary is not aware that the supervisor exists.

The supervisor can be used to constrain a system's ability to reach certain states for up to an indefinite amount of time when the system is stable, but only up to a finite time when working with an unstable system. For an unstable system, even if a ship's rudder is constrained to a magnitude of 5 degrees in either direction, the ship will still be capable of reaching dangerous states, but it will just take a longer time than if the rudder was only constrained to its hard limits. For this reason, it is necessary to constrain time within some relevant finite bound. For example, it may only be necessary for the supervisor to calculate limits for up to two minutes into the future. Whatever the time horizon, it should provide enough time for the system to return to its initial state. In the case of a ship driving dangerously towards an obstacle, the supervisor's time horizon must be large enough for the supervisor to force a turn and avoid a collision.

3.2 Limits Calculated Analytically

A system of equations govern the relationship between a plant's input and an output. The analytical approach aims to invert the system of equations in order to solve for the plant input required to achieve a given output. The plant's input limits can then be determined from constraints on the output.

3.2.1 Steady State Limits

First, one can calculate a limit on the steady state value of the response with the prerequisite that the system is stable. A system defined by transfer function (1) is considered, where $U(s)$ is a step input with magnitude M .

$$U(s) = \frac{M}{s} \quad (9)$$

The steady state output, $Y(\infty)$, can be constrained by upper and lower constant limits Y_l and Y_u respectively, such that

$$Y_l \leq Y(\infty) \leq Y_u \quad (10)$$

From control theory, (11) is known to hold true.

$$Y(\infty) = G(0)M \quad (11)$$

Therefore, the magnitude of the step input should be bounded by an upper and lower limit to keep the steady state response between Y_l and Y_u .

$$Y_l G^{-1}(0) \leq M \leq Y_u G^{-1}(0) \quad (12)$$

3.2.2 Transient Limits

Alternatively, constant limits may be found to constrain output $Y(s)$ throughout the transient and steady state responses. A step input, $U(s) = \frac{M}{s}$, can be assumed for transfer function (1), which allows for expansion through partial fractions, followed by the inverse Laplace Transform shown in (13) [25]. The resulting equation can be rearranged to solve for limit M given a value for y at time t .

$$y(t) = \mathcal{L}^{-1} \left\{ \frac{M}{s} Y(s) \right\} \quad (13)$$

3.3 Limits Based on a Reachable Set

3.3.1 Assumptions

For the reachable set approach, it is assumed that a cyber attack modifies the signal to the actuator by overwriting the signal in an unpredictable manner. Consequently, this injected signal can be modeled as a random variable. It is useful to assume this injected signal follows a normal distribution so it can be used in (17). The assumption that this injected signal is sampled from a Gaussian distribution is reasonable when one considers the multiple layers an attack would have to travel through before its effects are felt by the actuator. Assuming a Gaussian distribution means the injected signal, in theory, has infinite bounds. Regardless of the control signal's magnitude, the physical system is constrained by the natural physical limitations of its actuators.

In this research, it is assumed that the system can be modeled with continuous linear state equations with matrices (A, B) . It is also assumed that one or more of the system's states require restraint.

3.3.2 Approximating the Reachable Set

The reachable set defines the region of a state space that a system is capable of reaching based on its current state and actuation capabilities. The reachable set can be constrained when desired through the application of actuator limits. The method explored in this section requires first approximating the reachable set from control signal limits before determining the control signal limits necessary to produce a reachable set that avoids unsafe regions in the state space.

Since it is assumed that the control signal injected by the cyber attack is sampled from a normal distribution, the state of the system is not definite. A Markov process is used to propagate the state covariance matrix through time based on the uncertainty of the control [26]. The attacking control signal is a normally distributed random variable given by $\mathcal{N}(\mu_u, \sigma_u)$, where μ_u is the baseline control signal and σ_u is the standard deviation.

The effect of the current state on the future state of the system is given by matrix Φ , the state transition matrix. The state transition matrix can be approximated with (6) when T , the sample time, is small. The effect of the control signal on the future state of the system is dependent on Γ , the input transition matrix, approximated with (7).

Matrix P is the covariance matrix of the states and contains the state variances along its diagonal, as shown in (14). The example matrix in (14) is for a 2×2 state equation with a 2×1 input matrix. The standard deviations describing the uncertainty of each state variable are given by σ_{x_1} and σ_{x_2} respectively.

$$P = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1}\sigma_{x_2} \\ \sigma_{x_1}\sigma_{x_2} & \sigma_{x_2}^2 \end{bmatrix} \quad (14)$$

Matrix Q is the covariance matrix of the disturbance vector, and contains input variances along its di-

agonal. This report assumes a system with only one input, so $\mathbf{Q}$ is a 1×1 matrix as shown in (15).

$$\mathbf{Q} = \begin{bmatrix} \sigma_u^2 \end{bmatrix} \quad (15)$$

The states are propagated through time with (16), where $\mathbf{x}_k$ is the state vector and $\mathbf{u}_k$ is the mean input vector containing the control signal's expected value at step k .

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k \quad (16)$$

The state covariance matrix is propagated through time with (17), where $\mathbf{P}_k$ denotes the state covariance matrix at step k .

$$\mathbf{P}_{k+1} = \mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T + \mathbf{\Gamma}\mathbf{Q}\mathbf{\Gamma}^T \quad (17)$$

From the state covariance matrix, the standard deviation associated with each state can be extracted and used to produce a multidimensional confidence interval portrayed graphically as an ellipsoid in the state space. The ellipsoid is an approximation of the reachable set. The probability that the state of the system is bounded by the ellipsoid at step k is dependent on the Mahalanobis distance, m . In the two dimensional case, the ellipse's geometry is defined by a horizontal radius, vertical radius, and angle given by $m\sqrt{\lambda_1}$, $m\sqrt{\lambda_2}$, and θ respectively [27]. The state covariance matrix's eigenvalues, λ_1 and λ_2 , are given by (18) and (19) respectively.

$$\lambda_1 = \frac{\sigma_{x_1}^2 + \sigma_{x_2}^2}{2} + \sqrt{\left(\frac{\sigma_{x_1}^2 - \sigma_{x_2}^2}{2}\right)^2 + (\sigma_{x_1}\sigma_{x_2})^2} \quad (18)$$

$$\lambda_2 = \frac{\sigma_{x_1}^2 + \sigma_{x_2}^2}{2} - \sqrt{\left(\frac{\sigma_{x_1}^2 - \sigma_{x_2}^2}{2}\right)^2 + (\sigma_{x_1}\sigma_{x_2})^2} \quad (19)$$

The ellipse's angle is given by θ in (20).

$$\theta = \begin{cases} 0 & \text{if } \sigma_{x_2}\sigma_{x_1} = 0 \text{ and } \lambda_1 \geq \lambda_2 \\ \frac{\pi}{2} & \text{if } \sigma_{x_2}\sigma_{x_1} = 0 \text{ and } \lambda_1 < \lambda_2 \\ \tan^{-1} \left(\frac{\lambda_1 - \sigma_{x_1}^2}{\sigma_{x_2}^2} \right) & \text{else} \end{cases} \quad (20)$$

The coordinates that shape the confidence ellipse in the state space at any step, k , can be solved for using parametric equations (21) and (22) where $0 \leq t \leq 2\pi$ radians.

$$e_{1,k}(t) = m\sqrt{\lambda_1} \cos \theta \cos t - m\sqrt{\lambda_2} \sin \theta \sin t + x_{1,k} \quad (21)$$

$$e_{2,k}(t) = m\sqrt{\lambda_1} \sin \theta \cos t + m\sqrt{\lambda_2} \cos \theta \sin t + x_{2,k} \quad (22)$$

When $m = 2$, the confidence ellipse is expected to contain 95% of the reachable set. This approximation is used to determine the appropriate control signal limits.

The ellipse will grow infinitely in size with time. Therefore, it is necessary to implement a time horizon: the time at which the final reachable set is approximated.

3.3.3 Determining Control Signal Limits

It is often desirable to constrain the reachable set in the state space to avoid certain combinations of states. The states to be avoided are denoted as the danger area in Fig. 3.

With the constraint that the reachable set must not overlap the danger area, maximizing the ellipse's size provides the system with the greatest functionality. The original control signal's standard deviation, σ_u , is directly proportional to ellipse size, so maximizing σ_u maximizes system functionality.

To determine the maximum control signal subject to danger area constraints, an iterative process is followed in which the value of σ_u is decreased until an ellipse is produced that is tangential to the danger area, such as the ellipse in Fig. 3. Assuming a 95% confidence interval, the control signal limits are set at $\pm 2\sigma_u$.

A system's states can be bounded by an ellipse at a certain point in time, but not necessarily for all of time. Systems can carry momentum that pushes states beyond their instantaneous limits. In other words, a trajectory in the state space may fall inside the ellipse at the time horizon, but run outside the ellipse bounds and into the danger area after the time horizon. This issue is addressed in the example of ship motion.

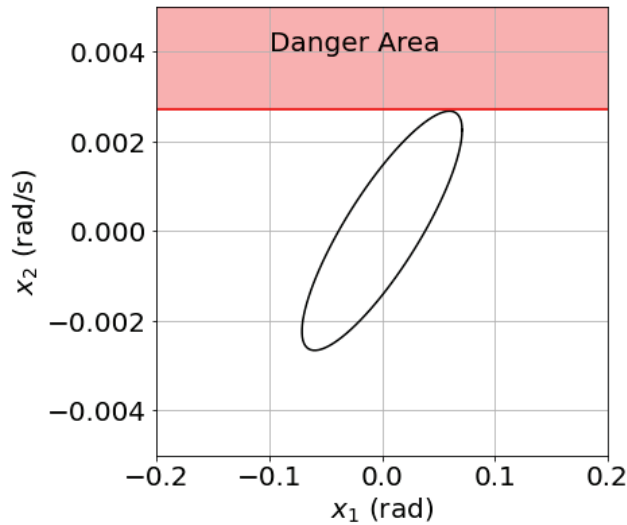


Figure 3: A 95% prediction ellipse that is tangential to the danger area in the state space approximates the reachable states.

3.4 Limits from Monte Carlo Simulation

3.4.1 Random Control Signals

To determine control signal limits through Monte Carlo simulations, control signals are randomly sampled from a pseudo-population to produce trajectories in the state space. The method of random sampling replicates the anticipated method of cyber attack. For the purposes of this report, it is assumed that cyber attacks overwrite existing control signals, but the Monte Carlo approach can operate outside of that assumption if desired. The Monte Carlo approach can be used to protect against a variety of cyber attack formats because it essentially simulates many cyber attacks to determine which control signals are safe or unsafe.

Similar to the reachable set approach, control signals can be randomly sampled from a Gaussian distri-

bution, but other distributions are also possible. A normal distribution may be desirable if each control signal has the probability of occurring in the cyber attack. A random walk is another method that can be used to replicate a cyber attack for the Monte Carlo approach. A Monte Carlo simulation following a random walk steps the control signal up, down, or holds it constant at each step in time. A random walk method may be desirable when the control signal's rate of change is limited by a constant value.

3.4.2 Trajectories in the State Space

An example of a Monte Carlo simulation is demonstrated with the random walk method. Starting at the origin, the control signal steps up or down by 5 degrees at each sample period. Each sequence of control signals is applied to the system and produces a trajectory in the state space. The trajectory shows the combination of states resulting from each sequence of control signals. Figure 4 shows a collection of these trajectories in the state space.

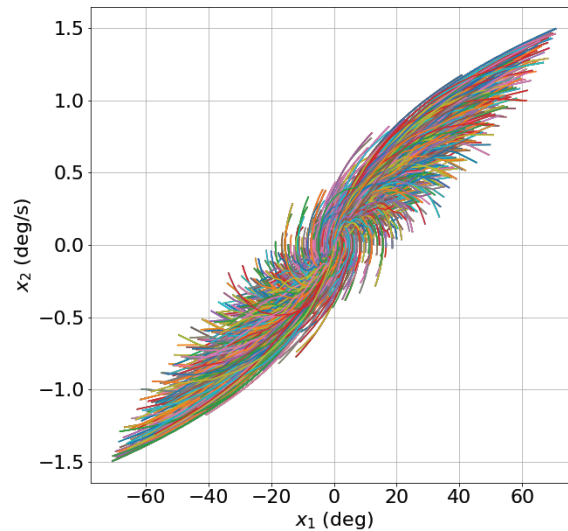


Figure 4: 10,000 state space trajectories resulting from a random walk.

3.4.3 Determining Control Signal Limits

The state space may be bounded by danger areas that define state combinations to be avoided. Trajectories that overlap danger areas are classified as unsafe along with their corresponding sequence of control signals. The supervisor intercepts control signal sequences sent from the controller to the actuator, saturates those that are known to produce unsafe trajectories, and accepts those that produce safe trajectories.

3.5 An Illustrative Example

The three theories proposed for calculating actuator limits are first be applied to the simple kinematics of an unmanned underwater vehicle (UUV) model with two degrees of freedom before they are applied to ship dynamics in the plane of motion. The UUV model offers simplicity in that it has two inputs that are decoupled, and the variables to be constrained in the plane of motion are the UUV model's state variables. The UUV model's dynamics are inspired by the motion of the UUV shown in Fig. 5. With all three methods of constraining limits, the goal is to define unsafe regions in the plane of motion to be avoided by the UUV and determine the control signal limits required to avoid those regions while maximizing the UUV's freedom of maneuverability.

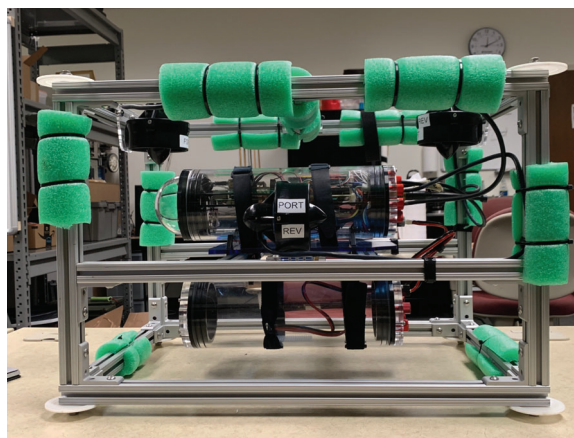


Figure 5: The physical UUV used as inspiration for the UUV model.

Differential equations (23) and (24) describe the dynamics of the UUV model. State x_1 is the position on

the longitudinal axis in meters, and state x_2 is the position on the vertical axis in meters. Constant $a = 0.1$ is the drag coefficient in s^{-1} , and $b = 1$ is the conversion factor from force to velocity in units of $\frac{s}{kg}$

$$\dot{x}_1(t) = bu_1(t) \quad (23)$$

$$\dot{x}_2(t) = -ax_2(t) + bu_2(t) \quad (24)$$

To generate an understanding of the UUV model's dynamics, sample trajectories are provided in Fig. 6 and Fig. 7 for x_1 and x_2 , where a unit step input is applied to both u_1 and u_2 simultaneously.

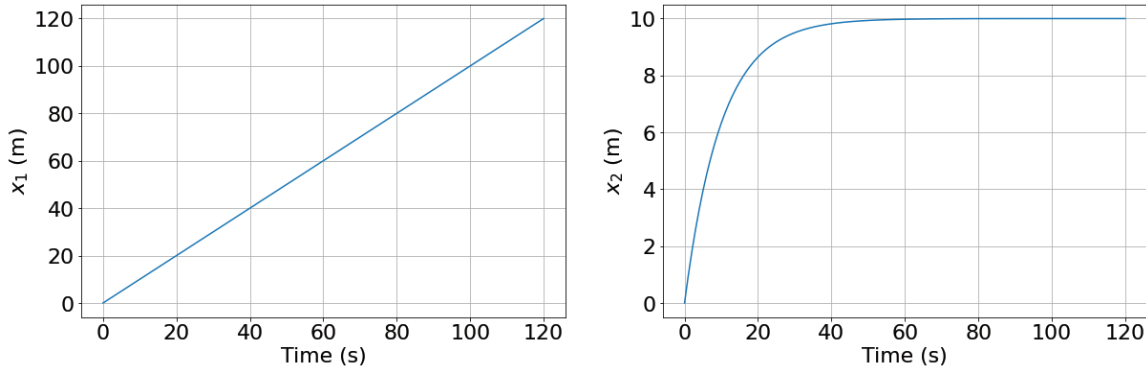


Figure 6: The unit step response for each state variable.

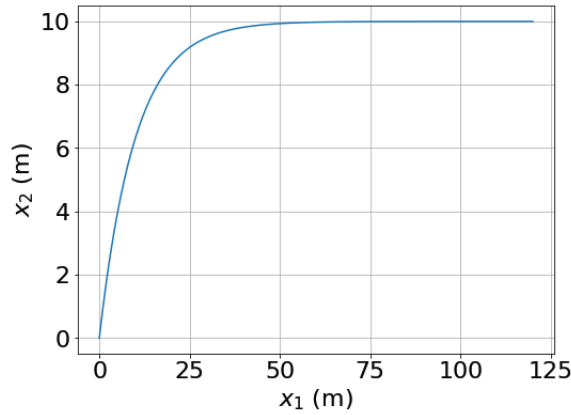


Figure 7: The state space trajectory resulting from a unit step input applied to both u_1 and u_2 .

3.5.1 Problem Statement

The UUV uses sonar sensors to detect obstacles and make decisions as it navigates through the water. After the UUV takes a sensor reading, its control system calculates the series of inputs necessary to reach a desired state. In this scenario, the desired state is the blue star in Fig. 8. A cyber attack introduces error to the sonar reading and shifts the UUV's perspective of its environment. To the UUV, the desired state, the blue star, appears where the red star is shown. The UUV's autopilot is tasked with navigating to the red star, but it also attempts to plan a safe trajectory for after it passes the red star. The UUV will be forced to pass through a small opening in a large obstruction, represented by the gap in the vertical black line. With safety as a hard constraint, the UUV's navigation and control system must find the combination of inputs that allow the UUV to avoid the obstruction, but maximize its performance, by bringing the UUV as close as possible to the desired state.

The navigation and control system is tasked with solving for the optimal combination of inputs for u_1 and u_2 that generate a trajectory passing from the origin through the opening, and minimizing the euclidean distance between the red star and it's nearest point along the trajectory. This optimization problem has three decision variables, M_{21} , M_{22} , and T_1 , that relate in piece-wise function (25). Given any two decision variables, a feasible region can be found for the third variable. With the feasible region as a constraint, the third variable can be solved for as the value that minimizes the distance between the red star and its nearest point along the trajectory.

$$u_2(t) = \begin{cases} M_{21} & T_0 \leq t \leq T_1 \\ M_{22} & T_1 \leq t \leq T_2 \end{cases} \quad (25)$$

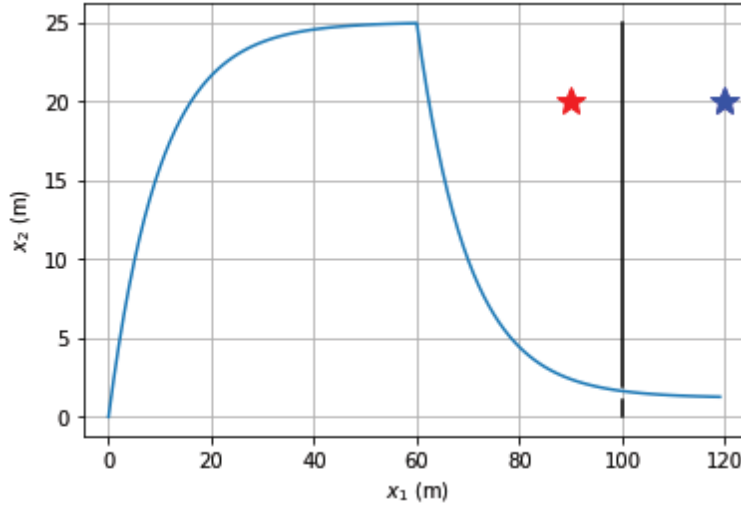


Figure 8: Setup of the UUV problem with a feasible trajectory plotted.

In the following example, the red star is located at position $s_1 = 90$ m and $s_2 = 20$ m. The simulation start and end times are $T_0 = 0$ s and $T_2 = 120$ s respectively. The magnitude of input u_1 remains constant at $M_1 = 1$ N for the duration of the simulation. The initial magnitude for input u_2 is $M_{21} = 2.5$ N. The obstruction is located at a horizontal distance of $w_h = 100$ m from the origin. The top and bottom points of the opening are positioned at respective distances of $w_T = 2$ m and $w_B = 1$ m from the horizontal axis. It is assumed that input u_2 changes from magnitude M_{21} to M_{22} at time $T_1 = 60$ s, and the UUV passes through the opening at time T_w . In this example problem, the objective is to find the value for M_{22} in newtons that generates a trajectory through the opening, and minimizes the distance between the red star and its nearest point on the trajectory.

$$\begin{aligned}
\min_{M_{22}} \quad & d = \sqrt{(x_1 - s_1)^2 + (x_2 - s_2)^2} \\
\text{s.t.} \quad & \dot{x}_1(t) = bu_1(t) \\
& \dot{x}_2(t) = -ax_2(t) + bu_2(t) \\
& x_2(T_w) \leq 2 \\
& x_2(T_w) \geq 1 \\
& x_1(T_w) = w_h
\end{aligned} \tag{26}$$

3.5.2 Analytical Approach

The analytical approach aims to solve for feasible values for the decision variables through analytical means. Continuing with the example problem, the feasible range of values for M_{22} can be solved for through analytical means using (27), (28), and the bottom three constraints from (26).

$$x_1(t) = bM_1t + x_1(T_0) \tag{27}$$

$$x_2(t) = \begin{cases} \frac{bM_{21}}{a} (1 - e^{-at}) + x_2(T_0)e^{-at} & , T_0 \leq t \leq T_1 \\ \frac{bM_{22}}{a} (1 - e^{-aT_1}) + (\frac{bM_{21}}{a} (1 - e^{-aT_1}) + x_2(T_0)e^{-aT_1})e^{-a(t-T_1)} & , T_1 \leq t \leq T_2 \end{cases} \tag{28}$$

The feasible region for M_{22} is given as interval (29), and shows the two boundary conditions in Fig. 9.

$$0.055 \, N \leq M_{22} \leq 0.157 \, N \tag{29}$$

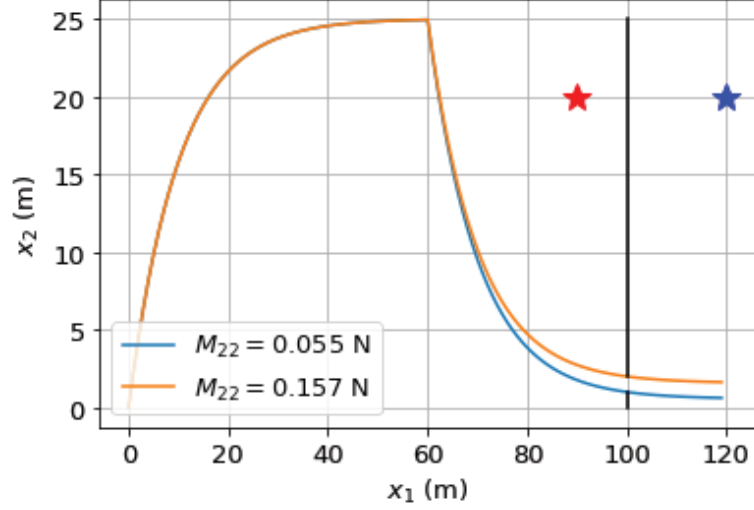


Figure 9: The feasible region for M_{22} displayed graphically on the state space. The two extreme trajectories are shown.

The objective is to minimize the distance, d , between the red star and the nearest point along the UAV's trajectory. The euclidean distance is provided in (30) as d .

$$d = \sqrt{(x_1 - s_1)^2 + (x_2 - s_2)^2} \quad (30)$$

Solving the optimization problem requires setting the derivative of d with respect to t equal to zero and solving for the time, t_m , at which the minimum d occurs. The objective function can be simplified as z , given by (31), which is proportional to d .

$$z = \frac{1}{2}((x_1 - s_1)^2 + (x_2 - s_2)^2) \quad (31)$$

Unfortunately, t_m cannot be solved for analytically due to the fact that $\frac{dz}{dt} = 0$ is a transcendental equation, containing both an exponential and a linear term with t_m in it. Instead, t_m is solved for graphically from inspection of Fig. 10.

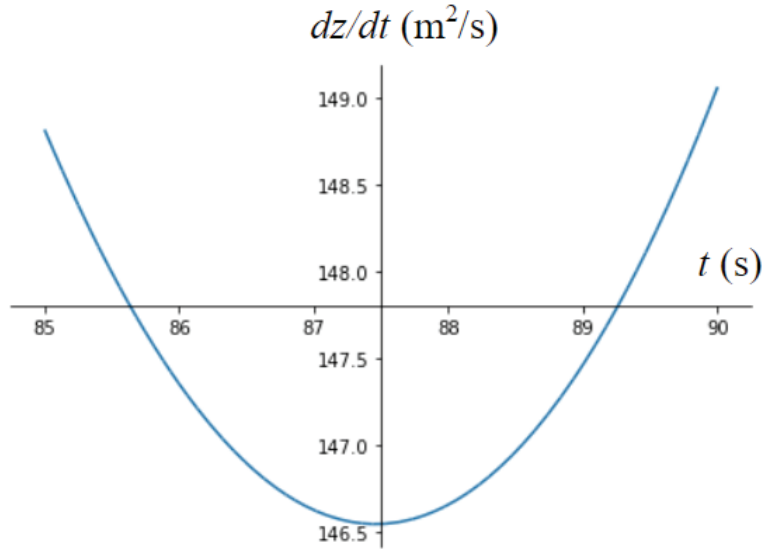


Figure 10: Graph of the function $\frac{dz}{dt}$ with $M_1 = 1$ N, $M_{21} = 2.5$ N, and $T_1 = 60$ seconds. From inspection, $t_m = 87.4$ seconds.

With a time for which the minimum distance between the trajectory and the red star occurs, the value of M_{22} that minimizes z graphically can be found from Fig. 11.

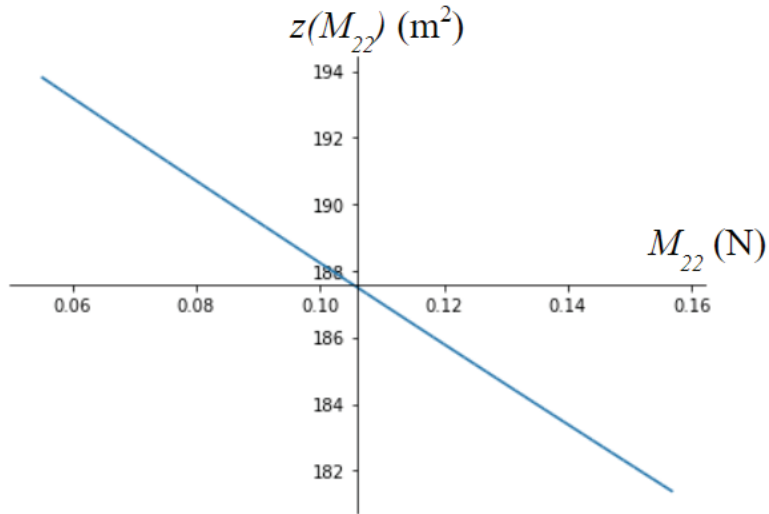


Figure 11: Graph of the function $z(M_{22})$ with $M_1 = 1$ N, $M_{21} = 2.5$ N, and $T_1 = 60$ seconds. The domain is the feasible region for M_{22} . As expected, z is minimized when $M_{22} = 0.157$ N.

3.5.3 Reachable Set Approach

For the reachable set approach, a more general example of the problem is considered in which the UUV must again pass through a small opening, and avoid colliding with the obstruction regardless of what direction the control system tells the UUV's actuators to travel. The reachable set approach relies on the assumption that the UUV's inputs are sampled from a Gaussian distribution.

The reachable set method bounds the UUV's state with a 95% confidence ellipse that approximates the UUV's reachable set. The reachable set is the region the UUV is capable of reaching after a specified time frame with given input limits. To avoid colliding with the obstruction as the UUV passes through the opening, the input limits are adjusted to shrink the confidence ellipse, and therefore the UUV's reachable set. To constrain the control signal, σ_{u_2} , the standard deviation associated with u_2 , decreases. The ellipse size is determined by the covariance matrix $\mathbf{P}$, given by (17), for each step, k , along the trajectory. Figure 12 shows an example in which this procedure is used to maintain safety of the UUV while passing through an opening.

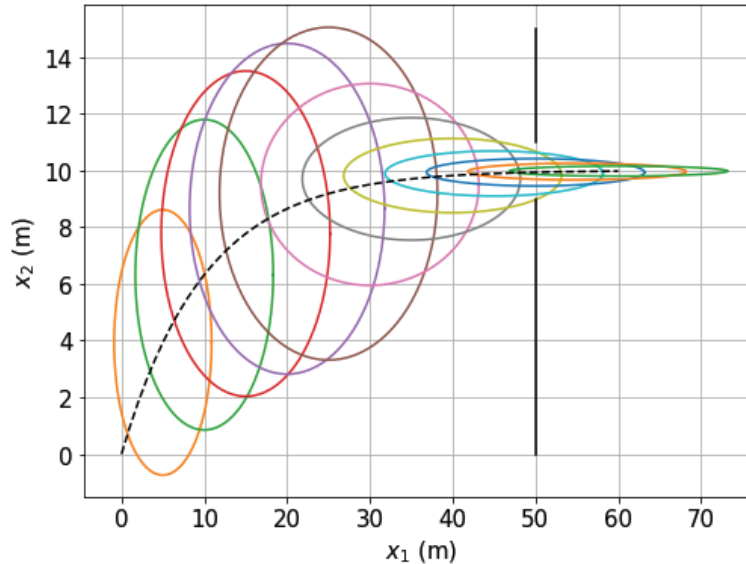


Figure 12: An ellipse describes the approximate reachable set along the trajectory for each step, k . Only every 5th ellipse is shown in the plot for clarity.

3.5.4 Monte Carlo Approach

The Monte Carlo simulation is also considered for solving the problem of the UUV passing through the small opening. Once again, the aim is to find the combination of decision variables, (M_{21}, M_{22}, T_1) that produce the trajectory with the minimum distance between the red star and its nearest point on the trajectory, while still passing through the opening.

A pseudo-population is defined for each decision variable based on the hard limits associated with each variable. It is assumed that $0\ N \leq M_{21} \leq 3\ N$, $0\ N \leq M_{22} \leq 3\ N$, and $0\ s \leq T_1 \leq 120\ s$. Random combinations of the decision variables are selected and simulated to produce trajectories. Those trajectories that reach danger areas in the state space by intersecting with the wall are filtered out. Only the feasible trajectories remain, and are shown in Fig. 13. The decision variable combinations that produce the feasible trajectories are shown in Fig. 14. The Monte Carlo simulation is complete after finding the feasible trajectories.

The next part of the problem is to find the optimal trajectory based on the constraints. To find which of the feasible trajectories reaches nearest the red star, a brute force algorithm is used to find the minimum distance between the red star and each trajectory. Comparing the minimum distance for each trajectory then allows for finding the trajectory with the minimum distance to the star. In the case of this example problem, the combination of decision variables that produced the optimal trajectory is $(M_{21}, M_{22}, T_1) = (1.504\ N, 0.011\ N, 79\ s)$

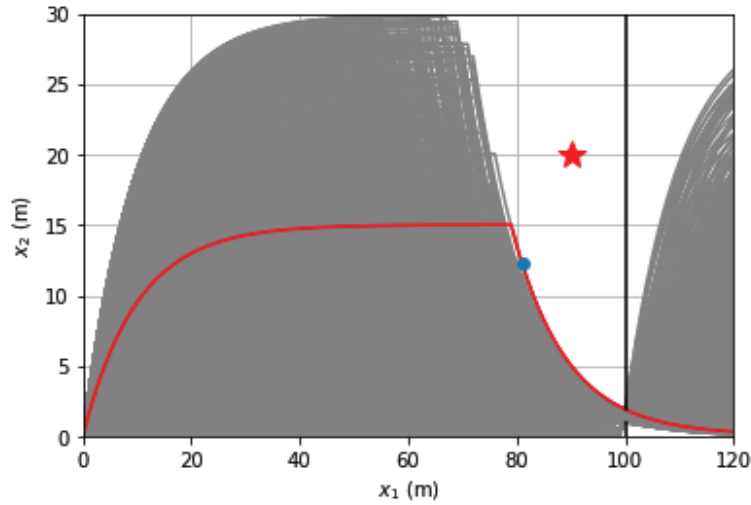


Figure 13: From 100,000 random trajectories, the feasible trajectories that fit through the aperture are shown in gray, and the optimal trajectory is in red. The blue dot is the point on the optimal trajectory nearest the red star.

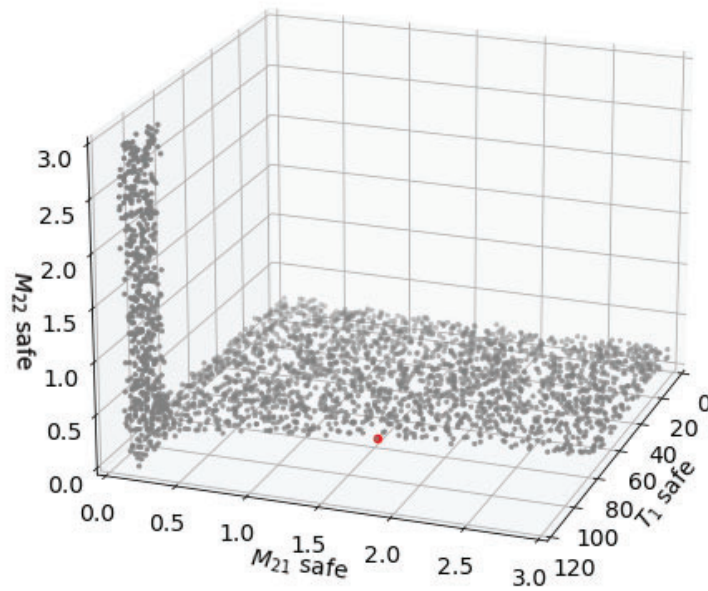


Figure 14: From 100,000 random decision variable combinations, the feasible input combinations for the UUV to pass through the aperture are shown in gray. The optimal input combination is shown as the point highlighted in red.

4 Ship Application

4.1 Autonomous Surface Ships

Surface ships are an application that allow for demonstration of the supervisor. Maritime autonomous surface ships (MASS) are the particular brand of surface vessels to which this research is applied. In the current state of technology, MASS have varying levels of autonomy when it comes to navigation [22]. At the first level, a task such as steering may be automated so that the vessel may be set to follow a desired course made up of waypoints set by a seafarer. At the second level, the ship is controlled from a different location, but seafarers are still onboard and available to take control if necessary. The third level is the same as the second, but with the seafarers removed from the vessel. Finally, at the fourth level, fully autonomous ships operate with internal control systems.

This research is meant to focus its application on levels one, two, and three of MASS. It is important that there is a human available to take over the controls because the supervisor can only protect the MASS for the duration of the time horizon. It is also important that the ship has, at the very least, an autopilot sending control signals to the ship's rudder based on a desired state and sensor readings, such as the ship's heading or geographic position.

4.2 Nomoto Model

Since this research aims to demonstrate a method of determining control signal limits for surface ships, a model is necessary to describe the system dynamics; the Nomoto model is used for the application of this research [28]. True ship motion has six degrees of freedom: heave, sway, surge, roll, pitch, and yaw. The first order Nomoto model only considers three degrees of freedom: surge, sway, and yaw. Also, the model adheres to the following assumptions [29].

1. The ship is symmetric about the x-z plane.

2. The ship's mass is homogeneously distributed.
3. The ship's body has a fixed origin.
4. Heave, roll, and pitch are zero.
5. The sway, yaw rate, and rudder angle are small.

The first order model is given by (32) and (33) where ψ is the ship angle in radians, r is the yaw rate in radians per second, and δ is the rudder input in radians [30]. Constant τ is the specific time constant in seconds given by $\tau = \frac{Tp \cdot L}{U}$. Constant K is the specific rudder gain in s^{-1} given by $K = \frac{Kp \cdot U}{L}$. Constant $L = 160$ m is the length of the vessel, $Kp = 2.9$ is the unitless rudder gain, $Tp = 10$ is the unitless time constant, and $U = 8$ m/s is the constant surge speed. These constants are based on hydrodynamic coefficients calculated for a mariner class tanker ship [31].

$$\frac{d\psi}{dt} = r(t) \quad (32)$$

$$\frac{dr}{dt} = \frac{-1}{\tau} r(t) + \frac{K}{\tau} \delta(t) \quad (33)$$

The Nomoto model can also be described in the form of a continuous state space:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (34)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \quad (35)$$

Where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ K/\tau \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \psi \\ r \end{bmatrix}, \mathbf{u} = \delta$$

As stated previously, dynamic models can also be represented in the form of transfer functions. Each state variable in the Nomoto model is defined by a transfer function.

$$\frac{\Psi(s)}{\Delta(s)} = \frac{K/\tau}{s(s + 1/\tau)} \quad (36)$$

$$\frac{R(s)}{\Delta(s)} = \frac{K/\tau}{s + 1/\tau} \quad (37)$$

From the transfer function or state equations, trajectories of ship dynamics can be generated in the state space. For example, Fig. 15 shows a trajectory in the state space when a 15 degree rudder angle is applied for 120 seconds.

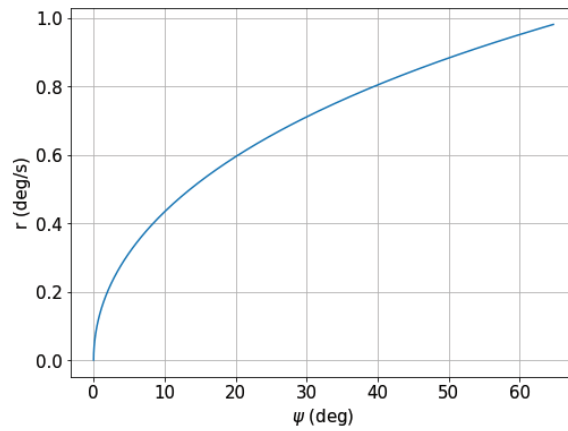


Figure 15: State space trajectory for the Nomoto model with a rudder angle of 15 degrees applied for 120 seconds.

The rudder angle, δ , is treated as the control signal. Through integration, the model's position in the plane of motion can be calculated with (38) and (39) and plotted in Fig. 16.

$$x(t) = \int_0^t U \cos(\psi(\tau)) d\tau \quad (38)$$

$$y(t) = \int_0^t U \sin(\psi(\tau)) d\tau \quad (39)$$

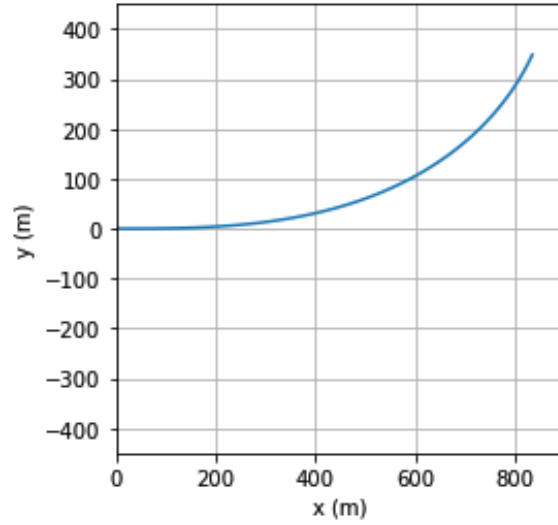


Figure 16: Nomoto model trajectory for a rudder angle of 15 degrees applied for 120 seconds. Initial conditions are zero.

4.3 Autopilot Controller

The Nomoto model assists in determining the controller necessary to drive a physical ship. A linear path following controller is one aspect of the ship's guidance and navigation control system. Assuming a linear path has already been planned, the linear path following control system must latch onto the path and determine the appropriate rudder angle to steer the ship along the planned path [22]. The controller adheres to the line of sight guidance law to calculate the desired yaw angle based on the ship's current position relative to the path. The guidance law, given by (40), attempts to drive the ship to the point on the path that is one lookahead distance ahead of the point on the path perpendicular to the ship [32].

$$\psi_d = \gamma_p + \arctan\left(\frac{-y_e}{\Delta_h}\right) \quad (40)$$

Where ψ_d is the desired yaw angle, y_e is the perpendicular distance between the ship and the nearest point on the path, Δ_h is the lookahead distance, and γ_p is the horizontal path-tangential angle given by (41).

$$\gamma_p = \arctan\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \quad (41)$$

where x_k and y_k are the horizontal and vertical coordinates of waypoint k of the path in the plane of motion in meters. The heading controller used for simulations in this report compares ψ_d from (40) to the ship's actual yaw angle, and uses control parameters $K_p = 5$ and $K_d = 400$ to determine the control signal.

4.4 Analytical Approach

The goal of the analytical approach is to develop a closed form solution describing the relationship between the rudder angle and ship position in the plane of motion. From this equation, a rudder angle limit can be solved for from any given bound in the plane of motion.

4.4.1 Steady State Limits

First, control signals can be bounded from defining constraints for the steady state. A transfer function takes the form of (42) with a step input of magnitude D radians given as $\Delta(s) = \frac{D}{s}$.

$$G(s) = \frac{R(s)}{\Delta(s)} = \frac{K/T}{s + 1/T} \quad (42)$$

First, the steady state output is constrained with upper and lower limits as

$$r_\beta \leq r(\infty) \leq r_\alpha \quad (43)$$

From control theory, it is known that $r(\infty) = G(0)D = KD$. Therefore, (43) can be rewritten as (44), where the magnitude for the rudder input that bounds the steady state is given by the lower and upper limits.

$$\frac{r_\beta}{K} \leq D \leq \frac{r_\alpha}{K} \quad (44)$$

This approach requires a dynamic system like the transfer function from rudder angle to yaw rate that will eventually reach a steady state. Not all transfer functions are stable in this way. For example, the Nomoto model's transfer function from rudder angle to ship angle given by (45) is unstable. Plugging in zero for s leads to an undefined solution. The transfer function from Δ to Ψ does not have a steady state that can be bounded by upper and lower limits.

$$G(s) = \frac{\Psi(s)}{\Delta(s)} = \frac{K/T}{s(s + 1/T)} \quad (45)$$

One issue with the steady state technique is that it does not allow for the ship's yaw rate to be constrained in the transient state where bounds are most desired. This means a ship making a turn can reach yaw rates and rudder angles outside of the prescribed limits before the yaw rate reaches stability. Additionally, this method only works for stable state variables and does not satisfy the ultimate goal of bounding the system from danger areas in the plane of motion.

4.4.2 Transient Limits

From the Nomoto model's transfer functions, heading and yaw rate can be solved for as functions of time, assuming a constant rudder angle of D radians. First, (36) and (37) are expanded through partial fractions to get (46) and (47) respectively.

$$\Psi(s) = -\frac{TDK}{s} + \frac{DK}{s^2} + \frac{TDK}{s + 1/T} \quad (46)$$

$$R(s) = \frac{DK}{s} - \frac{DK}{s + 1/T} \quad (47)$$

Applying the inverse Laplace transform to (46) and (47) returns solutions (48) and (49) respectively.

$$\psi(t) = -TDK + DKt + TDK e^{-t/T} \quad (48)$$

$$r(t) = DK - DK e^{-t/T} \quad (49)$$

Each closed form solution can be manipulated to solve for D . With (50) and (51), a constant rudder limit can be found given t_f , and $\psi(t_f)$ or $r(t_f)$.

$$D = \frac{\psi(t_f)}{TK e^{-t_f/T} + K t_f + TK} \quad (50)$$

$$D = \frac{r(t_f)}{K - K e^{-t_f/T}} \quad (51)$$

The problem remains that the rudder limit cannot be solved for given time and a position in the plane of motion. This step requires solving integrals (52) and (53). Unfortunately, an exact closed form solution to either of these integrals has not been found.

$$x(t) = \int U \cos(-TDK + DKt + TDK e^{-t/T}) dt \quad (52)$$

$$y(t) = \int U \sin(-TDK + DKt + TDK e^{-t/T}) dt \quad (53)$$

However, a Maclaurin series expansion about $t = 0$ can be applied to approximate $\frac{dx}{dt}$ and $\frac{dy}{dt}$ as poly-

nomials that can be integrated for $x(t)$ and $y(t)$.

$$x(t) \approx -\frac{D^2 K^2 U t^5}{40 T^2} + \frac{D^2 K^2 U t^6}{72 T^3} - \frac{5 D^2 K^2 U t^7}{1008 T^4} + U t \quad (54)$$

$$y(t) \approx \frac{D^3 K^3 U t^7}{336 T^3} + \frac{D K U t^3}{6 T} - \frac{D K U t^4}{24 T^2} + \frac{D K U t^5}{120 T^3} - \frac{D K U t^6}{720 T^4} + \frac{D K U t^7}{5040 T^5} \quad (55)$$

As shown in Fig. 17, these approximations can be used to calculate the x and y position of the ship at any point in time given a constant rudder angle, D .

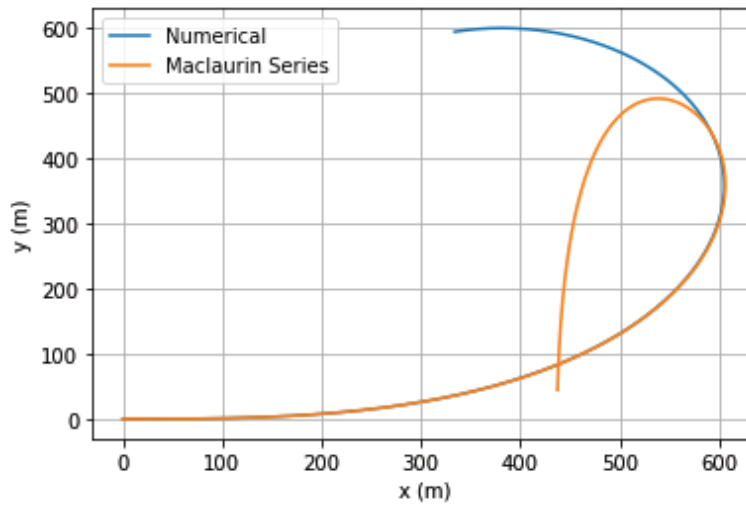


Figure 17: Position trajectories for the numerical and Maclaurin series approximations for a rudder angle of 15 degrees applied for a duration of 150 seconds.

The error between the numerical and Maclaurin series approximations is proportional to time and rudder angle magnitude. The error is defined as the euclidean distance between the trajectories at each point in time. Figure 18 shows the error in the approximation for various constant rudder angles applied to the Nomoto model over time.

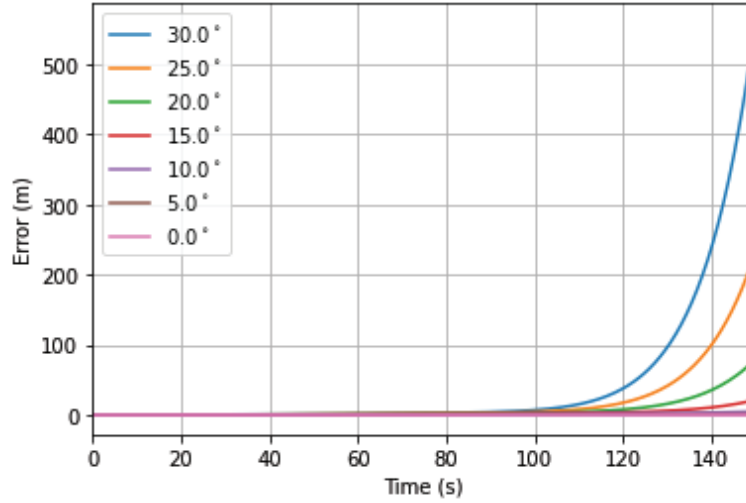


Figure 18: Error between the Maclaurin series approximation and numerical approximation. The error increases with time and rudder angle magnitude.

For constant, shallow rudder angles, the Maclaurin series can accurately predict the ship's position up to two minutes into the future as shown in Fig. 18. Increasing the number of terms in the series can further increase the approximation's accuracy.

4.5 Reachable Set Approach

4.5.1 Attack Model

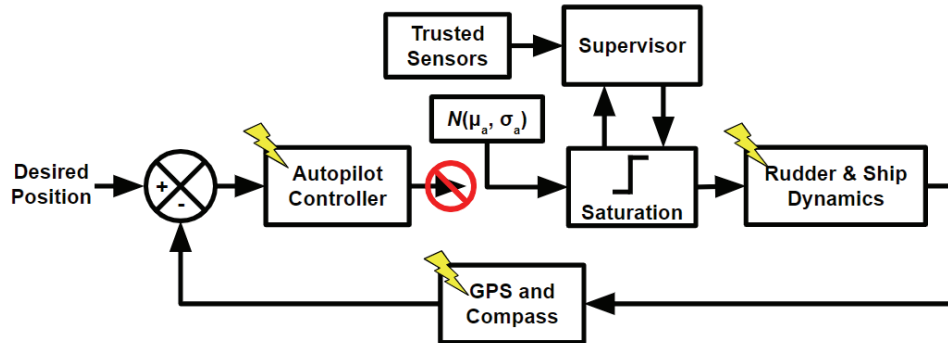


Figure 19: Attack model for the autopilot's feedback loop. The desired rudder command is overwritten with Gaussian noise to model the effects of an attack on any of the loop's vulnerable components (denoted with lightning bolt symbols). The supervisor, a high authority component, saturates the injected control signal to protect the system from reaching unsafe states.

The reachable set approach requires more specific attack assumptions. Like other methods, it is assumed that the signal to the rudder is overwritten in an unpredictable manner. For the reachable set approach, the injected signal is modeled as a normally distributed random variable $\delta_a \sim \mathcal{N}(\mu_a, \sigma_a)$ degrees. The injection may be considered the result of an attacker targeting any one of the vulnerable system components shown in Fig. 19. For example, the injection could be the effect of electromagnetic interference or sensor manipulation. The supervisor is assumed to be invulnerable to attack because it is engineered as a separate subsystem with its own protected information channels.

It is assumed that magnitude of the attack is estimated as $\sigma_a = 17.5^\circ$. This value is selected so that a 95% confidence interval ranges from -35° to 35° , which covers the most extreme rudder angle magnitudes that a ship can achieve. The standard deviation can be further estimated from data sampled during a cyber attack.

4.5.2 Determining Control Signal Limits

The method in section 3.3.3 determines control signal limits for danger areas in the state space. In the case of ship motion, it is desirable to determine control signal limits to stay away from danger areas in the plane of motion. Defining danger areas in the plane of motion is important for protecting the ship from collisions. To extend the reachable set to the plane of motion, it is necessary to convert the ellipse to geometry in the plane of motion with (38) and (39). Each ellipse in the state space is converted to an arc in the plane of motion. Repeating this process for each time step leading up to the time horizon produces reachable sets as shown in Figs. 20 and 21. The time horizon should be estimated as the duration for which an attack can safely take place before a collision occurs. The time horizon may depend on the systems safety mechanisms and human response time.

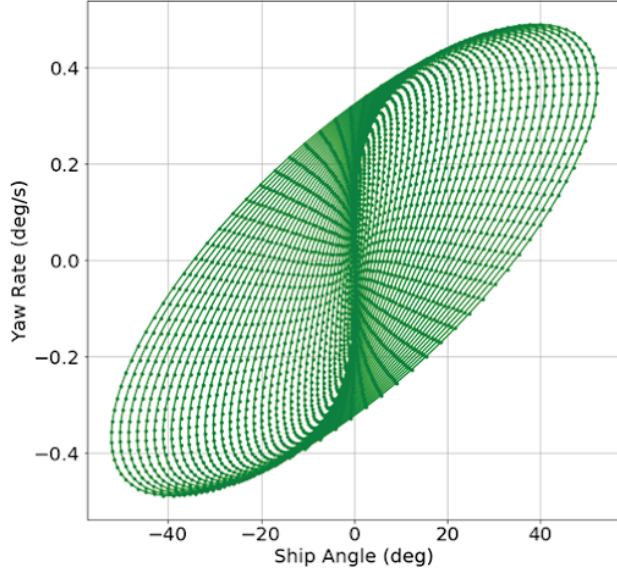


Figure 20: The approximate reachable set in the state space for an injection attack with the distribution $\mathcal{N}(0, 17.5)^\circ$ propagated out to a time horizon of 180 seconds with a 5.0 second sample time.

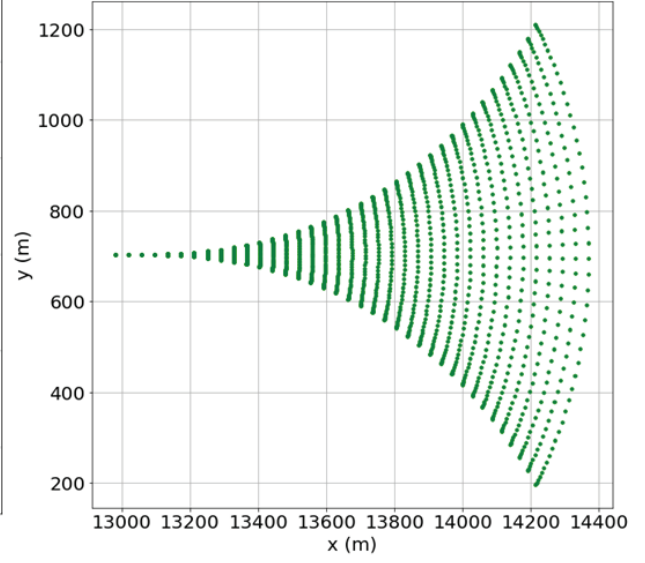


Figure 21: The approximate reachable set in the plane of motion for an injection attack with the distribution $\mathcal{N}(0, 17.5)^\circ$ propagated out to a time horizon of 180 seconds with a 5.0 second sample time.

The standard deviation, σ_δ , which produces the largest reachable set that does not overlap the danger area determines the optimal rudder limit and is calculated by starting at a large value and decreasing until the reachable set in the plane of motion does not overlap the danger area. Assuming a 95% confidence interval, the rudder limits are set at $\pm 2\sigma_\delta$. In implementation, the supervisor follows piece-wise function (56) to saturate δ_a .

$$\delta = \begin{cases} \delta_a & \text{if } |\delta_a| \leq 2\sigma_\delta \\ 2\sigma_\delta & \text{if } \delta_a > 2\sigma_\delta \\ -2\sigma_\delta & \text{if } \delta_a < -2\sigma_\delta \end{cases} \quad (56)$$

4.6 Monte Carlo Approach

For the Monte Carlo approach, the supervisor applies random sampling to identify control signal limits that are safe. The Monte Carlo approach's attack assumptions are not as restrictive as those of the reachable set approach. The injected control signal does not have to be sampled from a Gaussian distribution; the distribution can be uniform. Moreover, the attack can take the form of a random walk in which each injected rudder angle is a step up or down from the previous rudder angle. The random walk attack model is fitting for cases in which the rudder is attacked directly. In this example, the rudder is assumed to take on the form of random walk in which the rudder angle is randomly increased or decreased by five degrees every 10 seconds.

To set up the Monte Carlo simulation, a pseudo-population is defined from which samples are drawn. For this example, it is assumed that a ship can take on any rudder angle between -35 and 35 degrees, the standard physical rudder limits on a ship. The pseudo-population is defined as a uniform distribution with left and right limits of -35 and 35 degrees respectively. Second, the sample is selected from the pseudo-population in the form of a random walk starting at zero degrees and stepping up or down by 5 degrees every 10 seconds. An example is shown in Fig. 22. This limits the rudder's rate of change to 0.5 degrees per second which is based on the rudder limitations of the mariner vessel [31].

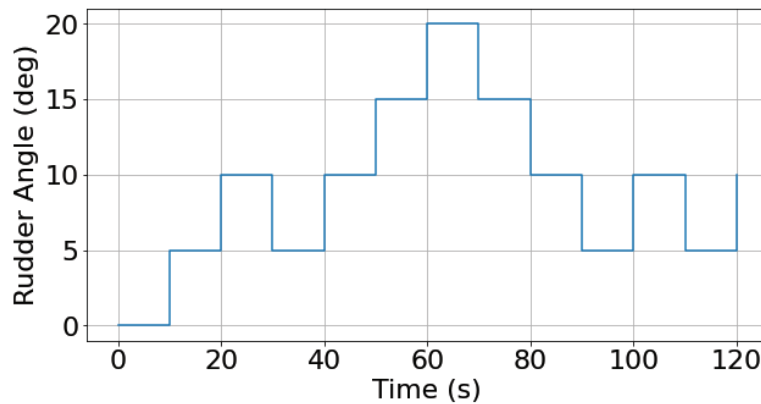


Figure 22: One example of a random walk input for the rudder angle. The rudder angle steps up or down randomly by 5 degrees at 10 second intervals for a duration of 120 seconds. For each random walk, the rudder angle is bounded between -35 and 35 degrees while the rudder angle rate is limited to a magnitude of 0.5 degrees per second.

Third, the resulting ship trajectories are observed in the state space and plane of motion in Figs. 23 and 24 respectively. After a large number of samples, the trajectories fill in the reachable set. Each trajectory leads to a state or position of which the system is capable of reaching after 120 seconds. When one of these trajectories intersects with an unsafe region in the state space or plane of motion, the rudder angle is recorded and classified as unsafe by the supervisor. The supervisor filters out the unsafe rudder commands, so that only safe rudder commands are sent to the rudder.

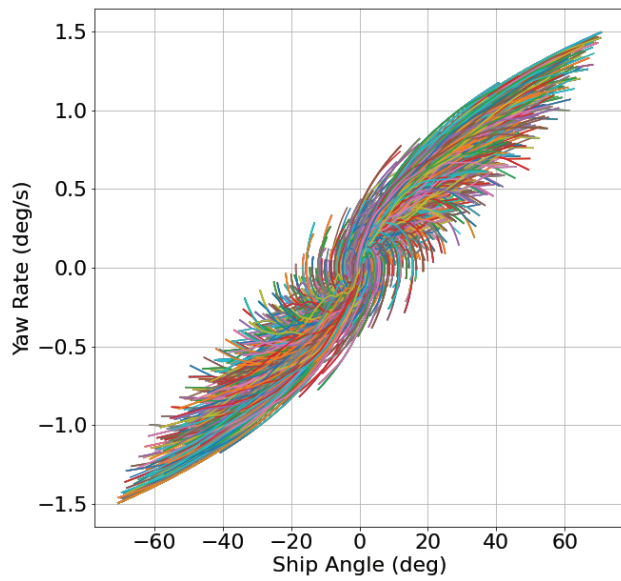


Figure 23: The ship angle and yaw rate state space shows 1000 possible trajectories.

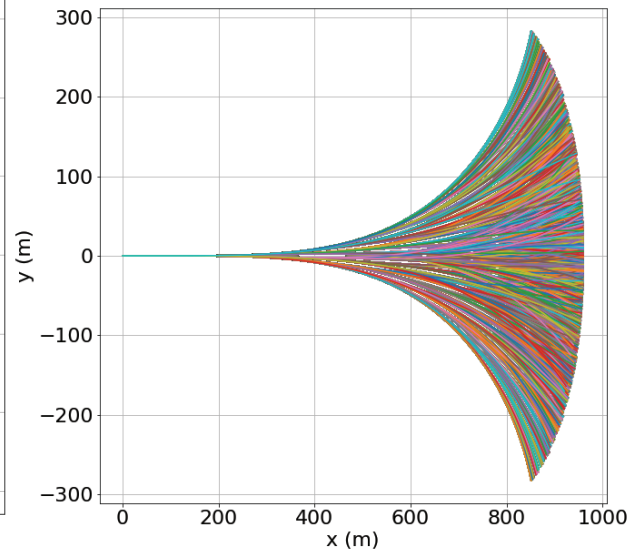


Figure 24: The plane of motion shows 1000 possible trajectories.

The Monte Carlo approach has some advantages over the reachable set approach. The Monte Carlo approach allows for complex filters to limit the rudder rather than simple left and right rudder limits. The Monte Carlo approach also has less restrictive attack assumptions in that it does not require the injected rudder angles to be sampled from a Gaussian distribution. The main disadvantage of the Monte Carlo approach, however, is that it requires more computing power than the reachable set approach.

5 Synthetic Channel Simulation

5.1 Problem Statement

For a basic demonstration of determining control signal limits, the supervisor is tested in simulation with the Nomoto model. A ship is tasked with navigating from left to right through the synthetic channel shown in Fig. 25, and is subject to a cyber attack at each channel width. The supervisor is tasked with calculating and enforcing rudder limits for the control signal passed from the autopilot controller to the rudder. The reachable set method is employed to determine rudder limits.

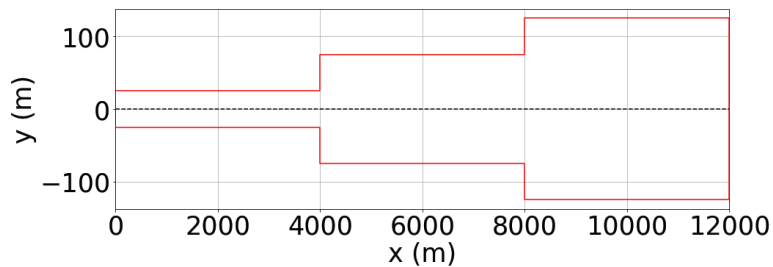


Figure 25: The synthetic channel through which the ship travels from left to right. Channel width increases from 50 m to 150 m to 250 m.

5.2 Solution

For the synthetic channel simulation, the supervisor does not calculate rudder limits as it drives through the channel. Instead, the supervisor calculates rudder limits ahead of time based on the assumption that the ship is always positioned on the desired path in the center of the channel. The supervisor then enforces the calculated limits while the ship is driving through the channel. The assumption that the ship is always in the center of the channel causes issues when an attack forces the ship to approach a channel boundary. The supervisor cannot recalculate rudder limits while the ship is driving, which means the ship can be forced to the outermost point of the calculated reachable set. This would not be an issue if the ship could immediately stop or turn at the edge of the channel, however, ships carry a large amount of momentum that resists changes in

speed and direction. The ship's momentum carries it beyond the channel boundary, running the ship aground. To address this issue, a buffer zone is implemented inside of the channel bounds.

To account for a ship's momentum carrying it outside of its reachable set, an artificial, or soft channel boundary is inserted inside the physical, or hard channel boundary in the plane of motion. The gap between the two channels acts as a buffer zone that allows time for the autopilot to correct the ship.

The soft channel width, w_s , is always less than the hard channel width, w_h . The relationship between the hard and soft channel widths is approximated through experimentation. The soft width is varied, and for each width, the rudder limit is calculated using the methods in section 4.5.2. Using each calculated rudder limit as a constraint, 1000 normally distributed attacks are simulated for each of the soft channel widths and last for 90 seconds in duration. The hard channel width is calculated as 6 times the standard deviation of the collection of the most extreme vertical position on each trajectory. With a prediction interval of $\pm 3\sigma_v$, it is expected that the hard channel bounds fully enclose 99.7% of the attack trajectories. Figure 26 depicts the soft and hard channel boundaries with the 1,000 attack trajectories overlaid. A polynomial curve is fitted to the data and inverted to return w_s as a function of w_h . Function (57) is plotted with the data points in Fig. 27. The function is piece-wise because as the widths increase, the largest reachable set reaches its own limit and no longer requires a soft channel boundary to constrain it. At that point, the soft channel boundary may take on its maximum value, w_h , to allow for the most freedom of maneuverability.

$$w_s = \begin{cases} -\sqrt{\frac{260.5-w_h}{0.02625}} + 98.08 & 0 \leq w_h \leq 260.5 \\ w_h & 260.5 < w_h < \infty \end{cases} \quad (57)$$

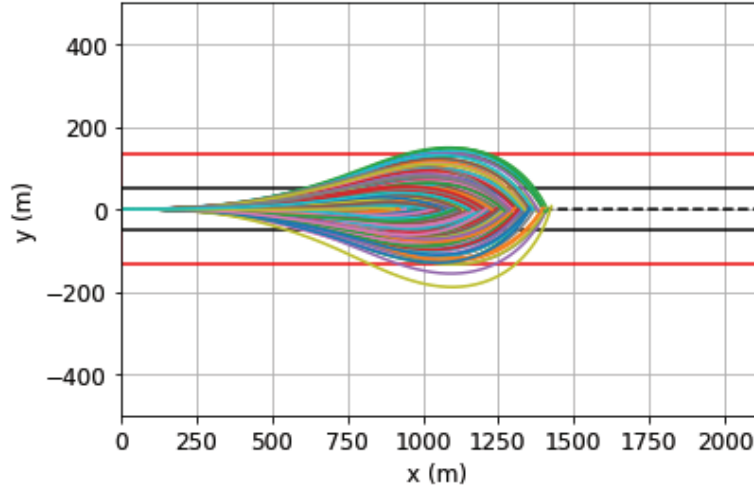


Figure 26: Trajectories resulting from normally distributed overwrite attacks on a 105.0 meter wide soft channel. After the 90 second attack duration, the autopilot employs a rudder with maximum magnitude to drive the ship back to the baseline trajectory. The hard channel width is determined as $6\sigma_v = 264.0$ meters. Of the 1,000 trajectories, only 9 collide with the hard channel boundary.

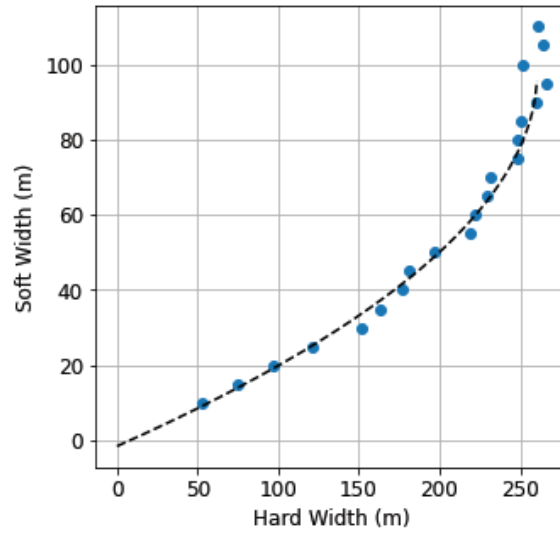


Figure 27: A curve is fitted to the experimental data and inverted to allow for the determination of soft channel width as a function of hard channel width.

5.3 Results

Cyber attacks adhering to the attack model outlined in section 4.5.1 occur at three locations along the route and last for a duration of 90 seconds. Two scenarios are compared. In the first scenario, artificial rudder

limits are disabled, so the rudder angle is only constrained to its natural limits of ± 35 degrees. In the second scenario, the supervisor enforces the rudder limit predetermined by the supervisor's search method.

The supervisor's search method is a planning process in which the supervisor determines the control signal limit at certain points along the desired route. At each search point, the supervisor uses the methods outlined in section 4.5.2 to calculate the reachable set and safe rudder magnitude associated with that point. The supervisor enforces the limit until the ship reaches the next search point. The planning process is depicted in Fig. 28 and the rudder limit, $2\sigma_\delta$, for each channel segment is listed in Table 1.

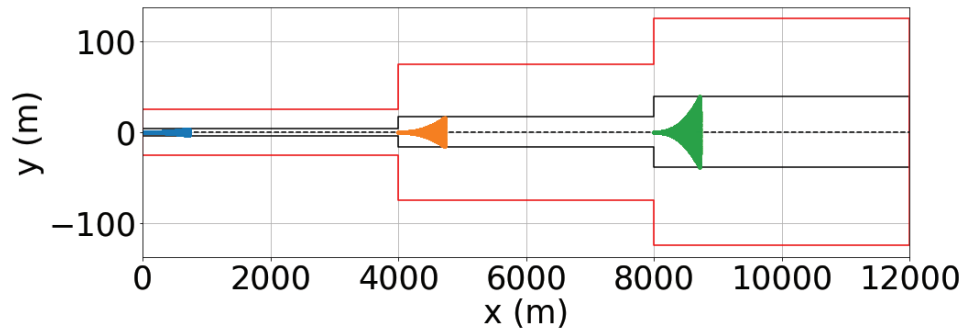


Figure 28: The rudder limit for each channel segment is determined from the largest reachable set in each segment. The reachable sets show the region of the plane of motion in which the ship is capable of reaching during an attack.

Table 1: Channel Widths and Rudder Limits

	Seg. 1	Seg. 2	Seg. 3
w_h (m)	50	150	250
w_s (m)	8.534	33.20	78.10
$2\sigma_\delta$ ($^\circ$)	2.143	10.00	25.00

For both scenarios, Monte Carlo simulations of 10,000 runs demonstrate the various trajectories that occur as a result of the three cyber attacks. Each attack lasts for 90 seconds. During the attack phase, the autopilot's control signal is overwritten with random rudder commands adhering to the distribution $\mathcal{N}(0, 17.5)^\circ$. When the attack ends, the recovery phase begins, in which the PD autopilot controller drives the ship back to the desired course.

The success of the supervisor is measured in its ability to prevent collisions with the hard channel boundary. Collision counts are located in Table 2. In the first scenario, shown in Fig. 29, 61.0% of trajectories collide with the hard channel boundary. In the second scenario, shown in Fig. 30, only 2.61% of trajectories collide with the hard channel boundary. Under the given assumptions, the ship is 23.4 times more likely to remain safe with the supervisor enabled than with the supervisor disabled.

For segment three of the channel, there is only a minor improvement between the number of collisions with the supervisor compared to when it is disabled. This difference is due to the fact that, in segment three, the rudder magnitude's limit, $2\sigma_\delta = 25.00^\circ$ is near the ship's strict physical rudder magnitude's limit of 35° . As the artificial limits near the physical limits, the effects of the supervisor are less visible.

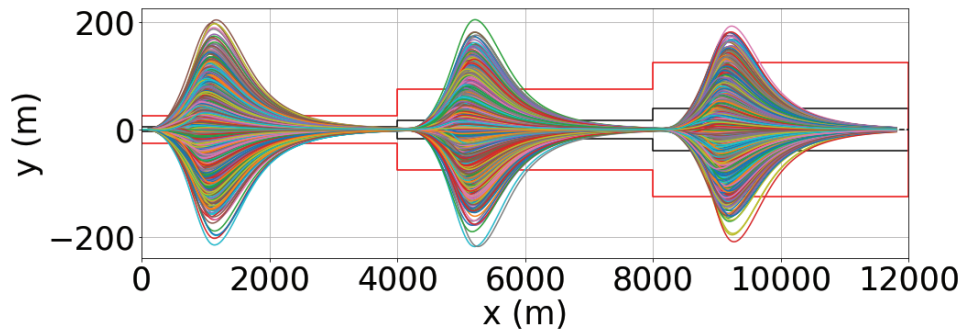


Figure 29: With the supervisor disabled, 61.0% of the trajectories collide with the hard channel boundary.

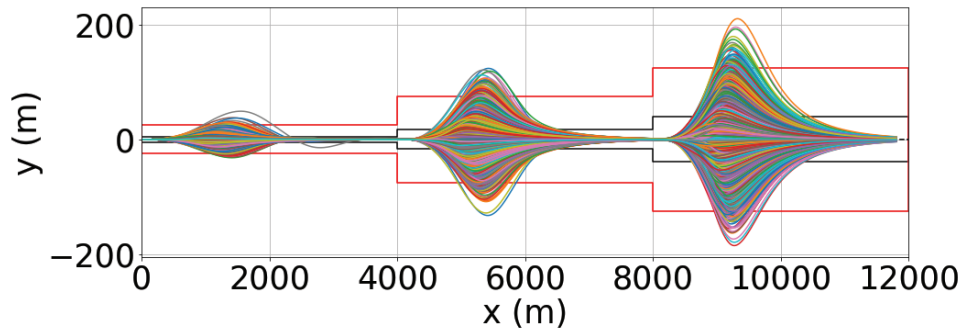


Figure 30: With the supervisor enabled, only 2.61% of the trajectories collide with the hard channel boundary.

Table 2: Number of Collisions per Channel Segment

	Seg. 1	Seg. 2	Seg. 3	Overall	Overall (%)
Collisions (limits disabled)	5102	882	111	6095	61.0
Collisions (limits enabled)	40	117	104	261	2.61

6 San Diego Bay Channel Simulation

To further demonstrate the supervisor’s capabilities, it is desirable to test its functionality in a more realistic environment. To increase realism, a simulation is designed in which a ship with Nomoto model dynamics must follow a desired path out of the San Diego Bay and avoid collision with the coastline while subject to three independent cyber attacks.

6.1 Coastline Data

One feature available that allows for increased realism in simulations is vector coastlines. The Global Self-consistent, Hierarchical, High-resolution Geography Database (GSHHG) is a resource for reliable shoreline data constructed from hierarchically arranged closed polygons [33]. To display the data, Basemap, a special toolkit for Python, allows for the plotting of coastlines and other data on 2D maps [34].

In selecting a map for a simulation, multiple ports and bays were considered. Ultimately, the San Diego Bay was selected for its curved and lengthy channel.

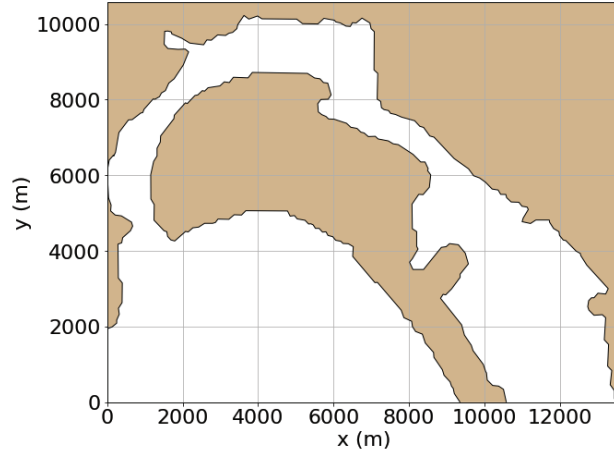


Figure 31: A Mercator projection of the San Diego Bay produced with Basemap.

6.2 Automatic Identification System Tracks

Another resource that is useful for generating simulations is Automatic Identification System (AIS) data retrieved from vessel tracking systems. While ships use radar for navigation, they also have the support of AIS for identifying any ship, its position, course, speed, and other identifying features. While AIS is intended for ships to track maritime traffic in a live environment, some shore-based AIS transceivers also upload the data to AIS service providers [35]. AIS data is publicly available on sites like Marine Cadastre, an internet-based AIS service provider [36]. AIS tracks can be plotted using Basemap as shown in Fig. 32.

For simulations in this report, vessel tracks are extracted from existing AIS data and used to determine desired paths for an artificial ship to follow. The points from the *Blue Ridge Highway's* track, shown in Fig. 34, are redefined as waypoints for the artificial vessel's desired path. AIS transmissions take place roughly every 60 seconds, so raw waypoints are spaced fairly close together. Typically, it is desirable for ships to navigate along straight paths as much as possible, so the raw path can be simplified with the Ramer-Douglas-Peucker (RDP) algorithm [37]. The RDP algorithm requires a λ parameter in order to simplify the path into fewer waypoints. The *Blue Ridge Highway's* outbound track in the San Diego Bay is displayed with its RDP simplified path in Fig. 33.

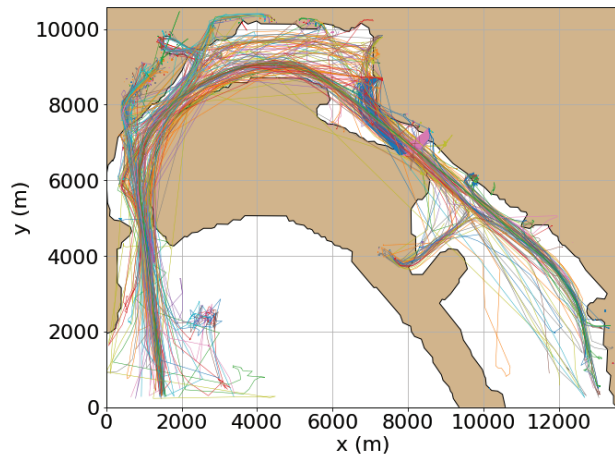


Figure 32: AIS vessel tracks in the San Diego Bay over a 48 hour period in October 2017.

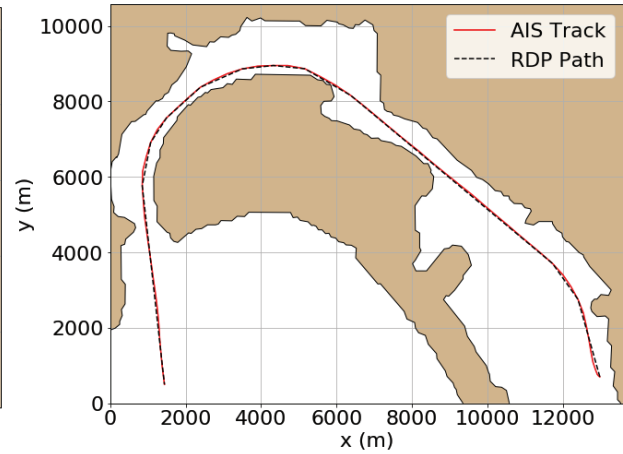


Figure 33: An isolated AIS track with its simplified RDP path overlaid. The RDP algorithm uses the parameter $\lambda = 100\text{m}$

BaseDateTime	LAT	LON	SOG	COG	Heading
2017-10-28T07:04:41	32.65575	-117.12280	0.6	-105.6	347.0
2017-10-28T07:05:51	32.65587	-117.12303	0.7	-103.6	344.0
2017-10-28T07:07:01	32.65623	-117.12340	1.8	-74.6	343.0
2017-10-28T07:08:11	32.65715	-117.12382	3.7	-69.6	343.0
2017-10-28T07:09:22	32.65867	-117.12443	5.6	-66.6	347.0
2017-10-28T07:10:31	32.66075	-117.12492	6.9	-61.6	350.0
2017-10-28T07:11:41	32.66318	-117.12540	7.9	-60.6	350.0
2017-10-28T07:12:51	32.66580	-117.12598	8.5	-60.6	350.0
2017-10-28T07:14:00	32.66855	-117.12672	8.8	-65.6	344.0
2017-10-28T07:15:12	32.67127	-117.12800	8.9	-75.6	334.0
2017-10-28T07:16:21	32.67383	-117.12967	9.1	-79.6	332.0
2017-10-28T07:17:30	32.67623	-117.13162	9.1	-86.6	323.0
2017-10-28T07:18:40	32.67848	-117.13407	9.1	-96.6	313.0

Figure 34: A sample dataframe for the Panamanian *Blue Ridge Highway* vessel with the time of each AIS transmission, the vessel's position in degrees latitude and longitude, speed over ground (SOG), course over ground (COG), and heading in degrees.

6.3 Advanced Features of the Supervisor

The narrow channel simulation from section 5 assumes that the ship is always positioned along the desired path when the supervisor calculates actuator limits. As a result, the supervisor cannot adjust limits as an attack forces the ship closer to a channel boundary. To compensate for this, a buffer zone is implemented for the narrow channel simulation that further restricts the rudder limits.

The following simulations remove the necessity for a buffer zone by calculating rudder limits while the ship is driving. Dynamically determined rudder limits introduce a new problem; when an attack forces the ship to drive towards a boundary, the rudder limits decrease in size until the rudder has no more freedom in either direction. The supervisor then forces the ship to proceed in the direction of its current heading and run aground. To fix this, the supervisor affords the rudder the ability to turn by shifting the baseline rudder angle as part of the search algorithm. In the first narrow channel simulation, the search algorithm considers only one degree of freedom while determining rudder limits: the standard deviation. Allowing for a shift in the baseline rudder angle introduces a second degree of freedom to the supervisor's search algorithm.

With a second degree of freedom for searches, the supervisor's search algorithm starts with a wide rudder limit and a zero degree baseline rudder. The supervisor decreases the rudder limit's magnitude until it either finds a reachable set that fits inside of the channel, or the supervisor fails to find a reachable set that fits. If the supervisor fails to find a reachable set that fits, it shifts the baseline to a small positive rudder angle and attempts to determine the largest reachable set that fits inside the channel. Again, if the supervisor fails to find a reachable set that fits, it shifts the baseline to a small negative rudder angle and tries again. The process repeats itself until either a solution is found or all possibilities are extinguished. When limits are found, the rudder is constrained with piece-wise function 58.

$$\delta = \begin{cases} \delta_a + \mu_\delta & \text{if } -2\sigma_\delta \leq \delta_a \leq 2\sigma_\delta \\ 2\sigma_\delta + \mu_\delta & \text{if } \delta_a > 2\sigma_\delta \\ -2\sigma_\delta + \mu_\delta & \text{if } \delta_a < -2\sigma_\delta \end{cases} \quad (58)$$

The baselines and standard deviations the supervisor uses to complete its searches are predetermined before the search takes place. The baseline, μ_δ varies between -35° and 35° , and the standard deviation, σ_δ , ranges from 17.5° to 0° . The spacing between each baseline is defined as the baseline resolution and the spacing between each standard deviation is defined as the noise resolution. A high baseline resolution provides the supervisor a greater number of possible trajectories for the supervisor to choose from. The search takes longer, but a high baseline resolution makes the supervisor more likely to find narrow openings in a channel. A high noise resolution allows for more accurate limits that maximize the reachable set. A low noise resolution allows for faster searches, but can overconstrain the rudder limits. The following simulation uses a baseline resolution of 5° , and a noise resolution of approximately 2.92° .

The duration between the start times of searches is defined as the search period. A short search period increases a vessels safety because the supervisor scans the environment for obstacles more frequently. A long search period reduces the computational burden, but may not detect limits in time for the ship to react. The following simulation uses a search time of 30 s.

The time horizon is the length of time the supervisor looks ahead of the ship to search for rudder limits. A long time horizon is safer because it allows for the supervisor to protect against longer attacks. A short time horizon, while riskier, also allows for more maneuverability in areas that require tight turns and can allow for faster searches. In the following simulation, the time horizon is 180 s, and it is assumed that each cyber attack lasts for a duration equal to the time horizon.

6.4 Problem Statement

Similar to the synthetic channel simulation, the San Diego Bay channel simulation compares the ship's ability to safely navigate with and without the supervisor while subject to three independent cyber attacks. Safe navigation is quantified by the number of collisions that take place along the ship's path. Cyber attacks take place starting at 100 s, 1050 s, and 2000 s, and last for a duration of 180 s. Each attack is modeled as overwriting the autopilot's control signal with noise sampled from normal distribution $\mathcal{N}(0, 17.5)^\circ$ every 5 seconds.

6.5 Results

First, the supervisor is disabled as the ship drives through the channel. The ship completes 100 runs through the channel, each with random cyber attacks, and the tracks are plotted in Fig. 35.

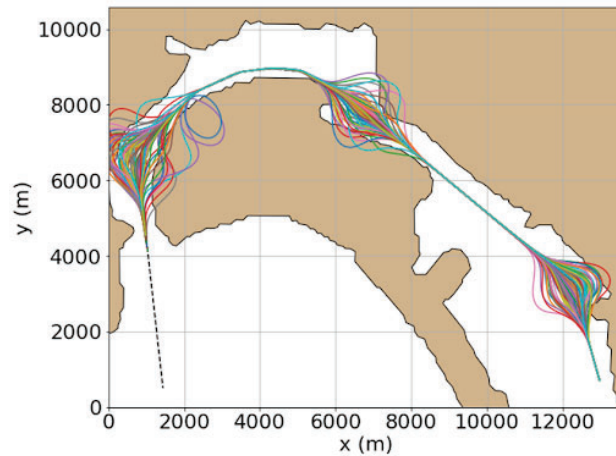


Figure 35: Ship tracks for 100 unsupervised runs through the San Diego Bay.

Second, the supervisor is enabled as the ship drives through the channel. The ship completes 100 runs through the channel with the same cyber attacks as the unsupervised runs, and the tracks are plotted in Fig. 36. The reachable sets used to determine the control signal limits at each search are shown in Fig. 37.

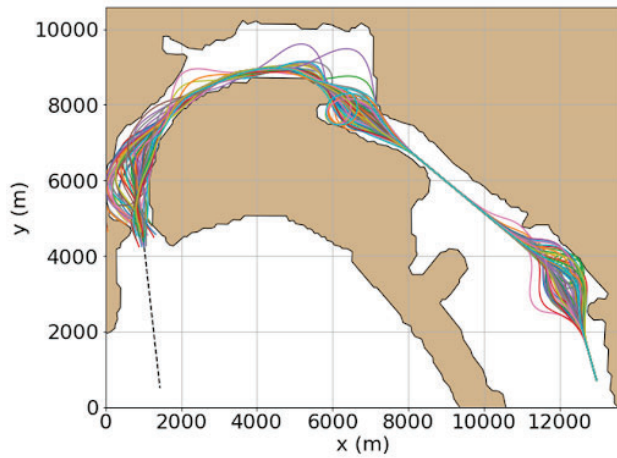


Figure 36: Ship tracks for 100 supervised runs through the San Diego Bay Channel.

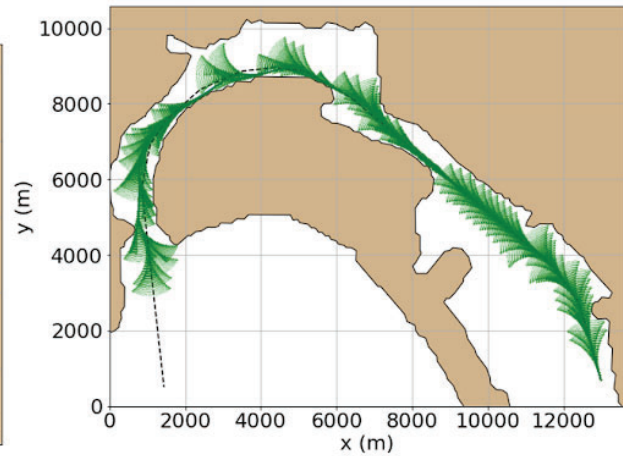


Figure 37: An example of the supervisor's search method as it navigates through the channel. Searches have a time horizon of 180 s and occur every 30 s.

For each search, the supervisor determines a minimum rudder limit, maximum rudder limit, and a baseline rudder. The average rudder angles, rudder limits, and baselines for all 100 supervised runs are shown in Fig. 38.

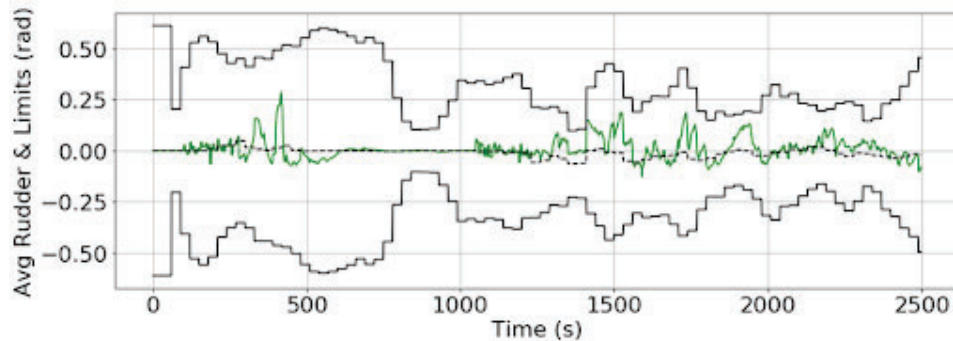


Figure 38: The average rudder limits and average rudder angle for all 100 supervised runs through the San Diego Bay Channel. The solid black lines are the average minimum and maximum rudder limits. The dashed black line is the average baseline rudder angle, and the solid green line is the average rudder angle command received by the rudder.

The simulation does not stop the run when the ship runs ashore. This is intentional in order to allow the ship time to recover for the next attack, which is viewed as independent from the previous attack. This allows for the summation of the number of points in time in which the ship is aground across all 100 simulations.

The sums are shown in Fig. 39 and suggest that, overall, the supervisor reduces the number of times the ship is on land.

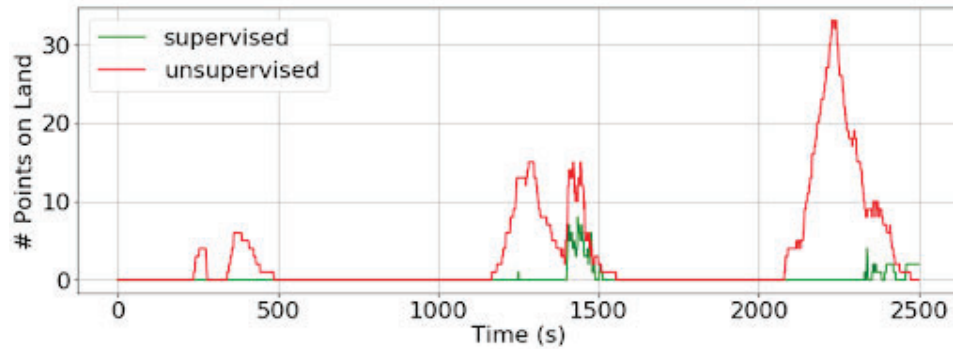


Figure 39: The number of discrete points of time in which the ship is on land for 100 runs of both the unsupervised and supervised systems.

Another way to measure performance is through comparing the number of collisions that take place after each attack. A collision is defined by the ship's transition from sea to land. The number of collisions are provided in Table 3 where the channel is split into three segments based on the start time of each attack.

Table 3: Number of Collisions per Channel Segment

	Seg. 1	Seg. 2	Seg. 3	Overall
Collisions (limits disabled)	6	39	35	80
Collisions (limits enabled)	0	30	11	41

Given the specified parameters, and based on the number of overall collisions, the ship is 1.95 times more likely to avoid running aground with the supervisor enabled than with the supervisor disabled.

6.6 Kalman Filter

The simulations in this report did not take into consideration the effects of measurement and process noise from the effects of sensor error and environmental disturbances respectively. Measurement noise introduces uncertainty into the state of the ship. Process noise also introduces uncertainty so that it is difficult to predict

the future state of the system. A Kalman filter is a tool used to find a better estimate of the state of the system based on a measurement and prediction of the state. The improved state estimate, or correction, is given by $\hat{\mathbf{x}}_{k,k}$ in 59, where $\hat{\mathbf{x}}_{k,k-1}$ is the mean value predicted from (4), $\mathbf{G}$ is the Kalman gain, and $\mathbf{v}_k$ is the measurement. The Kalman filter equations are rewritten from [38].

$$\hat{\mathbf{x}}_{k,k} = \hat{\mathbf{x}}_{k,k-1} + \mathbf{G}_k(\mathbf{v}_k - \mathbf{C}_k\hat{\mathbf{x}}_{k,k-1}) \quad (59)$$

The Kalman gain is provided in (60).

$$\mathbf{G}_k = \mathbf{P}_{k,k-1}\mathbf{C}_k^T(\mathbf{C}_k\mathbf{P}_{k,k-1}\mathbf{C}_k^T + \mathbf{R}_k)^{-1} \quad (60)$$

The estimated, or corrected, covariance matrix is provided in (61).

$$\mathbf{P}_{k,k} = (\mathbf{I} - \mathbf{G}_k\mathbf{C}_k)\mathbf{P}_{k,k-1} \quad (61)$$

An example is shown in Fig. 40, where a correction ellipse is determined from a measurement ellipse and a prediction ellipse.

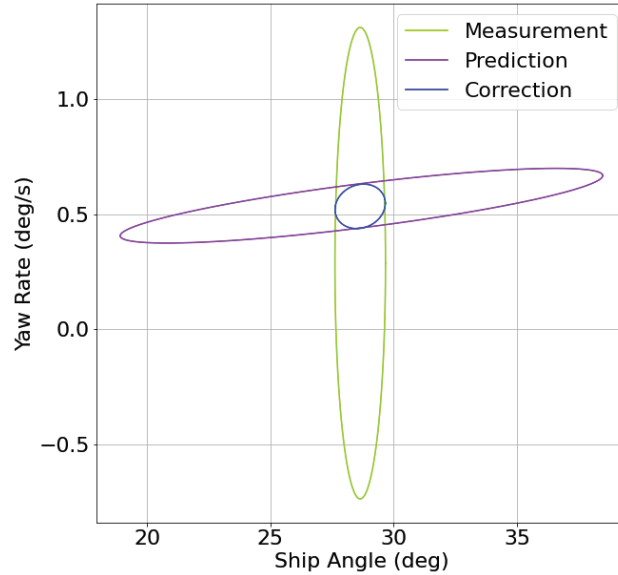


Figure 40: A plot of the prediction uncertainty ellipse, measurement uncertainty ellipse, and correction uncertainty ellipse produced with the Kalman filter. The blue correction ellipse is the smallest because the correction has less uncertainty than the prediction and measurement.

When process noise and measurement noise are introduced to the supervisor simulations, the Kalman filter may be implemented to decrease the size of the reachable set in the plane of motion. A smaller reachable set allows for less restrictive limits, and therefore more freedom of maneuverability.

7 Conclusion

This report explores three methods of determining control signal limits for the protection of cyber-physical systems in the event of a cyber attack. Analytical, reachable set, and Monte Carlo methods are proposed as functions of the high-authority supervisor that calculates and implements control signal limits. The reachable set approach is selected to demonstrate, through simulation, the application of a high-authority supervisor for the rapid calculation and employment of control signal limits. The synthetic channel simulation indicates that, under the assumed cyber attack, a ship is 23.4 times more likely to avoid a collision with a channel boundary with the supervisor enabled than with the supervisor disabled. The San Diego Bay channel simu-

lation indicates that, under the assumed cyber attack, the supervisor makes a ship 1.95 times more likely to avoid a collision with the shoreline.

Future research may see improvements with optimization of the supervisor's main parameters: time horizon, search frequency, and search resolution. While computational power is a limiting factor in the San Diego Bay channel simulation, future simulations should consider running the ship in the order of thousands of simulations. Process and measurement noise may be introduced to the simulations in order to determine the effect of the Kalman filter on maneuverability in the plane of motion. Additional ships may be integrated into future simulation environments to provide dynamic obstacles for the supervisor to avoid. Lastly, the supervisor's functions may be further validated through implementation on a physical testbed.

This report has relevance to the U.S. Navy's Large Unmanned Surface Vessels program, and the Defense Advanced Research Projects Agency's No Manning Required Ship program for the development of unmanned surface vessels [39]. As unmanned surface vessels are integrated into navies, it is important that defensive systems are in place to protect these critical pieces of infrastructure from exploitation by cyber attacks.

References

- [1] T. Seals, "Researcher: Not hard for a hacker to capsize a ship at sea," Feb 2019. [Online]. Available: <https://threatpost.com/hacker-capsize-ship-sea/142077/>
- [2] E. J. Byres and P. Eng, "Cyber security and the pipeline control system," *Lantzville, BC, Canada*, 2009.
- [3] I. Lopez and M. Aguado, "Cyber security analysis of the european train control system," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 110–116, 2015.
- [4] A. P. Mathur and N. O. Tippenhauer, "Swat: a water treatment testbed for research and training on ics security," in *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. IEEE, 2016, pp. 31–36.
- [5] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 2195–2201.
- [6] C. Konstantinou, M. Maniatakos, F. Saqib, S. Hu, J. Plusquellic, and Y. Jin, "Cyber-physical systems: A security perspective," in *2015 20th IEEE European Test Symposium (ETS)*. IEEE, 2015, pp. 1–8.

- [7] B. Krebs, "Cyber incident blamed for nuclear power plant shutdown," *Washington Post*, June, vol. 5, p. 2008, 2008.
- [8] A. Greenberg, "Hackers remotely kill a jeep on the highway-with me in it," Jul 2015. [Online]. Available: <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>
- [9] M. L. Psiaki, T. E. Humphreys, and B. Stauffer, "GPS lies: Attackers can spoof navigation signals without our knowledge. here's how to fight back," *IEEE Spectrum*, vol. 53, no. 8, pp. 26–53, 2016.
- [10] "Hoegh osaka ship was 'unstable' when it left southampton port," Mar 2016. [Online]. Available: <https://www.bbc.com/news/uk-england-hampshire-35823182>
- [11] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Design & Test*, vol. 34, no. 4, pp. 7–17, 2017.
- [12] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *2008 The 28th International Conference on Distributed Computing Systems Workshops*. IEEE, 2008, pp. 495–500.
- [13] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder, *Diagnosis and fault-tolerant control*. Springer, 2006, vol. 2.
- [14] C. Kwon, S. Yantek, and I. Hwang, "Real-time safety assessment of unmanned aircraft systems against stealthy cyber attacks," *Journal of Aerospace Information Systems*, vol. 13, no. 1, pp. 27–45, 2015.
- [15] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.
- [16] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [17] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [18] W. Wonham, K. Cai, and K. Rudie, "Supervisory control of discrete-event systems: A brief history–1980–2015," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1791–1797, 2017.
- [19] S. H. Kafash, N. Hashemi, C. Murguia, and J. Ruths, "Constraining attackers and enabling operators via actuation limits," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4535–4540.
- [20] C. Murguia and J. Ruths, "On reachable sets of hidden cps sensor attacks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 178–184.
- [21] C. Kwon and I. Hwang, "Reachability analysis for safety assurance of cyber-physical systems against cyber attacks," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2272–2279, 2017.
- [22] L. Wang, Q. Wu, J. Liu, S. Li, and R. R. Negenborn, "State-of-the-art research on motion control of maritime autonomous surface ships," *Journal of Marine Science and Engineering*, vol. 7, no. 12, p. 438, 2019.

- [23] G. C. Kessler, P. Craiger, and J. C. Haass, "A taxonomy framework for maritime cybersecurity: A demonstration using the automatic identification system," *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 12, no. 3, pp. 429–437, 2018.
- [24] B. Croteau, R. Robucci, C. Patel, N. Banerjee, K. Kiriakidis, T. Severson, and E. Rodriguez-Seda, "Alternative actuation paths for ship applications in the presence of cyber-attacks," in *2019 Resilience Week (RWS)*, vol. 1. IEEE, 2019, pp. 91–97.
- [25] N. S. Nise, *Control systems engineering*. John Wiley & Sons, 2020.
- [26] R. Stengel, "Propagation of uncertainty in dynamic systems," <http://www.stengel.mycpanel.princeton.edu/MAE546Seminar17.pdf>, 2018, accessed: 2020-10-22.
- [27] "How to draw ellipse of covariance matrix." [Online]. Available: <https://cookierobotics.com/007/>
- [28] K. Nomoto, T. Taguchi, K. Honda, and S. Hirano, "On the steering qualities of ships," *International Shipbuilding Progress*, vol. 4, no. 35, pp. 354–370, 1957.
- [29] S. Carrillo and J. Contreras, "Obtaining first and second order nomoto models of a fluvial support patrol using identification techniques," *Ship Science and Technology*, vol. 11, no. 22, pp. 19–28, 2018.
- [30] C.-Y. Tzeng and J.-F. Chen, "Fundamental properties of linear ship steering dynamic models," *Journal of Marine Science and Technology*, vol. 7, no. 2, pp. 79–88, 1999.
- [31] M. Chislett and J. Strom-Tejsen, "Planar motion mechanism tests and full-scale steering and manoeuvring predictions for a mariner class vessel," *International Shipbuilding Progress*, vol. 12, no. 129, pp. 201–224, 1965.
- [32] A. M. Lekkas and T. I. Fossen, "Line-of-sight guidance for path following of marine vehicles," *Advanced in marine robotics*, pp. 63–92, 2013.
- [33] N. G. D. Center, "Shoreline/coastline databases: Ncei," Feb 2016. [Online]. Available: <https://www.ngdc.noaa.gov/mgg/shorelines/>
- [34] J. Whitaker, "Matplotlib basemap toolkit documentation." [Online]. Available: <https://matplotlib.org/basemap/index.html>
- [35] "Global ship tracking intelligence: Ais marine traffic." [Online]. Available: <https://www.marinetraffic.com/en/ais/home/centerx:-12.0/centery:25.0/zoom:4>
- [36] "Vessel traffic data." [Online]. Available: <https://marinecadastre.gov/ais/>
- [37] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international journal for geographic information and geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [38] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*. Springer, 1991.
- [39] D. B. Larter, "Here's the darpa project it says could pull the navy a decade forward in unmanned technology," May 2020. [Online]. Available: <https://www.defensenews.com/2020/05/06/heres-the-darpa-project-it-says-could-pull-the-navy-a-decade-forward-in-unmanned-technology/>

A Appendix: San Diego Bay Channel Simulation Code

```

In [ ]: # Simulates driving a ship through the San Diego Bay Channel
# Requires desired path from .csv file, "AIS_Track_355558000_out.csv"
# Outputs "sim_dump.pkl" with simulation parameters and track data.

# import packages
import math
import numpy as np
import matplotlib.pyplot as plt
# https://pandas.pydata.org/
import pandas as pd
# https://pypi.org/project/Shapely/
from shapely.geometry import Polygon
from shapely.geometry import Point
from decimal import Decimal
import random
import _pickle as pickle
import multiprocessing as mp
import time
# https://matplotlib.org/basemap/
from mpl_toolkits.basemap import Basemap
# https://pypi.org/project/rdp/
from rdp import rdp
import string

# Time
tf = 2500

# Plant
T_plant=1.0
t_plant = np.round(np.arange(0, tf, T_plant),2)
Kp = 2.9
Tp = 10.0
L = 160.0
U = 7.7175
K = Kp*U/L
Tr = Tp*L/U
A = np.matrix([[0.0,1],
               [0.0,-1/Tr]])
B = np.matrix([[0],
               [K/Tr]])
C = np.matrix([1, 0])
Phi = np.identity(2) + A*T_plant + 1/2*np.linalg.matrix_power(A,2)*T_plant**2 +
      1/3*np.linalg.matrix_power(A,3)*T_plant**3
Gamma = (np.identity(2)*T_plant + 1/2*A*T_plant**2 + 1/3*np.linalg.matrix_power(A,2)*T_plant**3)@B

# Supervisor
supe = True
baselines = sorted(np.linspace(-35*math.pi/180, 35*math.pi/180, 15), key=abs)
noises = np.linspace(17.5*math.pi/180, 0, 7)
time_horizon = 180
T_supervisor = 5.0
T_search = 30
Phi_supervisor = np.identity(2) + A*T_supervisor
Gamma_supervisor = B*T_supervisor

# Autopilot
kp = 5
ki = 0
kd = 400
T_autopilot = T_plant

# Channel
class Channel:
    def __init__(self, coords): # np array [[x0,y0],[x1,y1], [x2,y2], ...]
        self.polygon = Polygon(coords)
        self.xs = np.array(coords)[: ,0]
        self.ys = np.array(coords)[: ,1]
    def is_land(self, x, y):
        p = Point(x,y)

```

```

        return (not p.within(self.polygon))
    def drawmapboundary(self, *args):
        pass
    def drawcoastlines(self, *args):
        plt.plot(self.xs, self.ys, 'k')
    def fillcontinents(self, color='', lake_color = '',*args):
        pass

# San diego bay
lat_lower=32.6508061321601
lon_left=-117.23953194290745
lat_upper=32.7308041367081
lon_right=-117.11690131547194
bmap = Basemap(projection='merc', resolution='f',
                llcrnrlon=lon_left, llcrnrlat=lat_lower,
                urcrnrlon=lon_right, urcrnrlat=lat_upper,
                suppress_ticks=False)

# Attack
attack_starts = [100, 1050, 2000]
attack_duration = time_horizon
T_attack = T_supervisor

# Path
track_df = pd.read_csv("AIS_Track_355558000_out.csv")
track_x, track_y = bmap(np.array(track_df['LON']), np.array(track_df['LAT']))
track_y = track_y + 50 # shift track up 50m
track = np.concatenate([track_x.T, track_y.T],axis=0).T
path = rdp(track, 100) # Ramer Douglas Peucker Algorithm

# Guidance
delta_h = 200 # Lookahead distance (m)

# General
num_devs = 2
num_ellipse_pts = 100

```

In []:

```

# Saturation Function
def sat(val, val_min, val_max):
    if val < val_min:
        val = val_min
    elif val > val_max:
        val = val_max
    return val

# Convert covariance matrix to ellipse
def matrix_to_ellipse(psi0, r0, P, num_devs, num_ellipse_pts): # 2x2 covariance matrix.
    a = P[0,0]# rad^2
    b = P[0,1]
    c = P[1,1]# rad^2/s^2
    lambda1 = (a+c)/2 + math.sqrt(((a-c)/2)**2 + b**2)
    lambda2 = (a+c)/2 - math.sqrt(((a-c)/2)**2 + b**2)
    if (b==0 and a>=c):
        theta = 0
    elif (b==0 and a<c):
        theta = math.pi/2
    else:
        theta = math.atan2(lambda1-a,b)
    radius_psi = num_devs*math.sqrt(lambda1)
    radius_r = num_devs*math.sqrt(lambda2)
    psi_set = []
    r_set = []
    for t in np.linspace(0,2*math.pi, num_ellipse_pts):
        psi = radius_psi*math.cos(theta)*math.cos(t) - radius_r*math.sin(theta)*math.sin(t) + psi0
        r = radius_psi*math.sin(theta)*math.cos(t) + radius_r*math.cos(theta)*math.sin(t) + r0
        psi_set.append(psi)
        r_set.append(r)
    return psi_set, r_set

```

[illegible]

```

        u_max = sat(u_max, -35*math.pi/180, 35*math.pi/180) # Always the Limit
        break # stop testing noises
    if fits == True:
        break # stop testing baselines
    if fits == False: # Exhausted the search
        x_search_area = []
        y_search_area = []

    else:
        x_search_area = []
        y_search_area = []

# Guidance and Navigation
x_pos1 = path[path_idx,0]
y_pos1 = path[path_idx,1]
x_pos2 = path[path_idx+1,0]
y_pos2 = path[path_idx+1,1]
gamma_p = math.atan2(y_pos2 - y_pos1, x_pos2 - x_pos1)
x_error = (x_pos-x_pos1)*math.cos(gamma_p)+(y_pos-y_pos1)*math.sin(gamma_p)
if x_error + delta_h >= math.sqrt((x_pos2-x_pos1)**2+(y_pos2-y_pos1)**2):
    path_idx = path_idx + 1 # move onto the next segment next time.
y_error = -(x_pos-x_pos1)*math.sin(gamma_p)+(y_pos-y_pos1)*math.cos(gamma_p)
psi_des = gamma_p + math.atan(-y_error/delta_h)
if(psi_des>math.pi): psi_des = psi_des-2.0*math.pi
if(psi_des<-math.pi): psi_des = psi_des+2.0*math.pi

# Autopilot
psi_error = psi_des - psi # calculate heading error
if(psi_error>math.pi): psi_error = psi_error-2.0*math.pi
if(psi_error<-math.pi): psi_error = psi_error+2.0*math.pi
psi_error_sum = psi_error_sum + psi_error
u = kp*psi_error + ki*psi_error_sum + kd*(psi_error-psi_old_error)/T_autopilot # PID
psi_old_error = psi_error

# Attack
if Decimal(str(t)) % Decimal(str(T_attack)) == 0.0:
    u_attack = random.gauss(0, 17.5*math.pi/180)
    attack = False # default
    for attack_start_idx, attack_start in enumerate(attack_starts):
        if t >= attack_start and t < attack_start+attack_duration:
            u = u_attack
            attack = True

# Supervisor Saturation
if supe == True and not aground: # if supervisor is turned on
    u = u + u_base # shift
    u = sat(u, u_min, u_max) # chop
    u = sat(u, -35*math.pi/180, 35*math.pi/180) # Always the Limit

# Save Data
out_df.loc[len(out_df)] = [t,x_search_area, y_search_area, u_min, u_base, u_max,
                           psi,r,x_pos,y_pos,aground,y_error,psi_des,attack,u]

# Plant (where the ship is actually going)
x_pos = x_pos + U*math.cos(psi)*T_plant
y_pos = y_pos + U*math.sin(psi)*T_plant
x = np.matrix([[psi],[r]])
u = np.matrix([u])
x = Phi @ x + Gamma @ u
psi = x[0,0]
if (psi>math.pi): psi = psi - 2.0*math.pi
if (psi<-math.pi): psi = psi + 2.0*math.pi
r = x[1,0]

# Add the dataframe to list
return out_df

```

```

In [ ]: # Multiprocessing
num_sims = 100
tic = time.time()
pool = mp.Pool(mp.cpu_count())

```

```

results = pool.map(job, [x for x in range(0,num_sims)])
pool.close()
toc = time.time()
build_time = toc-tic
print(str(len(results)) + " sims complete in "+str(build_time)+'s')

```

```

In [ ]: # Save Parameters in a dict
params_dump = {
    'build_time':build_time,
    'supe':supe,
    'num_sims':num_sims,
    'path':path,
    'tf':tf,
    'T_plant':T_plant,
    'baselines': baselines,
    'noises':noises,
    'time_horizon':time_horizon,
    'T_supervisor': T_supervisor,
    'T_search': T_search,
    'kp':kp,
    'ki':ki,
    'kd':kd,
    'T_autopilot':T_autopilot,
    'attack_starts':attack_starts,
    'attack_duration':attack_duration,
    'T_attack':T_attack,
    'delta_h':delta_h,
    'num_devs':num_devs,
    'num_ellipse_pts':num_ellipse_pts
}

# Save Frames in a dict
frames = {idx:df for idx,df in zip(range(num_sims), results)}
panel_dump = pd.Panel(frames)

# Save full simulation in dict
sim_dump = {'params': params_dump, 'panel':panel_dump}

```

```

In [ ]: # Save full sim to pickle
with open('sim_dump.pkl','wb') as f:
    pickle.dump(sim_dump, f)

```

```

In [ ]: # Load a full sim from pickle
with open('sim_dump.pkl', 'rb') as f:
    sim_loaded = pickle.load(f)

# Extract Parameters and display in table
params = sim_loaded['params']
print("{:<20} {:<20}".format("Param", "Value"))
for key,value in params.items():
    print("{:<20} {:<20}".format(key, str(value)))

# Extract panel
panel = sim_loaded['panel']

```

```

In [ ]: # Plot all Tracks
# Plot map
WIDTH = 10
HEIGHT = 10
plt.figure(figsize=(WIDTH, HEIGHT))
plt.rcParams.update({'font.size': 22})
plt.grid()
plt.xlabel('x (m)')
plt.ylabel('y (m)')
bmap.drawmapboundary()
bmap.drawcoastlines()
bmap.fillcontinents(color='tan',lake_color='paleturquoise')

```

```

# Plot path
plt.plot(path[:,0], path[:,1], '--k')

# Plot Tracks
for item in list(panel.items):
    out_df = panel[item]
    plt.plot(out_df['x_pos'], out_df['y_pos'])
    #plt.plot(out_df['x_pos'][0:90], out_df['y_pos'][0:90], 'r') # attack areas are red

```

```

In [ ]: # Plot a specific Reachable Set
# Plot map
WIDTH = 10
HEIGHT = 10
plt.figure(figsize=(WIDTH, HEIGHT))
plt.rcParams.update({'font.size': 22})
plt.grid()
plt.xlabel('x (m)')
plt.ylabel('y (m)')
bmap.drawmapboundary()
bmap.drawcoastlines()
bmap.fillcontinents(color='tan',lake_color='paleturquoise')

# Plot path
plt.plot(path[:,0], path[:,1], '--k')

# Plot Specific Track and Reachable Set
item = panel.items[0]
out_df = panel[item]
plt.plot(out_df['x_pos'], out_df['y_pos'])
for idx in out_df.index:
    plt.plot(out_df['x_search_area'][idx],
             out_df['y_search_area'][idx], '.g', markersize=0.5, alpha=0.25)

```

```

In [ ]: # Average Rudders and Limits
WIDTH = 15
HEIGHT = 5
plt.figure(figsize=(WIDTH, HEIGHT))
plt.rcParams.update({'font.size': 22})
plt.grid()
plt.xlabel('Time (s)')
plt.ylabel('Avg Rudder & Limits (rad)')
plt.plot(panel.minor_xs('t')[0], panel.minor_xs('u').mean(axis=1), 'g')
plt.plot(panel.minor_xs('t')[0], panel.minor_xs('u_min').mean(axis=1), 'k')
plt.plot(panel.minor_xs('t')[0], panel.minor_xs('u_base').mean(axis=1), '--k')
plt.plot(panel.minor_xs('t')[0], panel.minor_xs('u_max').mean(axis=1), 'k')
#plt.Legend(['u', 'u_min', 'u_base', 'u_max'])

```

```

In [ ]: # Number of Points on Land
WIDTH = 10
HEIGHT = 5
plt.figure(figsize=(WIDTH, HEIGHT))
plt.rcParams.update({'font.size': 22})
plt.grid()
plt.xlabel('Time (s)')
plt.ylabel('# Points on Land')
plt.plot(panel.minor_xs('t')[0], panel.minor_xs('aground').sum(axis=1))

```