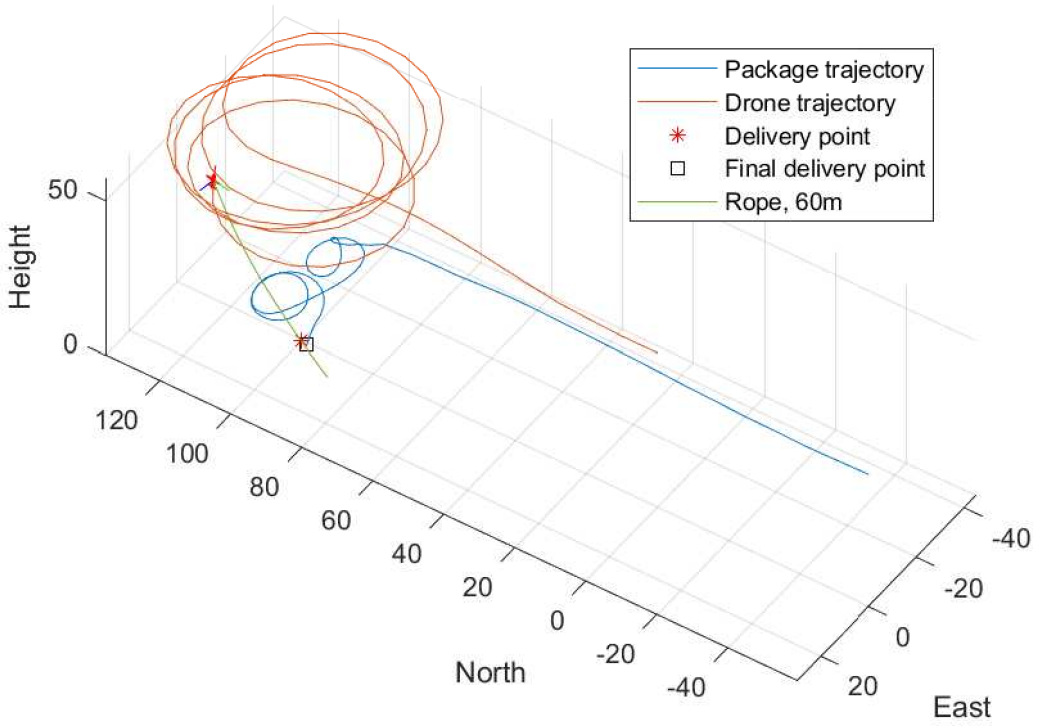

Spiral Drone Delivery

Can a fixed-wing UAV accurately deliver a package over long distances in a sea rescue scenario?

Master's thesis in Systems, Control and Mechatronics

Arvid Käck & Oskar Wångdahl

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Spiral Drone Delivery

Can a fixed-wing UAV accurately deliver a package over long distances in a sea rescue scenario?

Arvid Käck
Oskar Wångdhal

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2022

Spiral Drone Delivery
Can a fixed-wing UAV accurately deliver a package over long distances in a sea rescue scenario?
Arvid Käck, Oskar Wångdahl

Supervisor: Balázs Adam Kulcsár, Department of Electric Engineering
Examiner: Balázs Adam Kulcsár, Department of Electric Engineering

Master's Thesis 2022
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Spiral package delivery simulation in turbulence, with a drone and package trajectory.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Spiral Drone Delivery
Can a fixed-wing UAV accurately deliver a package over long distances in a sea rescue scenario?
Arvid Käck, Oskar Wångdahl
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis presents the design and simulated result of a fixed-wing drone, which follows a spiral trajectory to deliver a package. The thesis presents a mid-fidelity drone model with disturbances generated from wind and rope models. The presented simulations show the effects of a set of different parameters, especially airspeed, the length of the rope, and the orbit radius of the package. These simulations run with three different controllers, PID, LQR, and a non-linear reference tracking feedback linearization controller. The result shows an improvement in delivery accuracy over fly-over parachute drop-deliveries used in current applications. The nonlinear controller shows the greatest potential for continued real-world applications due to its potential for scalability to different drones and disturbance rejection capabilities.

Keywords: UAV, fixedwing, control, simulation, PID, LQR, feedback linearization,

Acknowledgements

We would like to thank Balaáz Adam Kulcsár for his academic guidance, support, and ideas during this thesis and Fredrik Falkman and Svenska Sjörräddnings Sällskapet for their support and encouragement.

Arvid Käck & Oskar Wångdahl, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CDF	Cumulative Density Function
HIL	Hardware in the Loop
LQE	Linear Quadratic Estimator
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
PID	Proportional Integral Derivative
SSRS	Svenska Sjörräddnings Sällskapet
UAV	Unmanned Aerial Vehicle

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

ref	Index for Reference signals
a	Index for Air variables (without wind disturbance)
e	Index for Earth variables (with wind disturbance)
w	Index for Wind variables
p	Index for Package variables
s	Index for Steady state

Parameters

S	Drouge size
L	Length of rope
K	Feedback Control gain matrix
Q, R	Weight matrices
m	Mass

Variables

V	Velocity
γ	Pitch angle
χ	Yaw angle
μ	Roll angle
X	3D-environment position
T	Thrust

D	Drag
L	Lift
α	Angle of attack
dt	Throttle
h	Height
F	Force

Contents

List of Figures

List of Tables

1
Introduction

This chapter presents the background of the thesis. The chapter also includes the research questions that are intended to be answered.

1.1 Background

This thesis is done in collaboration with Svenska Sjöräddningssällskapet (SSRS), who is exploring the application of using remote fixed-wing drones for different tasks at sea. Fixed wing drones are beneficial for their speed and energy efficiency, but lack the ability to hover in place, as they need to remain in motion for generating lift to remain airborne. The commonly used solution for delivery using these kinds of drones uses parachute drops [1, 2], which are largely affected by wind and are inaccurate when dropping in not optimal conditions. And when at sea, where windy conditions are normal, this inaccuracy would prove even worse. As an improvement, SSRS wants to utilize a winch mechanism on the fixed-wing drone instead of using parachutes to achieve more accurate deliveries at sea.
The usage of unmanned aerial vehicles (UAV) is currently used for the delivery of medical supplies to areas that are hard to reach, and this kind of usage is developed by multiple different companies [3, 4, 5]. The problem with these solutions is that they are inaccurate, require a large delivery area, or are implemented using a Quadcopter drone, which is slower, has a smaller operational area, and is less energy-efficient than fixed wing drones [6].
The concept of using a winch mechanism in airdrops was tested in 1969 by US Air Force [7]. This test showed that it was possible to accurately deliver objects by circling the aircraft and quickly ejecting a wire with a package, generating slack for the wire in the middle of the circle whereas a package could be detached or attached to the wire before the aircraft straightens out and flies away. This however required precision from a live aircraft pilot, and further work was deemed needed to automate the process.

1.2 Research questions

This section presents the research questions of the thesis.

- Is it possible to drop a load close to a target from a fixed wing UAV, by using a winch contraption combined with a circling solution?
- How to minimize the predicted landing area of a delivered package during windy weather.

- Is a mechanical winch contraption a good solution for delivering a package in a naval environment?
- What sort of control system is feasible for putting down a load while influenced by error dynamics of flying over open water?
- What measurement/measured data will be required to successfully make the delivery and landing during semi-harsh weather conditions?
- What model complexity is needed to design a model for delivery of a package using a winch mechanism?

1.3 Aim of project

At the end of the project, there should exist simulated results that show if it is deemed possible or not to use a circling solution to deliver a load at a defined target, given an amount of wind disturbance. If the possibility is given, real-life testing will be conducted of the system in flight. This will be limited to the availability of hardware and scaled accordingly. A secondary objective will be to evaluate the possibility to use a similar circling solution for executing a controlled landing near a landing target.

1.3.1 Sub-goals for project

- Create a model of the system
- Design controller for the system
- Closed-loop simulation with error dynamics
- High-fidelity simulation
- Implementation of real world package delivery system
- Implement algorithm for landing plane* (Possible extension for the project if it seems to be a good application of previous discoveries)
- Real world testing (if possible)

1.3.2 Limitations

- This project will mainly focus on drone delivery using winch a mechanism, no other alternatives will be explored.
- The physical test will be done using modified drones from SSRS. No drones will be bought/designed for this project.

1.4 Ethics and sustainability

This thesis is written in collaboration with SSRS with absolutely no military interests in mind. The thesis aims to examine the use of drones as a safe and energy-efficient mode of delivering common trade goods, medical supplies, or rescue materiel. During the last few years, the normalization of the idea to use drones for fast and efficient delivery of common trade goods has become prominent as large companies have begun testing such systems in practice [8]. But this is not the first sighting

of these systems, as drones have for a long time been used in military applications where an autonomous drone can be equipped with anything from military materiel, weapons, or even bombs, So the ethical aspect of improving an algorithm for pinpoint delivery is highly relevant. A long-range drone could be used for military appliances or for nefarious deeds and if the proposed solution works well, a pinpoint delivery system could be exploited for delivering bombs or other bad things. On the other hand, can the same technology be used for quick and efficient delivery of medical supplies, a life jacket, or a defibrillator to a boat in need. From a sustainability aspect, this kind of solution could prove useful as an urban delivery method of different goods. This is relevant due to the increasing demand for product delivery and the decrease of motorized vehicles in urban cities. So utilizing lightweight electric drive-lines proves an interesting possibility for the future of package delivery as no fossil fuels are burned and the drones only need to occupy the airspace so roads and streets can be reserved for other vehicles and pedestrians.

1.5 Data

No data used in this thesis, neither parameters nor generated, cause any harm to people or property. No personal data is used or presented and all data generated during the project is publicly available post publishing.

1.6 Evaluation of results

The primary evaluation of the result from the simulations was primarily the accuracy of the landing location of the package when dropped from hanging on a rope attached to the fixed wing drone. The best possible outcome from this would be to successfully lower the package while circling the target, stabilizing its position to suspension over the target position, and dropping the package on the target position.
The results will then be re-evaluated while simulating different levels of wind and with different simulation parameters, not least of which indicates different builds or configurations of UAV: s. The success factor for each simulation was based on how far off target the package was delivered, where a boat-length or similar length parameter away would be considered a success, and more than the predefined length away from the target would be considered unsuccessful.
After the primary result is evaluated, a set of secondary objectives might be evaluated for further interest and completeness. Some discussed parameters which are of interest are listed below.

- Energy efficiency of flight algorithm
- Speed of delivery
- Possibility to land after a flight mission

2
Theory and Methods

The methodology for this project aims to efficiently implement a basic simulation environment with a simple control method. After some primary testing can be improved to a more complex environment and UAV simulation, with model-based controllers and a delivery method that is based on data from the early simulation results. The later simulations can also be iterated until satisfactory modeling and control of the systems are achieved.

2.1 Modeling

One of the primary parts of this project consisted of creating and integrating different models of both the UAV and environmental factors. At an early stage of the project, a simple model and control of the UAV were constructed to allow for rapid testing. This made it possible to verify if the delivery method seemed plausible and also get a better understanding of which parameters are most important for the flight algorithm and generate ideas for the continued work. After the simple control methods were implemented, the models could be iterated and improved while more complex model-based control methods were implemented.

2.1.1 Rope Model

This work uses information from a previous unpublished thesis that modeled the dynamics of a rope-like object hanging from an aerial vehicle. This model is used for simulating the winch-rope that is attached to the UAV which is the basis for the delivery mechanism proposed in this project. Furthermore, a rudimentary simulation tool for the rope dynamics was already implemented in Matlab which was used for early simulations. Also as long as data from a flight simulation is able to be connected with Matlab in some manner, the rope-simulation tool could also be used while flight simulations were running.
The model handles the dynamics of the rope by dividing it into N parts, where each part is treated as an individual point mass. These point masses are each subject to external forces which allow them to move semi-independently, but there are also constraints put upon the point masses to maintain the properties of the rope or cable modeled. For these point masses, the equation of motions can be modeled for the unconstrained system as follows

$$\mathbf{Ma}(t) = \mathbf{F}(x(t), \dot{x}(t), t) \tag{2.1}$$

Where $\mathbf{F}(t) = (\mathbf{F}_1^T, \mathbf{F}_2^T, ..., \mathbf{F}_N^T)^T$ is the net force acted upon each paint mass, $x(t)$ is the matrix containing the positions for each of the point masses and $\mathbf{a}(t) = (\mathbf{a}_1^T, \mathbf{a}_2^T, ..., \mathbf{a}_N^T)^T$ is the unconstrained acceleration that is given by the equation $\mathbf{a}_i = \mathbf{F}_i(t)/m_i$. The accelerations acting upon the system are then determined from Gauss's principle and the constraints for the system come from the conservation of length between each rope part due to the assumed inflexibility of the rope. The two differentiations of this constraint are used for velocity and acceleration constraints. The last segment of the rope model is modeled as a mass with a drogue. This makes the last segment much more affected by wind and velocity, slowing it down and making it possible to stabilize the package. With this model can the drag in 3D coordinates for the package be calculated using equation 2.2

$$F_{drag} = \frac{\rho_{air} V_p^2 S C_{dD}}{2} \tag{2.2}$$

Where ρ_{air} is the air density, V_p is the velocity vector of the package in a North-East-Down coordinate system with wind disturbance, S is the area of the drogue, and C_{dD} is the drag coefficient for a drogue.

2.1.2 Drone model

The drone model used is based on the Medium-Fidelity Model from Matlabs UAV Toolbox [9]. This model is a good starting point for its ease of use while also including some basic actuator dynamics and functional aerodynamics. The model describes the UAV as a point mass and uses roll, pitch, and throttle as control inputs. The pitch angle and throttle are then converted to an angle of attack and thrust respectively which combined with the roll are the primary control variables. To simulate the roll lag a second-order transfer function was implemented. The complete model is described in equations 2.3.

$$\begin{bmatrix} \dot{V} \\ \dot{\gamma}_a \\ \dot{\chi}_a \\ \dot{X}_e|_N \\ \dot{X}_e|_E \\ \dot{X}_e|_U \end{bmatrix} = \begin{bmatrix} \frac{T\cos\alpha - D - mg\sin\gamma_a}{m} \\ \frac{(L+T\sin\alpha)\cos\mu - mg\cos\gamma_a}{mV} \\ \frac{(L+T\sin\alpha)\sin\mu}{mV\cos\gamma_a} \\ V\sin\chi\cos\gamma_a + V_w|_N \\ V\cos\chi\cos\gamma_a + V_w|_E \\ V\sin\gamma_a + V_w|_U \end{bmatrix} \tag{2.3a}$$

$$T = 4.4482(T_o - 0.047\sqrt{T_o}V) \tag{2.3b}$$

$$T_o = 2.364e^{-7} T_{fact}(2000 + 7200(T_{hk_{min}} + T_{hk}dt))^2 \tag{2.3c}$$

$$D = 0.5\rho S(C_{D_0} + (\frac{(\alpha D_{CL_{da}} + C_{L_0})^2}{A_R \pi})V^2 \tag{2.3d}$$

$$L = 0.5\rho S(C_{D_0} + (\alpha D_{CL_{da}} + C_{L_0}))V^2 \tag{2.3e}$$

$$\alpha = \gamma_r - \gamma_a \tag{2.3f}$$

$$\mu = \frac{5.448}{s^2 + 4.6668s + 5.448}\mu_r \tag{2.3g}$$

Name	Value
A_R	6.9
C_{D_0}	0.022
C_{L_0}	0.38
$D_{CL_{da}}$	18.5
T_{hk}	2.17
$T_{hk_{min}}$	0.0770
T_{fact}	0.2
γ_r	$\in [-15°, 15°]$
dt	$\in [0, 1]$
μ_r	$\in [-90°, 90°]$

Table 2.1: Values of constants and variable parameter thresholds for the drone model

Where V is the drone's airspeed, γ_a is the pitch angle, χ_a is the yaw angle and X_e is the position of the drone vector relative to earth with North-East-Down orientation. T_{fact}, $T_{hk_{min}}$, T_{hk}, C_{D_0}, $D_{CL_{da}}$, C_{L_0} and A_R are drone constants and dt, γ_r and μ_r are input signals for throttle, pitch and roll.

2.1.3 Actuator Modelling

Actuator dynamics are not always modeled in UAV applications as these often are deemed as fast dynamics compared to external factors such as changes in wind speed and turbulence errors [10]. For this project however the actuators might have a large impact, both on their limiting factors as that decides how steep turn rate and how high thrust speeds can be achieved during turns and also how quickly the actuators can make the demanded changes, as that limits the ability of the system to respond to quick decisions which can be crucial for getting to and maintaining a desired orbit pattern.
The first actuator design that was implemented works as a filter between the desired control signal and the response from the UAV model. The filter worked as first-order delays for the flaps and elevator components that control the roll and lift motions respectively. The actuators for the motor throttle generally act with slower dynamics and were modeled with second-order dynamics as these are both complex and might have a large influence on how quick maneuvers can be achieved. The model can be described centralized by equation 2.4 and the matrices described in 2.5.

$$\dot{x}_a = A_a x_a + B_a u_a,\ y_a = C_a x_a,\ x_a = \begin{bmatrix} \delta_e \\ \delta_t \\ \dot{\delta}_t \\ \delta_f \end{bmatrix}\ u_a = \begin{bmatrix} u_e \\ u_t \\ u_f \end{bmatrix} \tag{2.4}$$

$$A_a = \begin{bmatrix} -\frac{1}{\tau_e} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\omega_t^2 & -2\zeta_t\omega_t & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_f} \end{bmatrix} B_a = \begin{bmatrix} \frac{1}{\tau_e} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \omega_t^2 & 0 \\ 0 & 0 & \frac{1}{\tau_f} \end{bmatrix} \tag{2.5}$$

Where the subscripts e, t and f denote elevator, thrust, and flaps respectively. δ is the out-going control signals and u are the input control signals. In the A and B matrices, τ are time constants for the components, and ω and ζ are the oscillation frequency and dampening ratios for second-order dynamics.

2.1.4 Disturbance design

One of the inputs for the drone model is an Environment struct. This struct was designed to contain the wind disturbance for the North, East, and Down coordinates in meters per second, as well as gravitational and air-density constants. The wind disturbance was designed as two different wind models. The first was a turbulence model and the second was a horizontal wind model. These models could work separately or in conjunction with one another to create many different simulation scenarios. The turbulence model used was the "Von Karman Wind Turbulence Model" from the Simulink package "Aerospace blockset" [11]. This model describes the linear and angular velocity as a spatially varying stochastic process. The linear Velocity model is then added to the environment struct. This is calculated using equations 2.6

$$\Phi_{u_g}(\omega) = \sigma_u^2 \frac{2L_u}{\pi V} \cdot \frac{1}{(1 + (1.339L_u\frac{\omega}{V})^2)^{\frac{5}{6}}} \tag{2.6a}$$

$$\Phi_{v_g}(\omega) = \sigma_v^2 \frac{L_v}{\pi V} \cdot \frac{1 + \frac{8}{3}(1.339L_v\frac{\omega}{V})^2}{(1 + (1.339L_v\frac{\omega}{V})^2)^{\frac{11}{6}}} \tag{2.6b}$$

$$\Phi_{w_g}(\omega) = \sigma_w^2 \frac{L_w}{\pi V} \cdot \frac{1 + \frac{8}{3}(1.339L_w\frac{\omega}{V})^2}{(1 + (1.339L_w\frac{\omega}{V})^2)^{\frac{11}{6}}} \tag{2.6c}$$

$$L_w = h, \quad L_u = L_v = \frac{h}{(0.117+0.000823)^{1.2}} \tag{2.6d}$$

Where Φ_i is the linear velocity, L_i is representing the turbulence scale length, σ_i is the turbulence intensity, ω is the circular frequency and h is the altitude of the drone. The subscript u,v, and w represent the longitudinal, lateral, and vertical direction with respect to the aircraft frame.
The horizontal wind disturbance model was implemented using the "Horizontal Wind Model" from the Aerospace blockset [12]. This disturbance is a constant disturbance that is controlled with an angle that defines the direction in the North-East plane it travels. Another disturbance acting upon the UAV is the forces that propagate from the rope swinging about. This was implemented in the simulation by adding the calculated velocity from the drag, lift and gravitational forces acting on the rope together with the turbulence and wind disturbance.

2.2 Control design

This section describes the different control methods used in the project. The main controllers used in the project are a PID controller, an LQE controller with observer feedback, and a nonlinear controller with cumulative error. This section also contains the design of the orbit reference signal and release mechanism for the package.

2.2.1 Package Release Mechanism

The release mechanism was designed to work interchangeably with many iterations of the different simulations and control algorithms. The implementation is made up of two separate algorithms for when the package will be dropped, in which the first one is triggered by a fixed time elapsing in the simulation. This method for dropping the package served as a way to make certain that the package would always be dropped even if a satisfactory drop vector could not be calculated.
The second algorithm is an error-based release mechanism, where the package's landing position is approximated using the current position and velocity of the package. The method aims to ensure that the package lands within a predetermined error margin of the target. This is done by first approximating the time that the package spends in free fall. This is done using the following equation

$$t = -\frac{v_{p|_h}}{2g} + \sqrt{(\frac{v_{p|_h}}{2g})^2 + \frac{h_p}{g}} \tag{2.7}$$

Where $v_{p|_h}$ is the package velocity in the height direction, h_p is the height of the package and g is the gravitational constant. With the time approximated the delivery position p_d can be approximated using the Pythagorean theorem.

$$p_d = t\sqrt{v_{p|_N}^2 + v_{p|_E}^2} \tag{2.8}$$

Combining the distance from the desired delivery position p_p, the angle between the points, and the package position, the relative error ε can be approximated using the law of cosines.

$$\varepsilon = \sqrt{p_d^2 + p_p^2 - 2p_d p_p \cos(\theta_{error})} \tag{2.9}$$

2.2.2 Orbit follower

To find a desired trajectory for making the UAV circulate the desired point, an orbit following algorithm was implemented in the simulation. This was achieved using the orbit follower from Matlabs UAV toolbox [13]. This function uses the drone's position, a point of interest, and a radius to generate a desired course for the drone. This course is set by the parameters for turn direction and a look-ahead distance, which describes how far along the desired path the UAV plans when calculating the angular velocities. The desired course is defined as a heading angle and can be converted to a roll angle reference by utilizing the equation 2.10

$$\mu_{ref} = \arctan\left(\frac{P_{head}(\chi - \chi_{ref})(-b + \sqrt{b^2 - c})}{g\cos(\chi - \Psi)}\right) \tag{2.10a}$$

$$b = \begin{bmatrix}\cos(\chi)\cos(\gamma), & \sin(\chi)\cos(\gamma), & -\sin(\gamma)\end{bmatrix} V_w \tag{2.10b}$$

$$c = V_w^T V_w - V^2 \tag{2.10c}$$

Where P_{head} is a control gain, χ is the heading angle, χ_{ref} is the target heading, Ψ is the yaw angle, γ is the flight path angle and V_w is the wind velocity vector.

2.2.3 PID controller

The early model was controlled by a set of two PI controllers which controls the height and roll inputs. The velocity of the UAV was then controlled with a PID controller. This controller setup successfully stabilized the UAV and handled the flight during full missions successfully, but not necessarily in an optimal way. The non-optimal control was especially noticeable when a turbulence model is added, where the PID regulators react slowly to the added disturbance, which sometimes means that the UAV veers off its course by a significant margin.

2.2.4 LQR with observer feedback

The second controller developed was a linear quadratic regulator (LQR) with an observer that makes an estimation of the states. The LQR-controller is designed by linearizing the plant model (2.3) at a steady state to create a linear state space model. From the linear model, it is then possible to synthesize an observer which estimates the system states. This is compared with the system output parameters and fed into a linear quadratic estimator (LQE) and its output is returned to the model by adding it to the derivative of the state estimate. The compensation gain that is generated from this is simultaneously fed through an LQR, added to the observers and to the plant model's input signals. This process can be seen in the figure 2.1 and is further explained in equation (2.11)

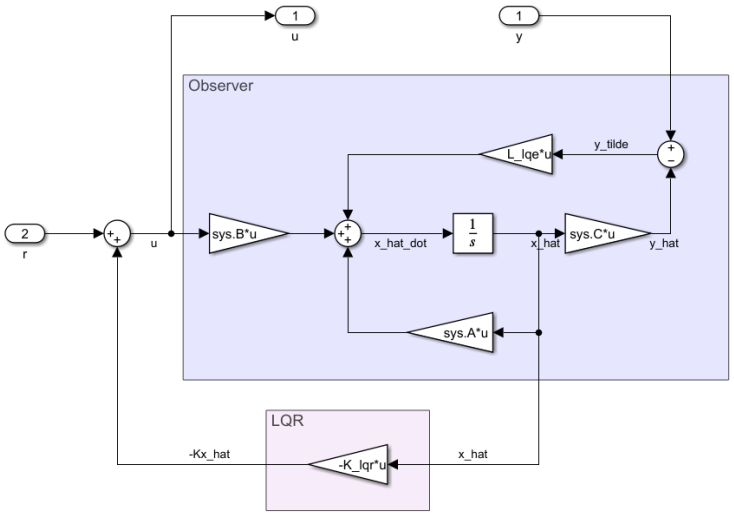

Figure 2.1: A systematic overview of LQR with observer estimation

Herewith, we use a Luemberger observer as

$$\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + B\mathbf{u} + L(\hat{\mathbf{y}} - \mathbf{y}) \tag{2.11a}$$

$$\hat{\mathbf{y}} = C\hat{\mathbf{x}} \tag{2.11b}$$

$$\mathbf{u} = \mathbf{r} - K\hat{\mathbf{x}} \tag{2.11c}$$

Where K is the optimal control feedback gain calculated using the continuous-time Riccati equation [14].

2.2.4.1 Defining the steady state

To define a steady state for the drone model was the function trim in MATLAB used[15]. This function takes a Simulink model and finds the closest steady state near a chosen initial state. By utilizing this trim algorithm, the plant model (2.3) could be linearizable by changing equation (2.3g), by adding the derivative of μ as a state, to

$$\begin{bmatrix} \dot{\mu} \\ \ddot{\mu} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -5.448 & -4.6668 \end{bmatrix} \begin{bmatrix} \mu \\ \dot{\mu} \end{bmatrix} + \begin{bmatrix} 0 \\ 5.448 \end{bmatrix} \mu_r \tag{2.12}$$

With this change the steady state after trim becomes

$$\begin{aligned} x_s = \begin{bmatrix} V \\ \gamma_a \\ \mu \\ \dot{\mu} \end{bmatrix} = \begin{bmatrix} 15.9782 \\ 0.0238 \\ 0.7932 \\ -1.2264e^{-22} \end{bmatrix} \quad u_s = \begin{bmatrix} \gamma_r \\ dt \\ \mu_r \end{bmatrix} = \begin{bmatrix} 0.0131 \\ 0.2498 \\ 0.7932 \end{bmatrix} \\ y_s = \begin{bmatrix} V \\ \gamma_a \\ \mu \end{bmatrix} = \begin{bmatrix} 15.9782 \\ 0.0238 \\ 0.7854 \end{bmatrix} \quad \dot{x}_s = \begin{bmatrix} 4.5080e^{-10} \\ -2.1312e^{-13} \\ -1.2264e^{-22} \\ 1.7764e^{-15} \end{bmatrix} \end{aligned} \tag{2.13}$$

Which by using MATLAB's linmod function gives the state space described in equation (2.14).

$$\begin{aligned} A = \begin{bmatrix} -0.3727 & 13.8960 & 0 & 0 \\ 0.0771 & -62.9352 & -0.6144 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -5.4480 & -4.6668 \end{bmatrix} \\ B = \begin{bmatrix} -23.7132 & 15.5413 & 0 \\ 62.9498 & -0.0074 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 5.4480 \end{bmatrix} \\ C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ D = \mathbf{0} \end{aligned} \tag{2.14}$$

With this the controller gain could be calculated, using the Riccati equation with the following weight matrices

$$\begin{aligned} K_{lqr} = \begin{bmatrix} -0.5192 & 0.3160 & 5.7751e^{-04} & 1.0995e^{-04} \\ 0.8321 & 0.1852 & -0.0080 & -3.4378e^{-04} \\ -1.2053e^{-04} & -3.5888e^{-05} & 0.4143 & 0.5167 \end{bmatrix} \\ Q = \mathbf{I}_4 \\ R = \mathbf{I}_3 \end{aligned} \tag{2.15}$$

2.2.5 Nonlinear control

The final controller that was tested in this thesis was a nonlinear controller with cumulative error reference tracking. Similar to the LQR controller the nonlinear controller also aims to linearize the model behavior and run the state feedback through a control filter to use as the input signal to the system. The difference between the systems is that where the LQR aims to find a steady state around which the model can be linearized around. The nonlinear feedback linearization instead utilizes the controller feedback to directly influence the input signal to the system to cancel out the nonlinearities that the system needs to handle. This retains the nonlinear dynamics of the system while from the outer loop, the function of the system appears linear in its dynamics, which makes it possible to apply a linear controller to the system.

2.2.5.1 Feedback linearization

The system was linearized by changing the input signal to account for the nonlinear modes, this is represented in equation (2.16).

$$u = \alpha(x) + \beta(x)v \implies v = \frac{u - \alpha(x)}{\beta(x)} \tag{2.16}$$

Where α and β are functions that cancel out the nonlinearities of the state and v is a new input signal to the linear system described in equation (2.17)

$$\dot{x} = A_c x + B_c v \tag{2.17}$$

Applying the system above to the state space that was implemented before, some changes and approximations were required to be made. The first one was an approximation of the thrust to remove the dependence of the speed. Changing it to

$$T = 4.4482(T_o - 0.047\sqrt{T_o}V) \approx 4.4482T_o \tag{2.18}$$

This change also made it possible to change throttle input to a thrust input. Another approximation that was applied was the small-angle approximation on all angle inputs. This was done in order to simplify the calculations and avoid unnecessary errors because the use of the inverse of sinus, cosine, and tangent might in some cases lead to the generation of imaginary numbers. The last approximation done to the system was to fixate the reference roll signal and the actual roll was the same. With these changes, the system was linearized using function (2.19) as the new input signal.

$$\begin{bmatrix} \gamma_{ref} \\ T_{o_{ref}} \\ \mu_{ref} \end{bmatrix} = \begin{bmatrix} \frac{Vm}{11.1273\cos(\mu)V^2 + T\cos(\mu)} \left(v_\gamma + \frac{0.6015\cos(\mu)(18.5\gamma - 0.38)V^2 + W\cos(\gamma) + T\gamma\cos(\mu)}{Vm} \right) \\ \frac{0.2248\,(D + W\,\gamma_a + m\,v_V)}{\cos(\alpha)} \\ \frac{V\,m\,v_\mu\,\cos(\gamma_a)}{L + T\,\sin(\alpha)} \end{bmatrix} \tag{2.19}$$

where v_γ, v_V and v_μ is the new input signals for the linear system described in (2.20)

$$\begin{bmatrix} \dot{V} \\ \dot{\gamma}_a \\ \dot{\chi}_a \end{bmatrix} \approx \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A_c} \begin{bmatrix} V \\ \gamma_a \\ \chi_a \end{bmatrix} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{B_c} \begin{bmatrix} v_V \\ v_\gamma \\ v_\mu \end{bmatrix} \tag{2.20}$$

2.2.5.2 Nonlinear controller design

When the system is linearized an LQR can be applied for control of the system. This controller is similar to the one used in section 2.2.4, but uses the error and cumulative error as states to determine the input signal v, shown in figure 2.2. Just as in the LQR case, the controller gain is calculated using the Riccati equation. The controller gain can be seen in (2.21).

$$\begin{gathered} K_{non} = \begin{bmatrix} 1.0954 & 0 & 0 & 0.1000 & 0 & 0 \\ 0 & 1.7321 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1.7321 & 0 & 0 & 1 \end{bmatrix} \\ Q = \mathrm{diag}([1, 1, 1, 0.01, 1, 1]) \\ R = \mathbf{I}_3 \end{gathered} \tag{2.21}$$

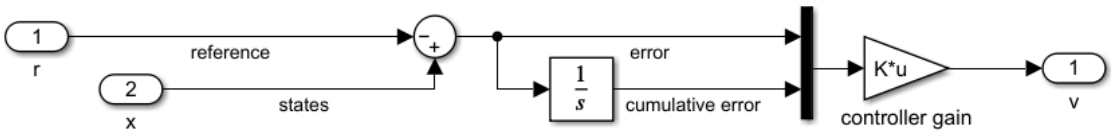

Figure 2.2: Nonlinear controller with cumulative error

To assure a more robust system and remove unwanted values, the input to the nonlinear system u is saturated as follows.

$$\gamma_r \in [-\tfrac{\pi}{12},\ \tfrac{\pi}{12}], \quad T_{o_r} \in [0.3085,\ 15.6215], \quad \mu_r \in [-\tfrac{\pi}{2},\ \tfrac{\pi}{2}] \tag{2.22}$$

LQR

3
Results

The results of this project consist of the data generated from all of the different simulations ran during the thesis. In this chapter, this data will be accounted for in various plots and collections to make it easy to compare and analyze.
Note that the results from the early simulations were used to influence how the model-based design and controllers were developed. This means that the development of the LQR was influenced by the PIDs early performance and the nonlinear feedback controller was influenced by both the PID and LQR.

3.1 Simulation setup

In this section, we describe what parameters are tested and what setups will be used for each simulation. To test the effectiveness of different controllers developed in the project, a series of tests were conducted to generate data that can be used to see the difference in performance between the control algorithms. The data that were compared are presented below.

- Package radius
- Roll angle
- The speed of drone
- The length of rope
- Effect of wind strength
- Size of the drouge

3.1.1 Straight fly-by drop

Acting as a point of comparison, the first simulations to be conducted were performed by letting the drone fly towards the delivery point and drop the package at full flight speed. This acted as a first result that was used as a baseline for further simulation results. When running the simulation with some added wind disturbance the fly-by drop performs poorly as the consistency of the delivery performance is low and the positional error is big, up at a 25m offset. The performance of the fly-by drop can be seen in figure 3.1, Where the result of the simulations is presented as cumulative distribution functions of the delivery error, meaning that it shows the percent of successful deliveries probability of getting an error or smaller than the selected value.

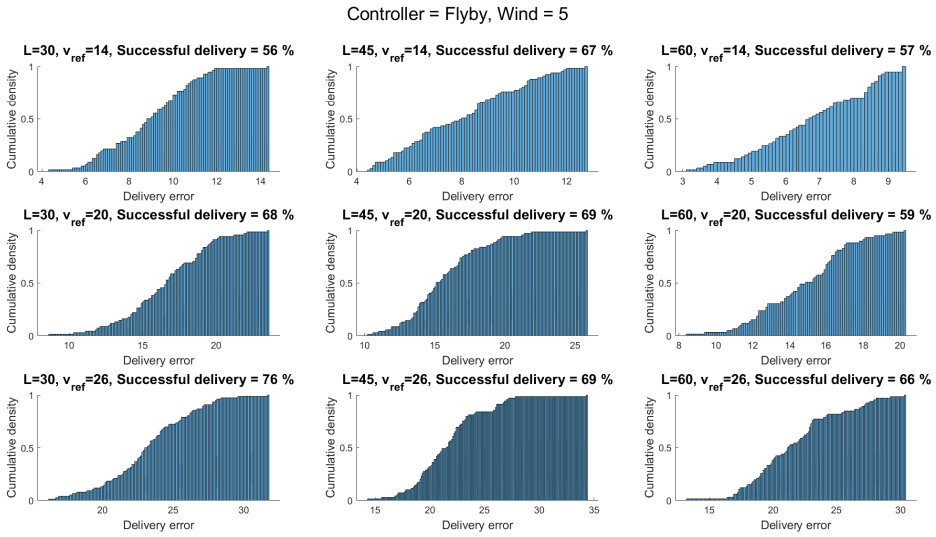

Figure 3.1: Cumulative density function plot of the fly-by drop performance test

3.1.2 Drop in turbulence

To test the drop accuracy of the controller were each controller tested to deliver a package to a specific point. This was simulated with 3 different velocities and 3 different rope lengths 100 times, to see the effect of these parameters and get a reliable result. To make it more likely to find a valid drop vector, the target point reference fed into the orbit follower will shift one meter along the east axis every 15 seconds. Meaning that if the system finds a stable orbit trajectory with a radius that is larger than the drop threshold, this trajectory will move along the east axis until it is in a best-case scenario intersects with the target. The result can be seen in table A.4 as well as figures 3.4, 3.7, and 3.10 in the appendix. In this test, the drop is classified as a failed delivery if the time-based release mechanism is activated, which is activated after 190 seconds. This was implemented to make sure the delivery is efficient and does not take too long. This test was done without horizontal wind disturbance activated so that the wind strength only affects the turbulence. An example of how the simulation looks while running is shown in figure 3.2.

3.1.3 Drogue size

The drogue generates the largest lift and drag disturbance on the rope and this is used to anchor the package during its orbit. This made it interesting to perform testing on the effect of the drogue size and this was done with added turbulence. The different drogue sizes were also tested at different speeds since this is a key variable in the drag equation 2.2. The result of this can be seen in figure 3.3 as well as figure C.24 and C.25 in the appendix. The result shows that attaching the drogue significantly decreases the package's orbit radius. But on the other hand, the result also shows that increasing the size of the drogue only has a small influence on further lowering the package radius. The result also shows that having a large drogue makes the package's flight trajectory less sporadic.

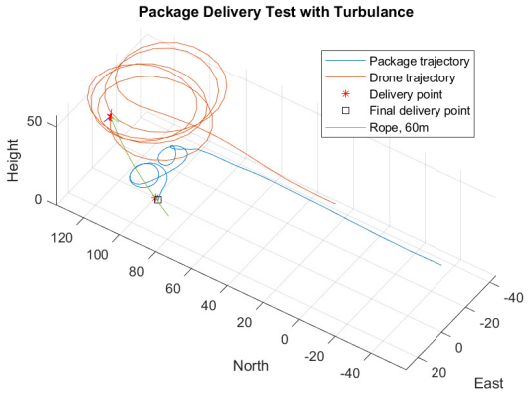

Figure 3.2: Trajectory of the drone and position of the package during simulated flight. Delivery position marked as a star from the delivery test

3.1.4 Package radius

To test the accuracy of the controllers and the effect of different rope lengths and velocities affecting the system was a package radius test carried out for each of the controllers. This test was done in an optimal (windless) environment where the drone was circling a predefined point with different values for the rope length and velocity reference. The complete result can be seen in table A.1 - A.3 in the appendix and in figures 3.5, 3.8 and 3.11.

3.1.5 Minimum and maximum package-radius in wind

One important aspect of the drone controller is its ability to handle wind disturbance effectively. The drone was simulated in different wind strengths to test the effect of turbulence on the controllers. The test included different lengths of the rope and drone speeds to see how these parameters affect the performance. This simulation ran 15 times for each combination of parameters to increase the reliability of the results. The result was outputted as the average maximum and minimum package radius, as well as maximum and minimum UAV flight radius. The result is shown in figures C.6 - C.23, tables A.5 and A.6 in the appendix. This test was done without side wind disturbance activated.

3.1.6 Side wind

To test the drone's susceptibility to side wind, a drop simulation with side wind was performed. The test was carried out to attest the different controllers' ability to act on and cancel out disturbances in constant side wind and turbulence. The test would also see how different velocities and rope lengths affected the accuracy of the delivery. For each parameter combination was the simulation repeated 30 times to ensure a more reliable result. The result of this test can be observed in figures 3.6, 3.9, 3.12 and table A.7 in the appendix.

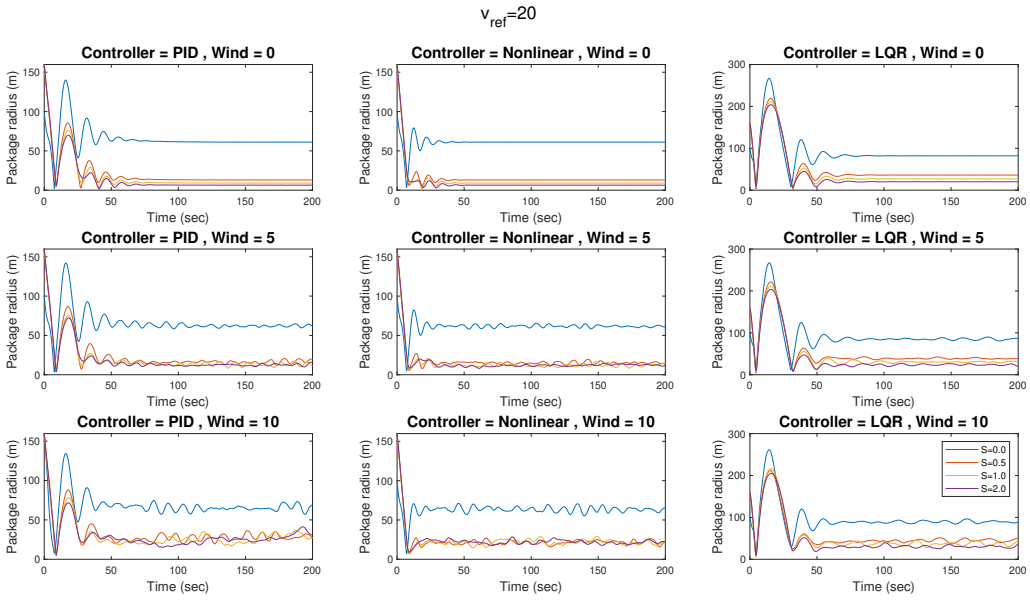

Figure 3.3: Package radius over time different controllers and drouge sizes when $v_{ref} = 20$

3.2 Controllers performance

In this section, the result of the different drop and package radius tests are presented. It also describes the result for each controller in the different test configurations.

3.2.1 PID Controller

The PID controlled system performed well as it often found a good possibility to drop the package and the trajectory of the drop is often accurate as can be seen in figure 3.4. As long as the wind gust disturbance is not too large to be handled by the PID controller, it is almost always successfully set into a controlled circling motion around the target position. As the PID controller is quite slow to react to the disturbance the circling motion does not converge to perfect circles but stays in the general area of the target position. This sporadic behavior can sometimes be used advantageously as the irregular movement sometimes creates an opportunity for a good delivery vector. This is also seen in that the time for delivery is higher than with the more refined control methods. It can also be seen in figure 3.4 that some combination of velocity and rope length has a very low successful drop. Meaning that the controller didn't find a drop chance where the error approximation was lower than a threshold of 2 meters error, leading to a time-based release instead of an error-based release. This can be seen as a more stair like behavior in the CDF plot or an empty plot if no delivery was successful, as well as a "Unsuccessful" in table 3.1.
In figure 3.5 and table A.1, it can be seen that it is possible to achieve a small circling radius for the package with a long rope while keeping the speed as low as possible. It is also possible to compensate for a higher speed with a longer rope to achieve a similar result. In an environment without wind disturbance the PID controller

gets results comparable to the nonlinear feedback controller, but with more sporadic control signal inputs.

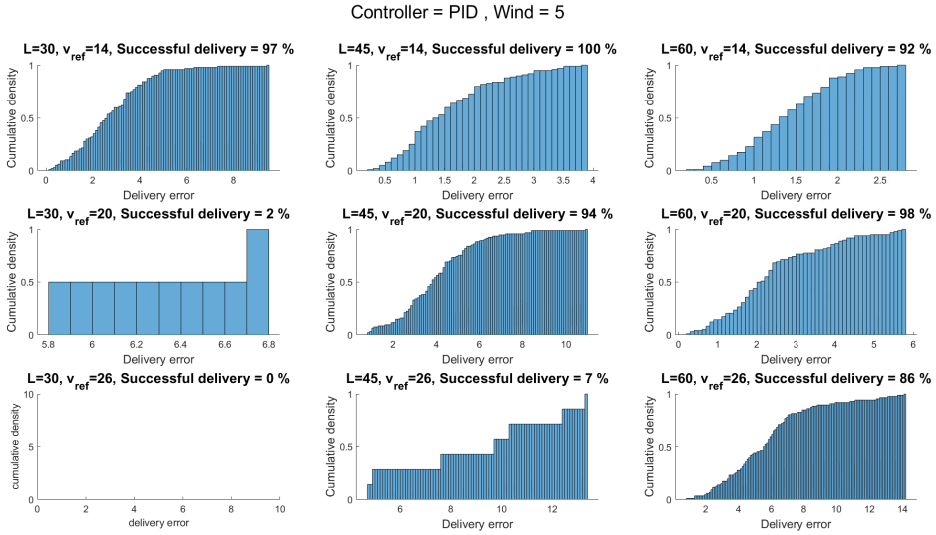

Figure 3.4: CDF plots of PID controller drop-performance. Empty plot indicates no successful activations of the error-based drop algorithm

Table 3.1: Data from delivery test with PID controller

Model	Speed reference (m/s) and rope length (m)	Mean delivery time (sec)	Mean delivery error (m)	Successful activation during test
PID	$v_{ref} = 14$, L = 30	84.6392	2.7780	97%
	$v_{ref} = 14$, L = 45	69.6500	1.5634	100%
	$v_{ref} = 14$, L = 60	82.9457	1.4123	92%
	$v_{ref} = 20$, L = 30	163.0000	6.3122	2%
	$v_{ref} = 20$, L = 45	100.3298	3.9953	94%
	$v_{ref} = 20$, L = 60	70.8265	2.3876	98%
	$v_{ref} = 26$, L = 30	Unsuccessful	Unsuccessful	0%
	$v_{ref} = 26$, L = 45	168.2857	9.0199	7%
	$v_{ref} = 26$, L = 60	99.3721	5.7401	86%

In the minimum/maximum package-radius test in wind, from table A.6, it can be observed that the PID controller sees the largest change in maximum radius, while simultaneously keeping a small average radius. This points toward the PID controller having a hard time handling large amounts of turbulence. Comparing it to the test with smaller turbulence in table A.5 where the maximum and average maximum is closer indicates a more stable orbit pattern.
The side wind test shows that the PID is the worst controller at handling wind disturbance. This can be seen in the increase in delivery error in figure 3.6 compared

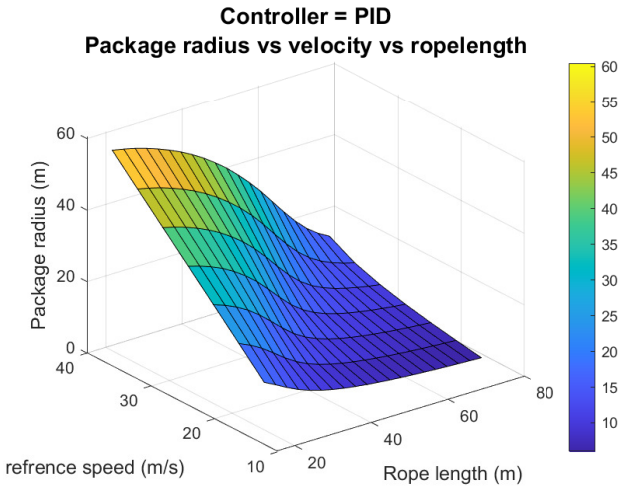

Figure 3.5: PID controller package radius performance

to 3.4, where the mean delivery error is doubled or tripled for almost all parameter configurations while side wind is activated. The result also shows a decrease in successful deliveries with increased disturbance.

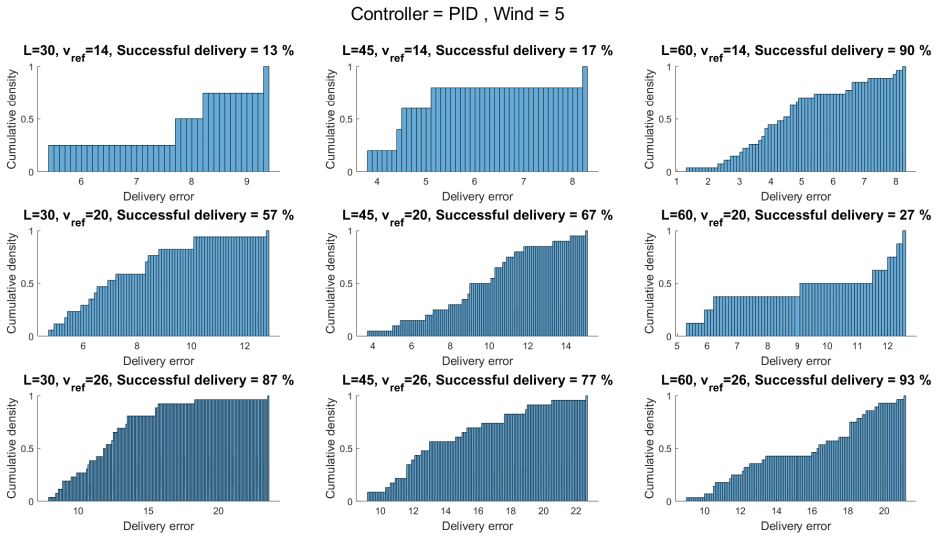

Figure 3.6: CDF plots of the PID controller drop-performance in the side wind test

3.2.2 Linear quadratic regulator

The LQR is the worst performing controller designed for this project. The regulator successfully allows the UAV to enter circulation of the target but compared to the PID controller and the nonlinear feedback controller the performance is not good. In figure 3.7 and table 3.2, it can be seen that the LQR has the largest delivery error and lowest successful delivery drop activation count compared to the other controllers. The circling radius where the system finds a steady-state is often large

and the time it takes for the circling to stabilize takes longer than for the PID system. It can also be seen in figure 3.7 that the LQR controller can is unable to deliver when the velocity is high. This is because the circulation radius of the drone is much larger, leading to the package never getting close to the target point and the error release for the package never activates. A reason for the long time for stabilization can be described by the error dynamics bringing the UAV too far away from the linearization point and after that, cannot react quickly enough to maintain the steady state in a tight circular motion.

Something positive that can be said about the LQR development is that it successfully finds linearization points for most of the test cases. This means that at least there exist stabilizable modes in the system, which in itself is a result that could be recognized and taken into consideration for the further development of even more sophisticated control methods, such as the Nonlinear feedback controller.
The package radius test result in figure 3.8 shows that the LQR controller has the largest package radius out of all the controllers. One thing to note is that the velocity affects this controller more than any of the controllers and this is probably because the higher velocity makes the UAV diverge further from the steady state used in the design of the controller.

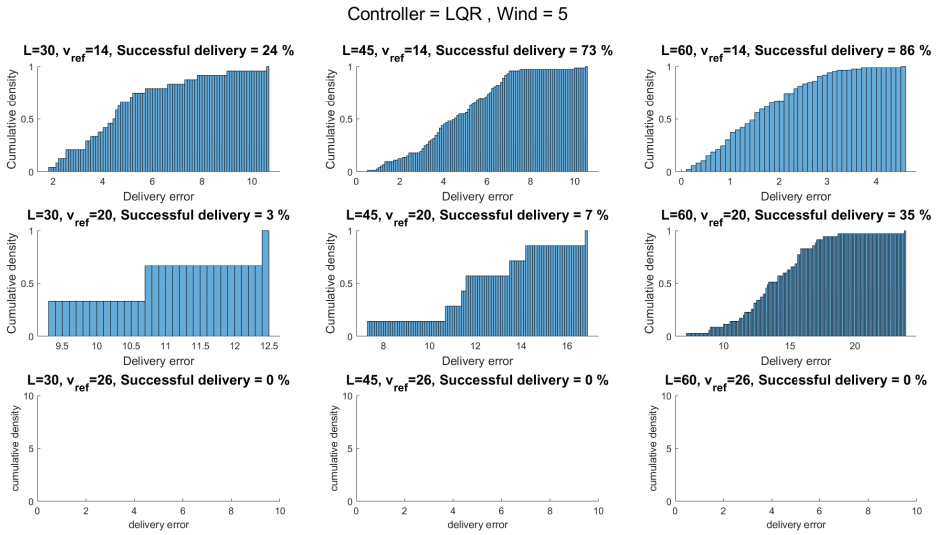

Figure 3.7: CDF plots of LQR controller drop-performance. Empty plots indicate no successful activations of the error-based drop algorithm

Table 3.2: Data from delivery test with LQR controller

Model	Speed reference (m/s) and rope length (m)	Mean delivery time (sec)	Mean delivery error (m)	Successful activation during test
LQR	v_{ref} = 14, L = 30	35.4583	4.7739	24%
	v_{ref} = 14, L = 45	56.3288	4.5818	73%
	v_{ref} = 14, L = 60	93.5930	1.6051	86%
	v_{ref} = 20, L = 30	31.0000	10.8761	3%
	v_{ref} = 20, L = 45	31.0000	12.2964	7%
	v_{ref} = 20, L = 60	31.0286	13.9243	35%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 60	Unsuccessful	Unsuccessful	0%

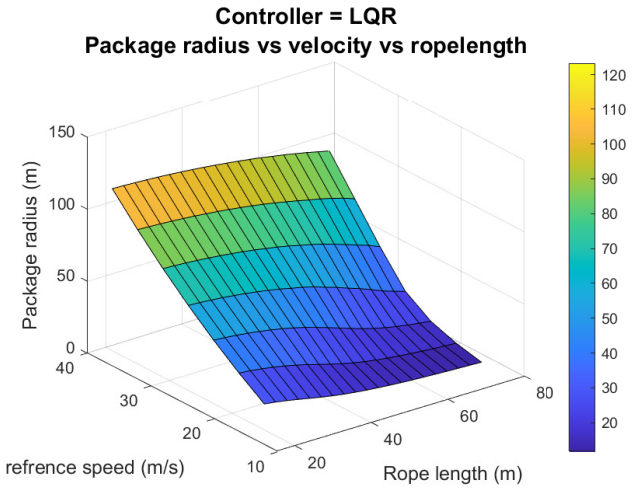

Figure 3.8: LQR controller package radius performance

The LQR performance in the minimum/maximum package radius test is shown in tables A.5 and A.6 shows that the controller is able to keep a stable package radius even in heavy turbulence. It can also be seen that the LQR flies with the largest orbit radius and is often not able to close in on the desired point. This can especially be seen in the simulations with reference speed v_{ref} = 26 m/s where the average minimum radius is high compared to the other controllers.
The side wind test shows that the LQR is able to handle side wind with a small increase in delivery error. This test also shows that the LQR controller has a hard time finding a suitable drop point, indicated by the increase in failed deliveries.

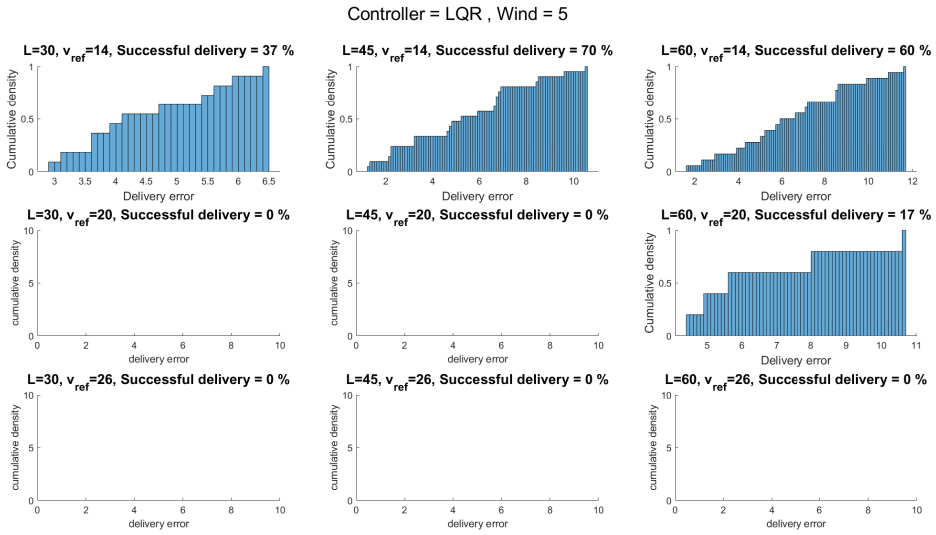

Figure 3.9: CDF plot of the LQR controllers drop-performance in side wind test. Empty plot indicates no successful activations of the error-based drop algorithm

3.2.3 Reference tracking feedback linearization

The nonlinear controller performs the most consistently of the three tested. In figure 3.10 and table 3.3 it can be seen that the controller successfully finds a stable mode of operation in more than 90% of simulations running with advantageous parameters. It can also be seen that similar to the PID is the nonlinear controller unable to activate the error-based release if the drone has a high speed and short rope. The feedback loop then successfully minimizes the error dynamics given a long enough rope length to facilitate the package orbit and a velocity that is slow enough to favor control. This result can be seen as rather stable because when parameters are favorable the system most always succeeds in its mission and when parametric factors are not good enough, the system will mostly not be able to find a stable mode of operation to achieve a successful drop. Another advantage that can be seen in the results from the simulations is that the nonlinear controller generally stabilizes more quickly in comparison to the PID and keeps a smaller delivery error compared to the LQR, meaning that it often only requires a few circling laps before it can find a possible delivery vector.
For the package radius test, the nonlinear feedback controller got similar but slightly better results compared to the PID controller. In figure 3.11 and table A.3 it can be seen that the controller can compensate for a higher speed with the use of a longer rope. This might allow for the possibility to re-scale the drone parameters and still maintain a similar result.

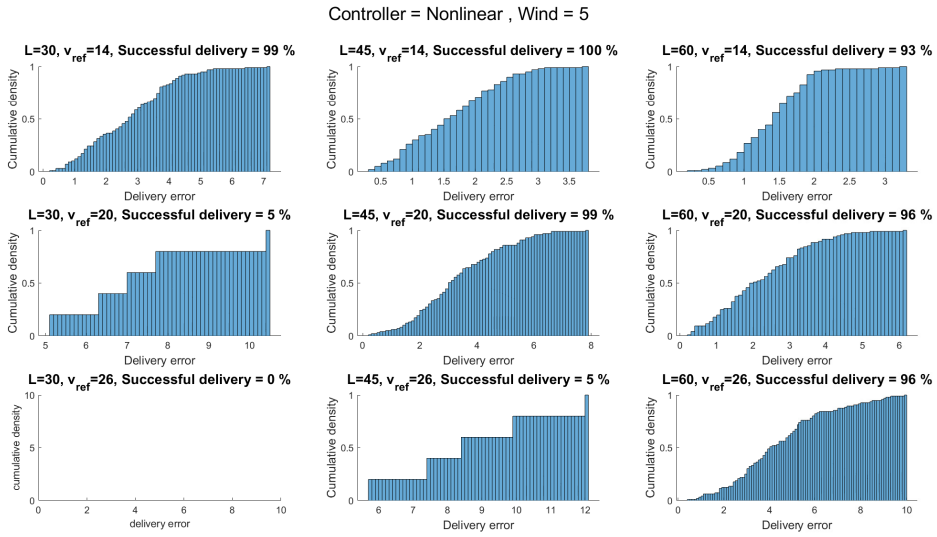

Figure 3.10: Nonlinear feedback controller drop-performance. Empty plot indicates no successful activations of the error-based drop algorithm

Table 3.3: Data from delivery test with nonlinear feedback controller

Model	Speed reference (m/s) and rope length (m)	Mean delivery time (sec)	Mean delivery error (m)	Successful activation during test
Nonlinear	v_{ref} = 14, L = 30	69.5556	2.7417	99%
	v_{ref} = 14, L = 45	58.9300	1.6316	100%
	v_{ref} = 14, L = 60	63.5269	1.4343	93%
	v_{ref} = 20, L = 30	161.6000	7.3453	5%
	v_{ref} = 20, L = 45	79.1818	3.3324	99%
	v_{ref} = 20, L = 60	62.5833	2.2411	96%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	178.2000	8.7469	5%
	v_{ref} = 26, L = 60	96.3646	4.4416	96%

The minimum/maximum test result of the nonlinear feedback controller in table A.5 shows that the nonlinear controller is able to handle moderate wind and maintain a small radius that does not diverge much. From the table A.6 it can be observed that the maximum radius increases with the increased wind, but the difference between the maximum and average values remain close, meaning that the orbit is kept stable.

The result of the side wind test for the nonlinear controller can be seen in figure 3.12. In this figure it can be seen that with the added disturbance, the package radius error increases by about 1-2 meters compared to the results from the drop test in figure 3.10. This also shows that the number of failed delivers increases for certain parameter combinations, but the nonlinear controller still achieves a high drop success chance.

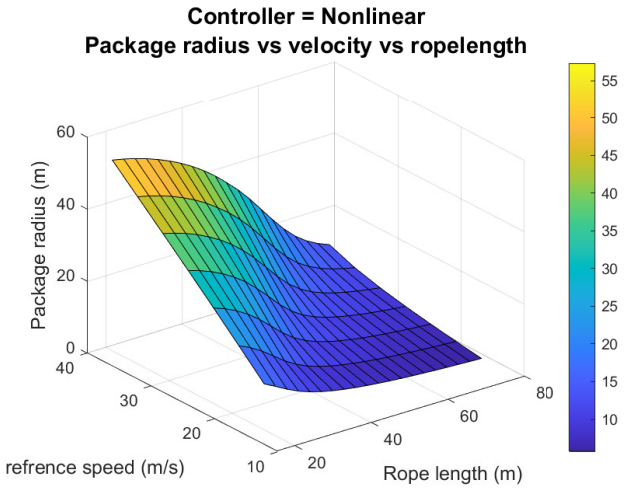

Figure 3.11: Nonlinear package radius performance

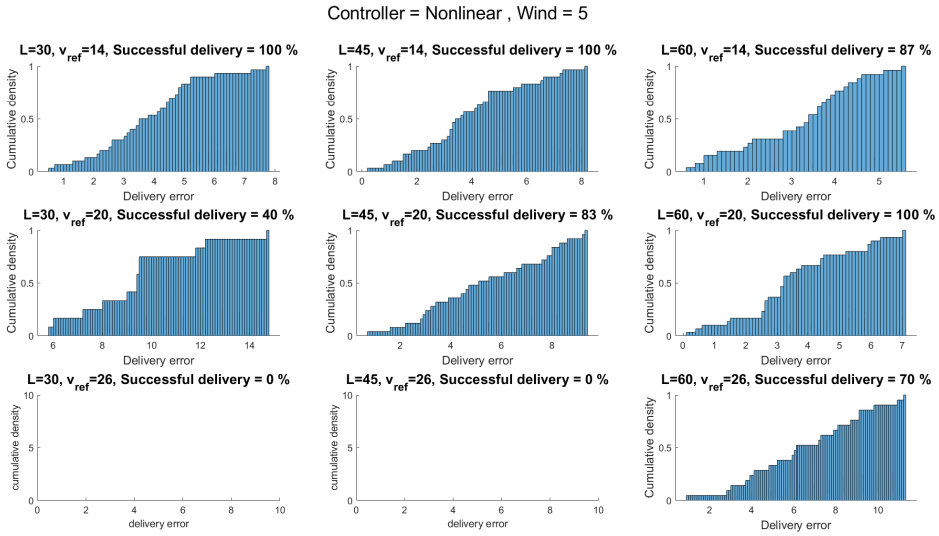

Figure 3.12: CDF plots of the Nonlinear controller drop-performance in side wind testing. Empty plot indicates no successful activations of the error-based drop algorithm

3.2.4 Control signals

The final test was run to monitor the control signals. This test was executed by running a simulation for each of the different controllers and letting the drone find an optimal orbit around a specific point. The result of these tests can be seen in figures 3.13-3.15 and shows the control signal over time. It can be seen that the PID has a noisy control signal, which changes with a high frequency, especially for the throttle. Meanwhile, the LQR and nonlinear controllers have more constant signals. It can also be seen that the nonlinear controller has the most smooth signal. It can also be seen that all the roll signals change quite drastically before the signal

becomes constant. This happens when the drone flies over the delivery point.

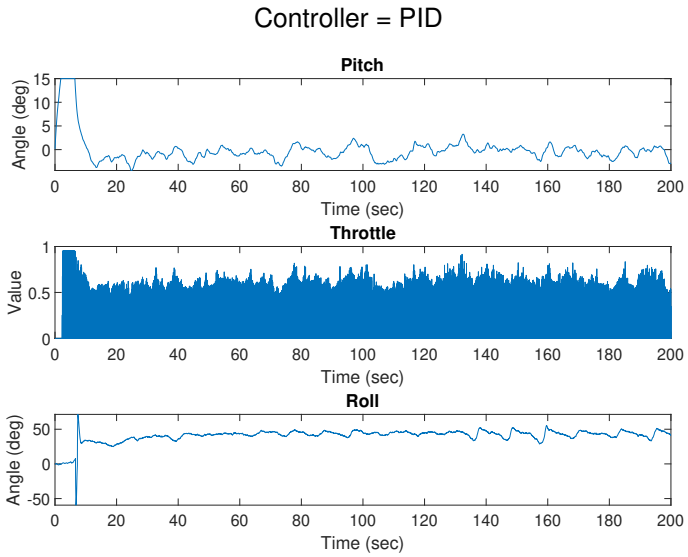

Figure 3.13: Control signals for PID controller

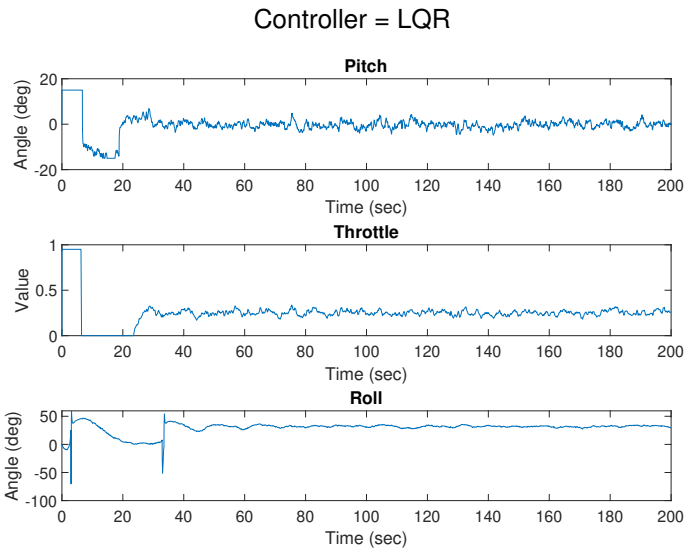

Figure 3.14: Control signals for LQR controller

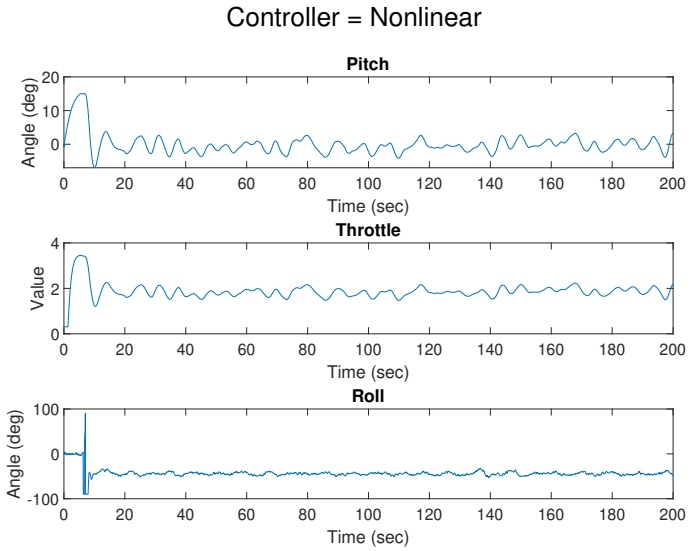

Figure 3.15: Control signals for Nonlinear controller

4
Discussion

This chapter presents the discussion of the result and conclusion of the thesis. The discussion is divided into different main areas of the thesis. Where the last section will cover some potential future development springing from this project.

4.1 Delivery release mechanism

The delivery release mechanism is a basic algorithm that can handle a simple drop with relatively good accuracy. It can be seen in the figures C.3-C.5 that with a spiral flight path the drone has the ability to deliver a package with a mean under the desired error threshold if the rope length and velocity are adequately chosen. But there are still improvements that can be made to the algorithm to make it more plausible to be used in a real-life environment. One is that the package starts to drop immediately after the release activates. In a real-life system, there will be a delay for when the system releases the package. Another improvement is that the algorithm is not currently able to approximate the effect of the wind in the drop calculations, leading to a larger delivery spread.
The drop area calculation should also be able to receive some improvements since the algorithm now assumes that the drop zone is completely flat. This is often not the case and can lead to larger errors if for if the drop area is slanted. It can also be a problem if the drop zone is raised since the current algorithm finds the distance to the floor directly under the package. One solution to this would be to ensure that the drop area is flat, but since this will be implemented at sea where waves can affect a boat this may not be possible. Another solution is to use the distance from the package to the height of the drop zone. However, this solution can be hard to implement, since this required sensors that can acquire this data no matter the position and rotation of the drone.
Another potential problem is that the delivery release mechanism is static. Since the algorithm will drop the package as long as it calculates that the drop will land within the acceptable error. This will make the drop algorithm susceptible to noise, leading to possible pre/post-release. The easiest solution to this problem is to use a bigger threshold for the acceptable error. But this will lead to an overall larger delivery error. Another possible solution may be to drop after some amount of calculations is within the error. But this may lead to longer delivery time and larger errors due to releasing the package too late. Another is to redesign the delivery release mechanism using a more advanced system, such as modular predictive control (MPC). But this requires much more computational power compared to the current

algorithm. Something that may or may not be possible depending on the drone's onboard computer system.
Another possible release mechanism is a manual one, where there is a receiver person in the delivery area that manually catches the package and detaches it from the drone. This solution guarantees that the delivery will be delivered accurately. But this solution also creates new problems, such as an increased possibility to hurt the receiver by hitting him/her with the package or destabilize the drone when the package is held still by the receiver.

4.2 Simulation environment

The simulation environment works overall as intended, as both the drone, rope, and wind/turbulence model work as intended and give the expected outcome. There are still improvements that can be made in all the models, but it works well enough to get a good approximation of a real environment.
Improvements that can be conducted for the drone model are quite simple yet hard to implement without a physical drone. As of now, the drone model used a mid-fidelity model. This can be upgraded to a high-fidelity model when specific parameters of a chosen drone are known. This will make the simulation more accurate to the real-life system. This improvement requires many different parameters that can only be known by testing the real fixed wing drone in wind tunnel tests or similar. So it was not possible in this thesis since the drone used was arbitrary.
The wind/turbulence model is working well as-is and does not have much to improve. One possible improvement is to make the side wind dynamic, as now the side wind is a constant disturbance which is often not the case in the real world, where the wind can change quite drastically.

4.3 Orbit follower

One part of the simulation environment that can be improved drastically is the orbit follower. When the orbit follower gets a specific radius reference is it not possible to follow it. As it either start a rotation at a larger radius or starts oscillations in its path. The oscillations happen because the look-ahead distance is too small, but increasing it makes the drone fly in a larger circle than requested. The solution used in this thesis was to keep the look-ahead distance small and set the radius reference small. This forced the drone to always try to fly at the smallest radius achievable, with the only restricting parameters being the drone's dynamics. This however made it impossible to follow a specific radius reference, making any efforts using the desired radius as a control input a fruitless endeavor.
To make the current orbit follower track a reference value it would have to be re-designed. One possible solution is creating a controller that treats the orbit follower as a system with radius as an input signal. With this can the radius reference be lowered until the drone flies at the desired radius. But this can create problems when the drone does not desire to orbit a specific point. So the better solution would probably be a complete redesign of the orbit follower.

4.4 Control algorithms

In the result of the thesis, it can be seen that the different methods of control achieve a different level of success depending on what control values are given to the simulation. An example of this that can be extrapolated from the resulting data from the nonlinear controller simulations, is that the nonlinear controller can make good use of a longer rope length, while the PID controller suffers stability losses with the increased error dynamics created from the rope length. More in-depth discussions about each of the controllers will be carried out below.

4.4.1 PID

The PID controller implemented works well in the context that for many of the simulations it successfully finds delivery vectors to deliver the package near the target. The UAV corrects its path consistently when it is subject to error dynamics such as wind and this makes it rare for the control to have a large positional error once the UAV has stabilized a trajectory. Where the PID does not make as good a job as other controllers are in the fact that it reacts slowly to changes in dynamics and it also takes time to reach a stable trajectory around the target. This shows in the data as a longer delivery time, especially with a high wind speed. Something to note regarding the high number of successful deliveries with the PID controller is that the slower control dynamics sometimes help to perturb the system to find good delivery vectors. This is because when the system counters an error dynamic slowly and henceforth veers off its circular path, when turning back it might over-correct and with this change the path of the package which has the possibility to generate a trajectory towards the target.
The control signals of the PID controller are limited but it can also be seen that it changes rapidly during runtime. It will constantly make small adjustments which can be seen as "noisy" behavior in the graphs, which can put a strain on real-world components such as servos. This behavior is not unusual for PID controllers where the integral part will become big, due to summing data over the full runtime. This is especially noticeable in the throttle.
The PID is deemed a satisfactory good solution that often finds a good delivery vector but suffers in how long it takes to find the solution. As such this is less energy efficient than some of the other controllers because of the requirement to circle the target for a longer time. Out of the three tested controllers, the PID is the easiest to understand and implement which might be beneficial to future work that wants to test this approach in a physical drone.

4.4.2 LQR

The LQR is the worst performing controller tested in this thesis. It has the largest delivery error and largest package radius. The reason this is the case is the linearization of the system. Since the system is linearized around a specific steady state point does the controller becomes the optimal controller for that specific state. Meaning that diverging from said point makes the controller perform worse and can

even make the system enter an unstable node. This can be seen in the package radius test A.2, where the velocity changes the radius much more than the PID and the nonlinear controller.
Taking a look at the control signals for the controller, it can be seen that the LQR controls the system with more consistent control signals than the PID, which can be seen especially in the throttle parameter. However the signals are still not smooth, small corrections are consistently being made so that the signal looks slightly noisy.
To improve the result of the LQR can a better steady state point be found closer to the input references used in the simulation. Another is to update the controller during the simulation, converting the LQR to an MPC. This will ensure that the linearization point is close to the current state. But will increase the computer power needed as well as slow down the system.
The result shows that linearizing around a specific state is not the best solution for a fixed wing drone. But the LQR shows an important aspect of the system. It shows that it is possible to linearize the system, which means that a more advanced controller can be applied to the system, such as the nonlinear feedback controller.

4.4.3 Nonlinear feedback controller

The nonlinear controller is the most complex controller designed for this thesis. This controller performed well and as the LQR controller did not function sufficiently well, the nonlinear controller is the only model-based controller that worked well in these scenarios. By looking at table A.4 in appendix A, it is possible to compare the performance of the PID and the nonlinear controllers at 5m/s wind turbulence. Here it can be seen that on average, the nonlinear controller achieves its deliveries quicker than the PID and the number of successful activations is higher, especially when using a longer rope. This can partially be explained by the higher robustness of the system that will be mentioned more later in this section and because the model-based controllers have the error dynamics modeled into them, the nonlinear controller will be able to handle the longer rope in a more efficient way then the PID as the rope is modeled as an error dynamic. Where we see that the PID outperforms the nonlinear controller is in the minimum radial value for the deliveries, as seen in table A.5 and similar. This shows that when flying in an advantageous configuration without too much wind, given enough time the PID tends to find good drop vectors, often better than the nonlinear controller.
The control signals for the nonlinear controller are significantly smoother compared to the other controllers. The controller does not try to make minor changes at small time intervals, but larger more significant changes slower, which usually is less damaging to servos and other mechanical components in a system. This can be seen as a general improvement in the efficiency of the nonlinear controller compared to the other two.
Where the nonlinear controller performs very well is in the perceived robustness of the output of the system. While the system is not inherently robust or stable, the feedback handles the unstable modes and because the input signal to the plant is saturated, the system remains in a robust state for the most parts. This is notable in table A.6 where it can be seen that the maximum radius error is kept to a similar

radius over the different configurations even with the higher wind speed at 10 m/s. This points towards that the nonlinear controller allows for robust control and this also points to the scalability of the UAV, as we can see that the nonlinear controller can handle both lower and higher airspeeds, while also handling many different rope lengths.

4.5 Conclusion

The results of this thesis show that with the use of different controllers, that can make use of the UAV speed limitations and the winch-rope length, an acceptable but not perfect delivery can be achieved. The result shows that the controllers are able to handle moderate wind disturbance, it does lower the accuracy of the delivery and increases the time to stabilize an orbit. Continued development of this solution is recommended before it can be applied to a real-world environment.

4.5.1 Answer to research questions

- Is it possible to drop a load close to a target from a fixed-wing UAV, by using a winch contraption combined with a circling solution?

The result shows that within a moderate wind and a low velocity and a long rope, can the expected delivery error be under 2 meters.

- How to minimize the predicted landing area of a delivered package during windy weather.

The result shows that the best controller tested is the nonlinear feedback controller. Since this has the smallest delivery error. The test also shows that a long rope and low airspeed reduce the delivery error.

- Is a mechanical winch contraption a good solution for delivering a package in a naval environment?

Yes and no. In its current state is it performing better than parachute and flyby drop. But it is still inaccurate and flings the package when released. This leads to eventual accidents if people are too close.

- What sort of control system is feasible for putting down a load while influenced by error dynamics of flying over open water?

In the different simulation tests, the nonlinear feedback controller is shown to be the best performing controller in both accuracy and disturbance handling.

- What measurement/measured data will be required to successfully make the delivery and landing during semi-harsh weather conditions?

The nonlinear feedback controller needs position and velocity data of the package in a north-east-down coordinate system or similar, airspeed, pitch angle, roll angle, yaw angle, and position of the drone. The system also uses a wind measurement when calculating the roll reference, but is not required to function.

- What model complexity is needed to design a model for delivery of a package using a winch mechanism?

The drone model needs to be at least mid-fidelity when developing the controllers. The important parts that need to be included are an approximation of the actuator mechanics and the transformation of input signals. For the controller, the result shows that a simple PID controller works to control the drone. But to make it more wind resistant can a more complex controller have a better performance.

4.6 Future Work

Due to the limited time available for the thesis was to run software in the loop simulation in a flight simulator. Doing this would give the simulation a more realistic environment to test the system in. The two simulation software analyzed for this task were Ardopilot and PX4 [16][17]. Both simulators are open source and designed to handle fixed wing-drone and quad-copters. Where Ardopilot is the easier to use of the simulators, it lacks an interface conversion between itself and Matlab/Simulink for fixed wing drones. It is also harder to adapt the model to implement the rope model in the simulation unless this were to be re-implemented in another coding language such as C++. Instead, the group recommends using PX4 as a flight simulator due to the possibility to interface directly with Simulink. This simulator requires more knowledge and time to get to work. But gives the user more freedom when it comes to adapting the drone model designed in Matlab and Simulink, especially considering the development of the "UAV Toolbox Support Package for PX4 Autopilots" which is a direct interface between Simulink and PX4. If the implementation of PX4 works the next step would be to allow the PX4 controller to be implemented on hardware for some real-world testing. The PX4 support toolbox has functionalities to allow for direct hardware in the loop (HIL) testing which is a big step toward a working prototype. The "UAV support package for PX4" is not that old and does not necessarily have all the functionalities to perfectly translate all the functions and modules written in this thesis with the required sample rates so further research into the limits and possibilities of this needs to be conducted.
Another future work is to update the drone model to a high-fidelity model. This can be implemented once the drone used in real world testing is defined, making the drone model even more accurate to the real world, removing some of the approximations used in this thesis model. One thing to note is that the controllers might need to be updated after this change has been applied since this can change how and what input signals are used.
Another future research potential after this thesis would be to implement and test different controllers for the UAV model. Some of the possibilities for controllers are to implement model predictive control to recalculate new controller parameters so that an LQR approach has a larger chance of succeeding due to the moving of linearization. This however would be dependent on the computational limitations on board the physical UAV, which is why this was not further researched in this thesis. Another possible control method would be to train a deep neural network to set the control parameters dynamically to try to achieve good control, this however

can be considered a more niche research subject. An alternate thesis is to modify the existing controllers so they can be implemented to minimize the landing area of the fixed wing drone in a landing algorithm. This would be done by letting the drone lower its velocity during its orbit and this makes it possible to approximate an optimal landing path. As we in this thesis have seen that it is possible to find stable circling trajectories, from where the system can make good approximations on the UAV: s position, it can be possible to design an algorithm, possible controlled with an MPC or similar, that efficiently makes a controlled landing towards a target. But more research is needed for the viability of this solution.

Bibliography

[1] N. Rees, J. Howitt, N. Breyley, P. Geoghegan, and C. Powel, "A simulation study of drone delivery of Automated External Defibrillator (AED) in Out of Hospital Cardiac Arrest (OHCA) in the UK," *PLOS ONE*, vol. 16, no. 11, pp. 1–9, Nov 2021.

[2] R. Pavithran, V. Lalith, C. Naveen, S. P. Sabari, M. A. Kumar, and V. Hariprasad, "A prototype of fixed wing UAV for delivery of medical supplies," *IOP Conference Series: Materials Science and Engineering*, vol. 995, no. 1, p. 012015, nov 2020.

[3] Zipline, "Flyzipline," 2022, [Online]. Available: https://flyzipline.com/ (Accessed on: 2022-04-20).

[4] Everdrone, "Everdrone," 2022, [Online]. Available: https://everdrone.com/ (Accessed on: 2022-04-20).

[5] Amazon, "Amazon Prime Air," 2016, [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011 (Accessed on: 2022-04-20).

[6] H. Nguyen, T. Quyen, V.-C. Nguyen, A. Le, H. Tran, and M. Nguyen, "Control Algorithms for UAVs: A Comprehensive Survey," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 7, p. 164586, may 2020.

[7] J. C. Simons and B. C. Dixon, "Long-line loiter: Improvement of some free-fall and circling-line techniques," United States Aeronatical Systems Division, Wright-Patterson, Ohio, USA, ASD-TR-69-95, Volume 1, 1969,.

[8] C. Michael, "Walmart to begin drone delivery service to 4 million households," 2022, [Online]. Available: https://www.theguardian.com/cities/2022/may/24/walmart-drone-delivery-4-million-us-households (Accessed on: 2022-06-08).

[9] Mathworks, "Transition from Low to High-Fidelity UAV Models in Three Stages," 2022, [Online]. Available: https://se.mathworks.com/help/uav/ug/transition-from-low-to-high-fidelity-uav-models.html (Accessed on: 2022-04-20).

[10] Y. Yan, J. Yang, C. Liu, M. Coombes, S. Li, and W.-H. Chen, "On the actuator dynamics of dynamic control allocation for a small fixed-wing uav with direct lift control," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 984–991, 2020.

[11] Mathworks, "Von karman wind turbulence model (continuous)," 2022, [Online]. Available: https://se.mathworks.com/help/aeroblks/vonkarmanwindturbulencemodelcontinuous.html (Accessed on: 2022-04-20).

[12] ——, "Horizontal Wind Model: Transform horizontal wind into body-axes coordinates," 2022, [Online]. Available: https://se.mathworks.com/help/aeroblks/horizontalwindmodel.html (Accessed on: 2022-04-28).
[13] ——, "Orbit follower: Orbit location of interest using UAV," 2022, [Online]. Available: https://se.mathworks.com/help/uav/ref/orbitfollower.html?s_tid=doc_ta (Accessed on: 2022-04-20).
[14] ——, "icare: Implicit solver for continuous-time algebraic Riccati equations," 2022, [Online]. Available: https://se.mathworks.com/help/control/ref/icare.html (Accessed on: 2022-04-20).
[15] ——, "trim: Find trim point of dynamic system," 2022, [Online]. Available: https://se.mathworks.com/help/simulink/slref/trim.html?s_tid=doc_ta (Accessed on: 2022-04-20).
[16] Ardupilot, "Ardupilot - Versatile, Trusted, Open," 2022, [Online]. Available: https://ardupilot.org/ (Accessed on: 2022-05-09).
[17] Px4 Autopilot, "Open Source Autopilot for Drone Developers," 2022, [Online]. Available: https://px4.io/ (Accessed on: 2022-05-09).

A

Appendix: Test Data

Table A.1: Data from radius test for PID-controller with 0 m/s wind

package radius (m)		velocity (m/s)						
		12	16	20	24	28	32	36
rope Length (m)	15	17.5511	24.7502	31.8702	38.8962	45.9223	53.0233	60.1010
	18	15.9317	23.2719	30.6946	37.9796	45.1664	52.2514	59.4609
	21	14.1658	21.4560	29.3167	36.8053	44.1145	51.4223	58.6029
	24	12.6296	19.2179	27.4501	35.3703	42.8471	50.2863	57.6621
	27	11.4037	16.8012	25.1170	33.5294	41.3887	48.9808	56.4900
	30	10.4477	14.7544	22.3631	31.2399	39.6136	47.5271	55.1938
	33	9.6910	13.2300	19.4902	28.7430	37.4871	45.7369	53.7069
	36	9.0779	12.0398	16.9447	25.5536	35.1448	43.7115	51.9402
	39	8.5698	11.1123	15.0118	22.2851	32.2128	41.3516	50.0005
	42	8.1407	10.4208	13.5728	19.2699	28.8604	38.7611	47.7117
	45	7.7726	9.8229	12.5101	16.9132	25.2496	35.7188	45.1578
	48	7.4522	9.3251	11.6777	15.1851	21.8942	32.1803	42.2749
	51	7.1704	8.9138	10.9965	13.9094	19.0990	28.4508	39.1311
	54	6.9200	8.5277	10.4395	12.9309	17.0023	24.8159	35.6531
	57	6.6954	8.2317	9.9597	12.1503	15.4641	21.6164	31.7797
	60	6.4924	7.9426	9.5569	11.5068	14.3028	19.0967	27.9917
	63	6.3077	7.6655	9.1706	10.9623	13.3903	17.2238	24.4691
	66	6.1388	7.4314	8.8552	10.4920	12.6475	15.8214	21.5105
	69	5.9832	7.2035	8.5488	10.0787	12.0248	14.7351	19.2261
	72	5.8389	7.0059	8.2797	9.7107	11.4906	13.8627	17.5053

Table A.2: Data from radius test for LQR-controller with 0 m/s wind

package radius(m)		velocity (m/s)						
		12	16	20	24	28	32	36
rope Length (m)	15	28.7715	41.1659	54.8259	69.8091	86.2170	104.0526	123.3541
	18	27.6540	40.2931	54.1232	69.2374	85.7938	103.7050	123.0642
	21	26.3081	39.2242	53.2951	68.5589	85.1898	103.2180	122.6644
	24	24.6606	37.8866	52.2830	67.8100	84.5705	102.6805	122.1309
	27	22.9908	36.3031	51.1152	66.9008	83.8383	102.0345	121.6328
	30	21.3616	34.4787	49.7392	65.8276	82.9423	101.3477	121.0605
	33	19.8893	32.3456	48.1059	64.6129	81.9657	100.4865	120.3473
	36	18.6105	30.0747	46.2690	63.1776	80.8418	99.5812	119.5966
	39	17.5178	27.8089	44.2608	61.6696	79.6159	98.6347	118.6746
	42	16.5846	25.6868	41.8870	59.7277	78.2022	97.5279	117.7653
	45	15.7818	23.8528	39.1685	58.1751	76.7631	96.2176	116.7704
	48	15.0846	22.2933	36.4430	55.8863	75.1411	94.9917	115.6228
	51	14.4730	20.9791	33.7667	53.1256	73.1065	93.3590	114.3833
	54	13.9314	19.8662	31.1230	50.6050	71.2031	91.7954	113.1270
	57	13.4477	18.9133	28.8762	47.6651	69.1027	90.0495	111.7348
	60	13.0121	18.0871	26.9476	44.9258	66.6559	88.1526	110.2599
	63	12.6172	17.3621	25.3179	41.8243	63.9399	86.3744	108.7180
	66	12.2568	16.7188	23.9402	38.4981	61.1302	84.4091	107.1880
	69	11.9261	16.1425	22.7652	35.7596	58.5051	82.3455	105.5364
	72	11.6211	15.6217	21.7508	33.3216	55.0822	79.2059	103.4659

Table A.3: Data from radius test for Nonlinear-controller with 0 m/s wind

package radius (m)		velocity (m/s)						
		12	16	20	24	28	32	36
rope Length (m)	15	17.0583	24.0362	30.8641	37.5651	44.2148	50.7342	57.2477
	18	15.3856	22.5455	29.6561	36.5598	43.3472	50.0368	56.5731
	21	13.6522	20.5978	28.1496	35.2867	42.3176	49.1053	55.7765
	24	12.1512	18.3091	26.2573	33.8313	40.9969	47.8878	54.6764
	27	10.9684	15.9626	23.8495	31.8665	39.3744	46.6249	53.4820
	30	10.0505	14.0204	21.0368	29.6276	37.6199	44.9427	52.2297
	33	9.3250	12.5547	18.1784	26.8174	35.2941	43.2419	50.4979
	36	8.7371	11.4592	15.8196	23.6329	32.6956	40.8850	48.5955
	39	8.2500	10.6152	14.0634	20.3774	29.6374	38.5576	46.5242
	42	7.8384	9.9418	12.7808	17.6315	26.1634	35.5774	44.1201
	45	7.4852	9.3885	11.8120	15.5690	22.6092	32.3709	41.2971
	48	7.1779	8.9228	11.0496	14.0785	19.5188	28.6032	38.3067
	51	6.9074	8.5231	10.4282	12.9692	17.1336	24.8842	34.8573
	54	6.6671	8.1747	9.9074	12.1068	15.4036	21.5125	31.0470
	57	6.4515	7.8669	9.4612	11.4098	14.1278	18.8052	27.1889
	60	6.2568	7.5918	9.0721	10.8288	13.1472	16.8041	23.6087
	63	6.0796	7.3436	8.7279	10.3325	12.3622	15.3302	20.6048
	66	5.9172	7.1178	8.4197	9.9004	11.7127	14.2074	18.3113
	69	5.7677	6.9107	8.1408	9.5182	11.1608	13.3177	16.6066
	72	5.6292	6.7197	7.8862	9.1758	10.6821	12.5880	15.3140

Table A.4: Data from delivery-test with wind speed at 5m/s

Model	Speed reference (m/s) and rope length (m)	Mean delivery time (sec)	Mean delivery error (m)	Successful activation during test
Straight delivery	v_{ref} = 14, L = 30	20.6300	8.9129	56%
	v_{ref} = 14, L = 45	18.6600	7.9809	67%
	v_{ref} = 14, L = 60	21.8000	6.7427	57%
	v_{ref} = 20, L = 30	15.4800	16.5374	68%
	v_{ref} = 20, L = 45	15.7600	15.7066	69%
	v_{ref} = 20, L = 60	19.1300	14.7353	59%
	v_{ref} = 26, L = 30	12.2000	23.2874	76%
	v_{ref} = 26, L = 45	14.5500	21.6749	69%
	v_{ref} = 26, L = 60	15.9000	21.6654	66%
PID	v_{ref} = 14, L = 30	84.6392	2.7780	97%
	v_{ref} = 14, L = 45	69.6500	1.5634	100%
	v_{ref} = 14, L = 60	82.9457	1.4123	92%
	v_{ref} = 20, L = 30	163.0000	6.3122	2%
	v_{ref} = 20, L = 45	100.3298	3.9953	94%
	v_{ref} = 20, L = 60	70.8265	2.3876	28%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	168.2857	9.0199	7%
	v_{ref} = 26, L = 60	99.3721	5.7401	86%
LQR	v_{ref} = 14, L = 30	35.4583	4.7739	24%
	v_{ref} = 14, L = 45	56.3288	4.5818	73%
	v_{ref} = 14, L = 60	93.5930	1.6051	86%
	v_{ref} = 20, L = 30	31.0000	10.8761	3%
	v_{ref} = 20, L = 45	31.0000	12.2964	7%
	v_{ref} = 20, L = 60	31.0286	13.9243	35%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 60	Unsuccessful	Unsuccessful	0%
Nonlinear	v_{ref} = 14, L = 30	69.5556	2.7417	99%
	v_{ref} = 14, L = 45	58.9300	1.6316	100%
	v_{ref} = 14, L = 60	63.5269	1.4343	93%
	v_{ref} = 20, L = 30	161.6000	7.3453	5%
	v_{ref} = 20, L = 45	79.1818	3.3324	99%
	v_{ref} = 20, L = 60	62.5833	2.2411	96%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	178.2000	8.7469	5%
	v_{ref} = 26, L = 60	96.3646	4.4416	96%

Table A.5: Minimum/maximum test with wind speed at 5 m/s

Controller	Speed reference (m/s) and rope length (m)	minimum value (m)	avrage minimum (m)	maximum value (m)	average maximum (m)
PID	v_{ref} = 14, L = 30	0.3015	1.0683	40.4319	31.4054
	v_{ref} = 14, L = 45	0.1768	0.9358	38.9159	30.0665
	v_{ref} = 14, L = 60	0.3960	1.2339	39.7554	32.6622
	v_{ref} = 20, L = 30	1.7309	6.6647	47.0005	40.9341
	v_{ref} = 20, L = 45	0.4472	1.0596	40.5416	32.6137
	v_{ref} = 20, L = 60	0.1239	0.8446	40.8547	31.8554
	v_{ref} = 26, L = 30	12.8678	18.9454	65.1122	53.3984
	v_{ref} = 26, L = 45	0.4054	4.7206	48.3224	41.1457
	v_{ref} = 26, L = 60	0.3023	1.0421	43.5680	34.6633
LQR	v_{ref} = 14, L = 30	4.3843	14.1428	66.5573	54.6037
	v_{ref} = 14, L = 45	0.7199	4.7320	60.4681	42.4353
	v_{ref} = 14, L = 60	0.3155	1.8082	52.3360	37.5211
	v_{ref} = 20, L = 30	30.8411	36.5215	78.7652	71.0591
	v_{ref} = 20, L = 45	16.4323	24.8177	75.7331	63.0174
	v_{ref} = 20, L = 60	2.8986	10.7321	63.6477	53.4185
	v_{ref} = 26, L = 30	43.5654	52.4678	99.8860	92.9505
	v_{ref} = 26, L = 45	39.2632	45.3660	97.7236	89.4941
	v_{ref} = 26, L = 60	25.5021	34.3825	92.8516	81.6690
Nonlinear	v_{ref} = 14, L = 30	0.3260	0.9490	36.0572	30.3786
	v_{ref} = 14, L = 45	0.2948	0.7788	35.2293	28.8783
	v_{ref} = 14, L = 60	0.2325	0.8834	35.8792	27.4071
	v_{ref} = 20, L = 30	4.8066	8.0209	45.5045	36.9691
	v_{ref} = 20, L = 45	0.3641	1.0418	36.2133	30.6153
	v_{ref} = 20, L = 60	0.2330	0.8268	39.5822	29.1771
	v_{ref} = 26, L = 30	15.1092	19.9359	54.5237	48.4217
	v_{ref} = 26, L = 45	1.2297	3.9822	40.2178	35.2435
	v_{ref} = 26, L = 60	0.2798	0.7298	41.8335	31.3187

Table A.6: Minimum/maximum test with wind speed at 10 m/s

Controller	Speed reference (m/s) and rope length (m)	minimum value (m)	avrage minimum (m)	maximum value (m)	average maximum (m)
PID	v_{ref} = 14, L = 30	0.3646	1.4569	140.4863	36.0572
	v_{ref} = 14, L = 45	0.4426	1.8279	102.1821	35.2293
	v_{ref} = 14, L = 60	0.6230	2.4573	105.6795	35.8792
	v_{ref} = 20, L = 30	0.4044	1.8042	166.7565	45.5045
	v_{ref} = 20, L = 45	0.3010	1.2242	149.2995	36.2133
	v_{ref} = 20, L = 60	0.1686	1.9331	153.5956	39.5822
	v_{ref} = 26, L = 30	3.4507	7.8359	87.7563	54.5237
	v_{ref} = 26, L = 45	0.6278	1.5865	79.5291	40.2178
	v_{ref} = 26, L = 60	0.1147	1.3739	177.4942	41.8335
LQR	v_{ref} = 14, L = 30	0.3838	6.2389	95.2169	73.3207
	v_{ref} = 14, L = 45	0.6102	1.7469	83.3574	64.8550
	v_{ref} = 14, L = 60	0.0374	1.1552	87.8841	59.0842
	v_{ref} = 20, L = 30	21.7544	29.5396	101.6117	91.5064
	v_{ref} = 20, L = 45	3.7772	14.3801	109.0390	88.9100
	v_{ref} = 20, L = 60	1.2845	5.5279	74.5826	64.5353
	v_{ref} = 26, L = 30	40.8130	51.2226	129.3572	108.5553
	v_{ref} = 26, L = 45	34.2352	44.7609	116.4286	104.0461
	v_{ref} = 26, L = 60	25.9599	32.8014	122.1133	101.4072
Nonlinear	v_{ref} = 14, L = 30	0.2409	1.0993	54.6921	46.9703
	v_{ref} = 14, L = 45	0.3737	1.8797	63.6447	50.0791
	v_{ref} = 14, L = 60	0.6695	2.1630	65.7749	51.6729
	v_{ref} = 20, L = 30	0.5618	1.8729	82.6323	59.4214
	v_{ref} = 20, L = 45	0.1372	1.2693	73.0962	51.7513
	v_{ref} = 20, L = 60	0.0748	1.3661	54.4580	46.9399
	v_{ref} = 26, L = 30	5.8195	9.8294	81.0316	66.7768
	v_{ref} = 26, L = 45	0.6668	1.5727	66.3927	51.0511
	v_{ref} = 26, L = 60	0.1602	1.3629	62.7818	47.9812

Table A.7: Data from side wind-test with wind speed at 5m/s

Model	Speed reference (m/s) and rope length (m)	Mean delivery time (sec)	Mean delivery error (m)	Successful activation during test
PID	v_{ref} = 14, L = 30	104.2500	7.7155	13.3333%
	v_{ref} = 14, L = 45	81.0000	5.2509	16.6667%
	v_{ref} = 14, L = 60	67.6296	4.6874	90%
	v_{ref} = 20, L = 30	90.0000	7.4644	56.6667%
	v_{ref} = 20, L = 45	97.4500	9.5059	66.6667%
	v_{ref} = 20, L = 60	132.8750	9.4007	26.6667%
	v_{ref} = 26, L = 30	84.5385	12.3719	86.6667%
	v_{ref} = 26, L = 45	80.0435	14.1759	76.6667%
	v_{ref} = 26, L = 60	71.9286	15.3931	93.3333%
LQR	v_{ref} = 14, L = 30	37.2727	4.5258	36.6667%
	v_{ref} = 14, L = 45	50.0000	5.3434	70%
	v_{ref} = 14, L = 60	76.4444	6.4938	60%
	v_{ref} = 20, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 20, L = 45	Unsuccessful	Unsuccessful	0%
	v_{ref} = 20, L = 60	31.8000	6.7630	16.6667%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 60	Unsuccessful	Unsuccessful	0%
Nonlinear	v_{ref} = 14, L = 30	40.8333	3.7794	100%
	v_{ref} = 14, L = 45	46.2667	3.8819	100%
	v_{ref} = 14, L = 60	88.4231	3.1375	86.6667%
	v_{ref} = 20, L = 30	81.8333	9.4196	40%
	v_{ref} = 20, L = 45	68.2400	5.4229	83.3333%
	v_{ref} = 20, L = 60	50.4667	3.5703	100%
	v_{ref} = 26, L = 30	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 45	Unsuccessful	Unsuccessful	0%
	v_{ref} = 26, L = 60	84.8571	6.5154	70%

B

Appendix: Code

B.1 General

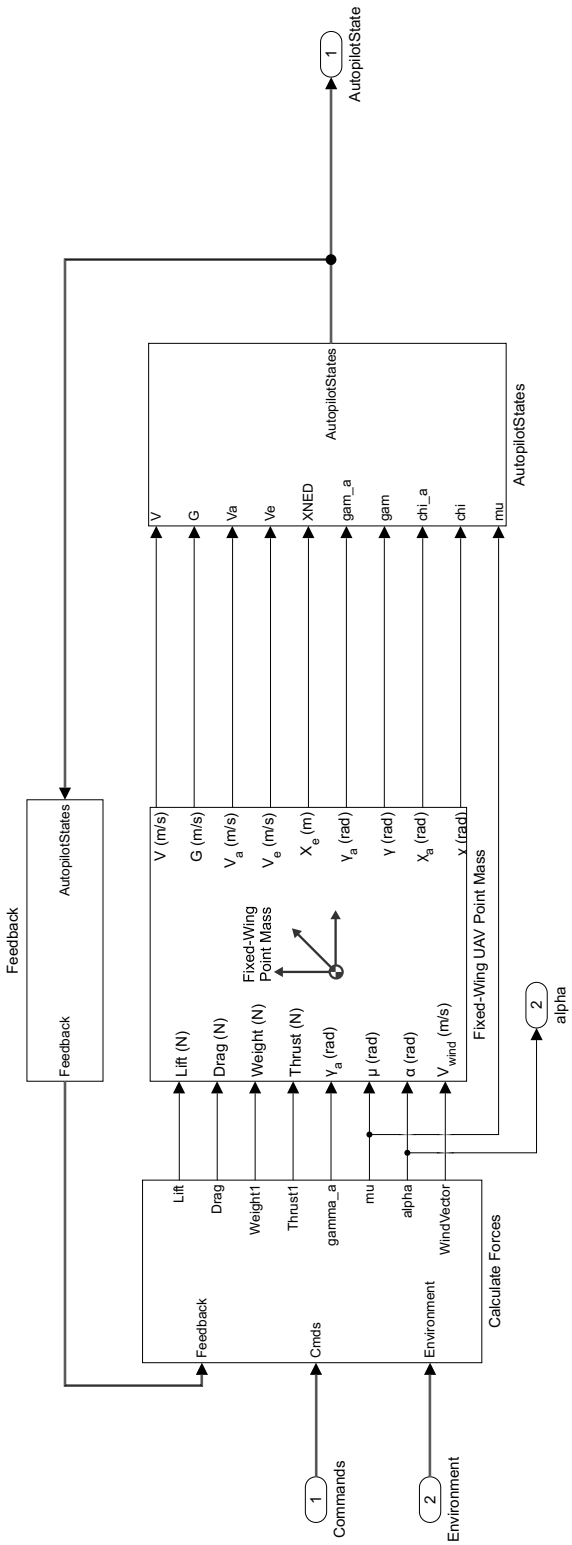

Figure B.1: Plant model of Fixed wing drone

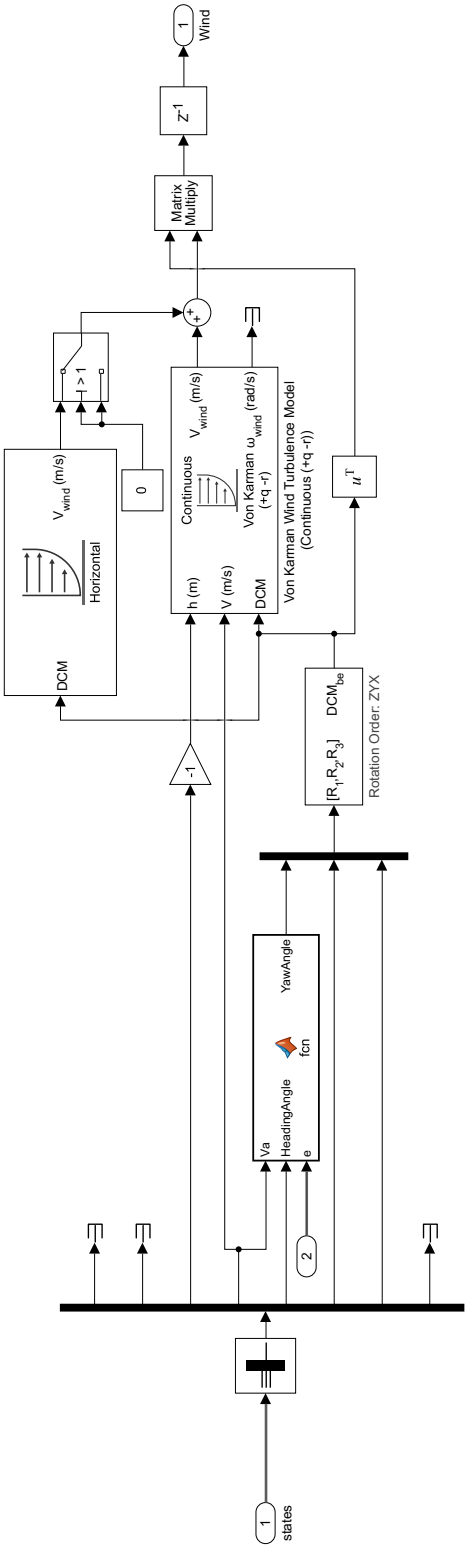

Figure B.2: Wind/turbulence model

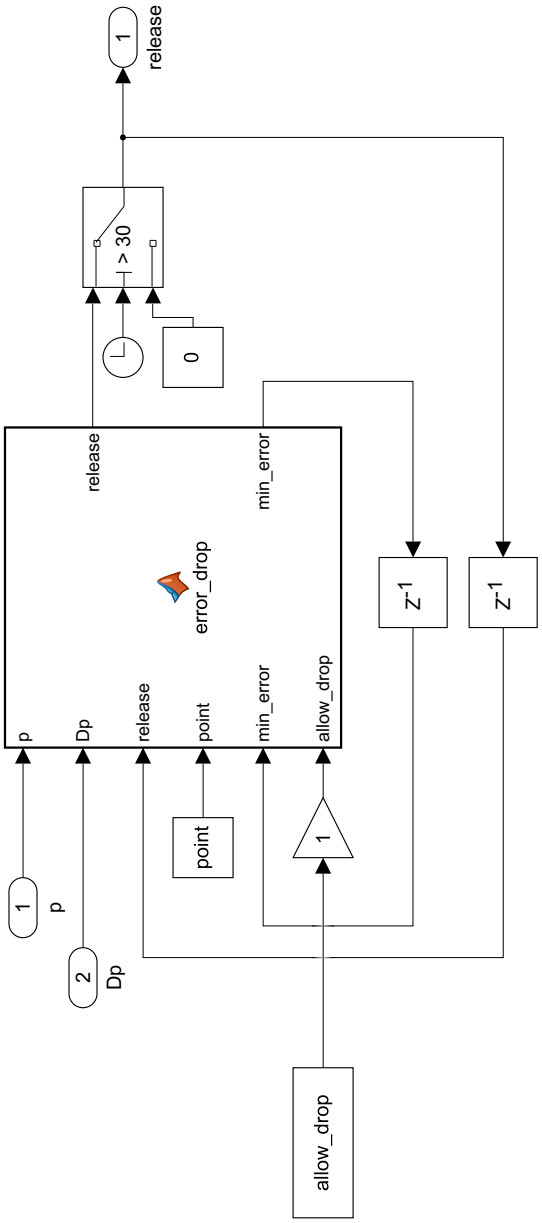

Figure B.3: Drop activation

```
function [release,min_error] = error_drop(p,Dp,release,point,min_error,allow_drop)
if allow_drop
    if p(3,end) < 0
        t = -Dp(3,end)/(2*9.82) + sqrt((-Dp(3,end)/(2*9.82))^2 - p(3,end)/9.82);
        if t > 0
            dist_release = sqrt(((Dp(1,end))^2+(Dp(2,end))^2))*t;
            dist_to_point = sqrt((p(1,end)-point(1))^2+(p(2,end)-point(2))^2);
            error_angle = atan2(-Dp(2,end),Dp(1,end)) - atan2(p(2,end)-point(2),-(p(1,end)-point(1)));
            error = sqrt(dist_to_point^2 + dist_release^2 - 2*dist_to_point*dist_release*cos(error_angle));

            if error < 2
                release = 1;
            end
            if  error < min_error
                min_error = error;
            end
        end
    end
end
```

Figure B.4: Drop error calculation

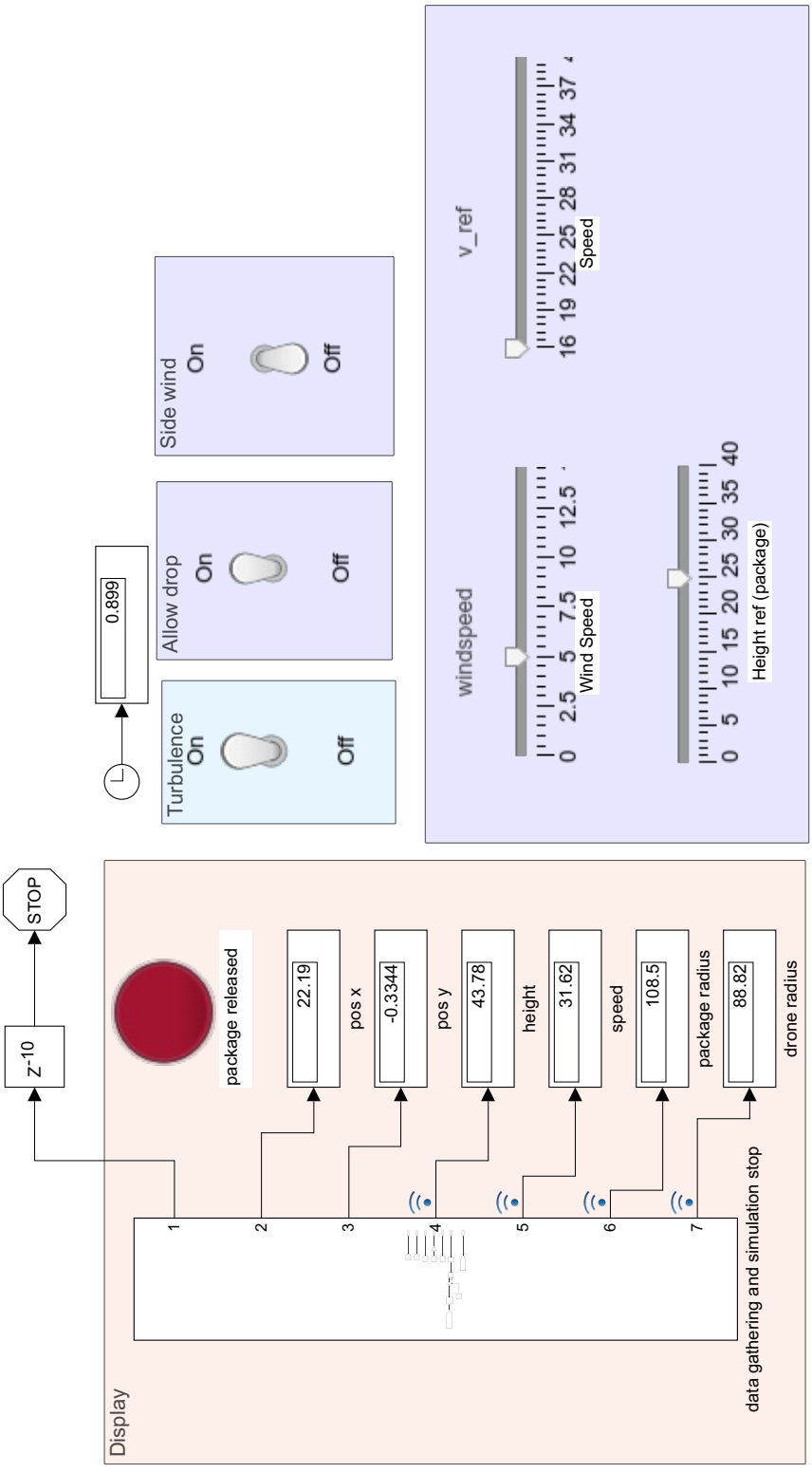

Figure B.5: Control room of simulation

B.2 PID

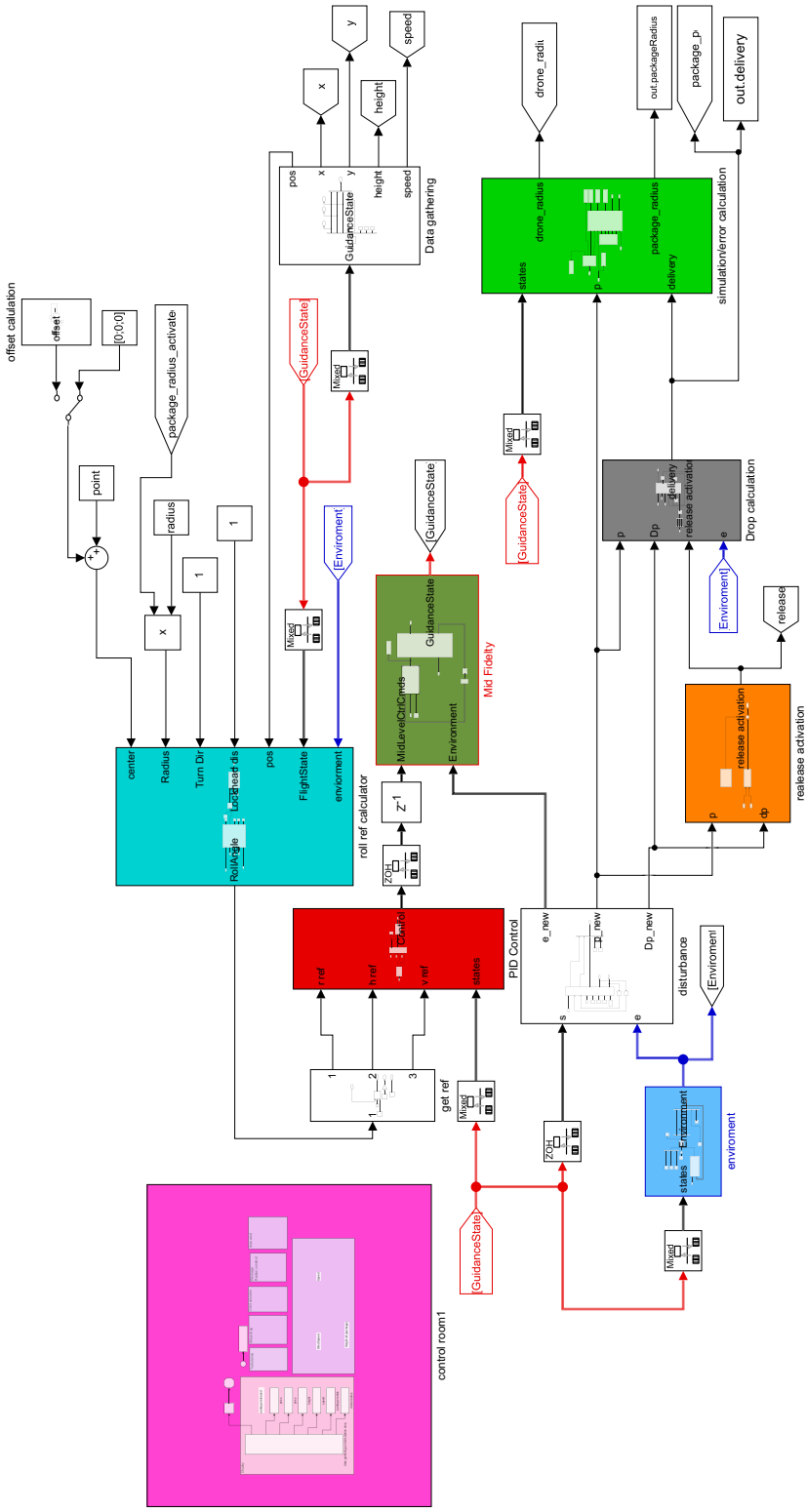

Figure B.6: Main for PID controller simulation

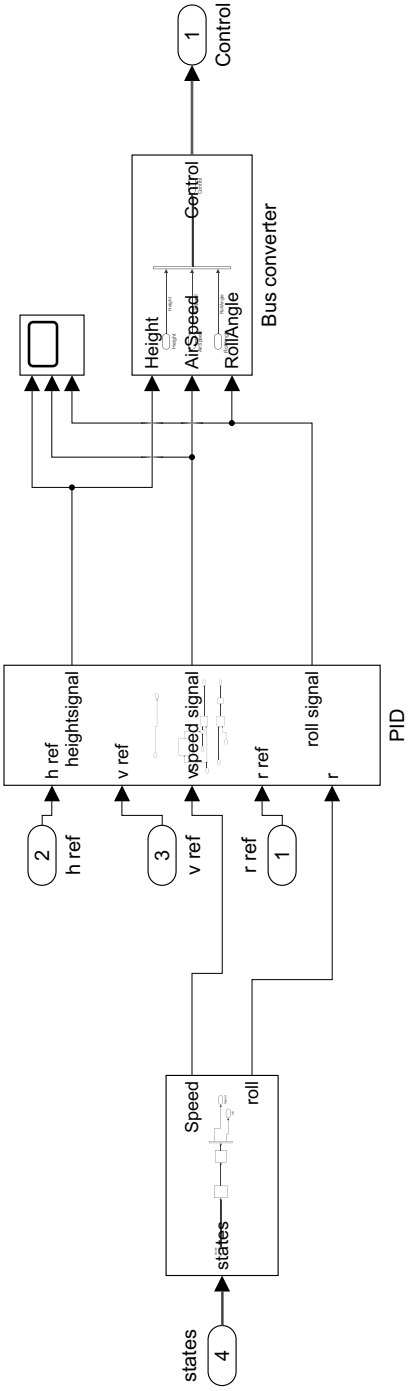

Figure B.7: PID controller

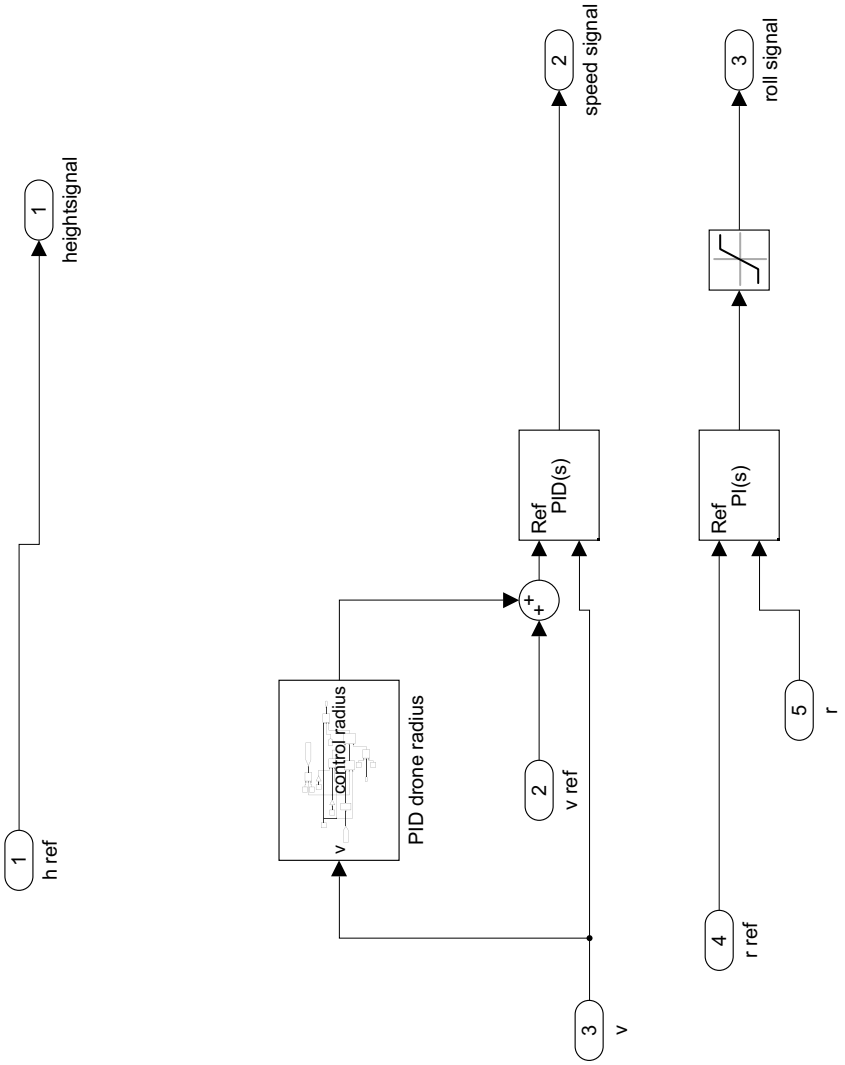

Figure B.8: PID controller subsystem

B.3 LQR

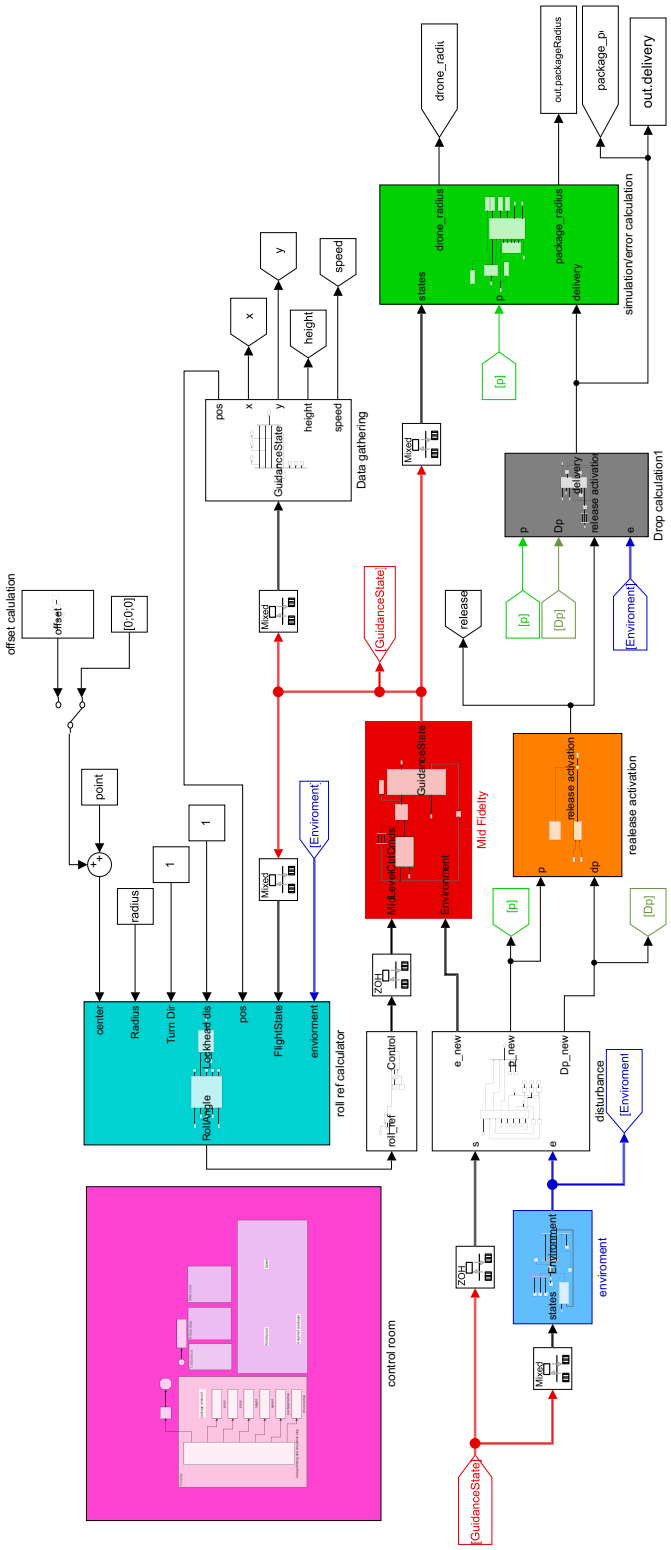

Figure B.9: Main for LQR controller simulation

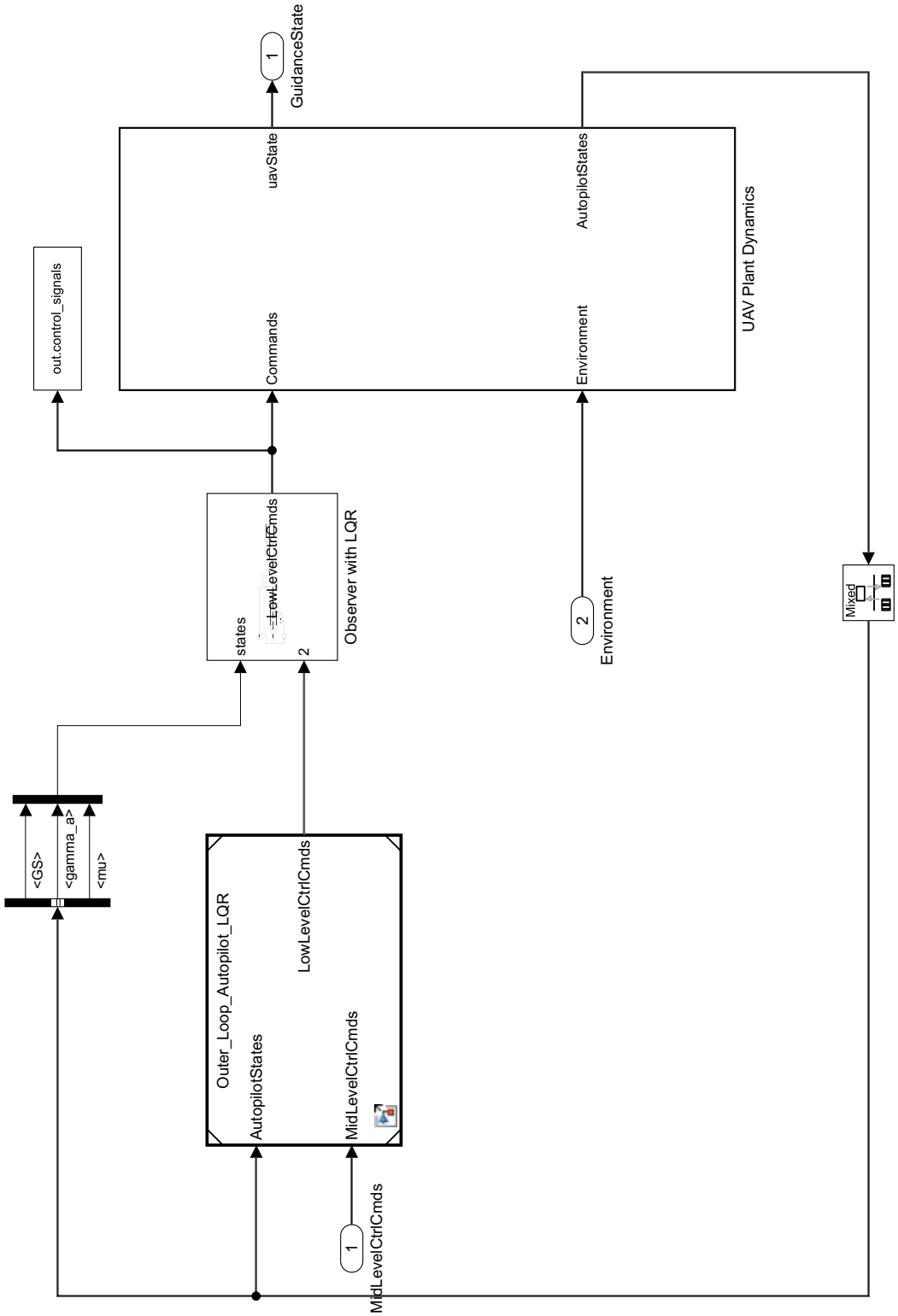

Figure B.10: Plant-controller loop for LQR controller

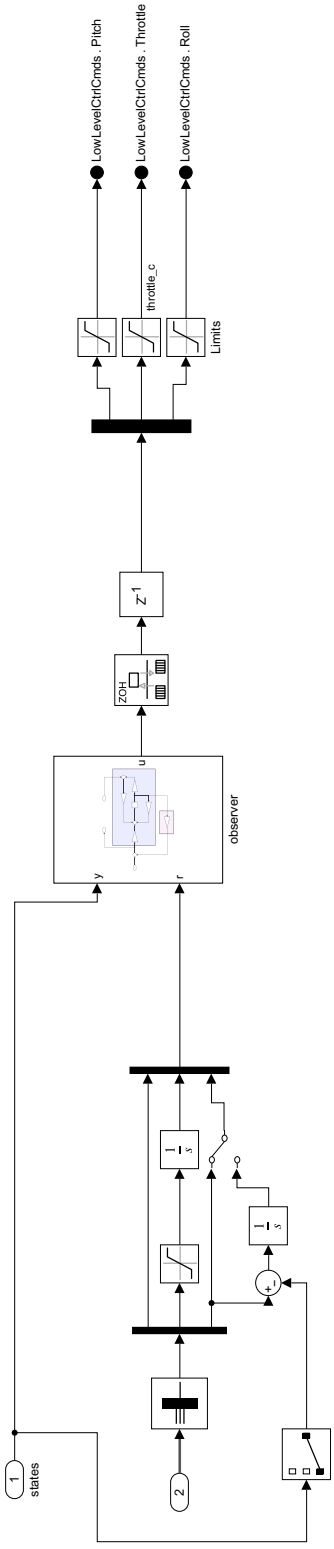

Figure B.11: LQR controller

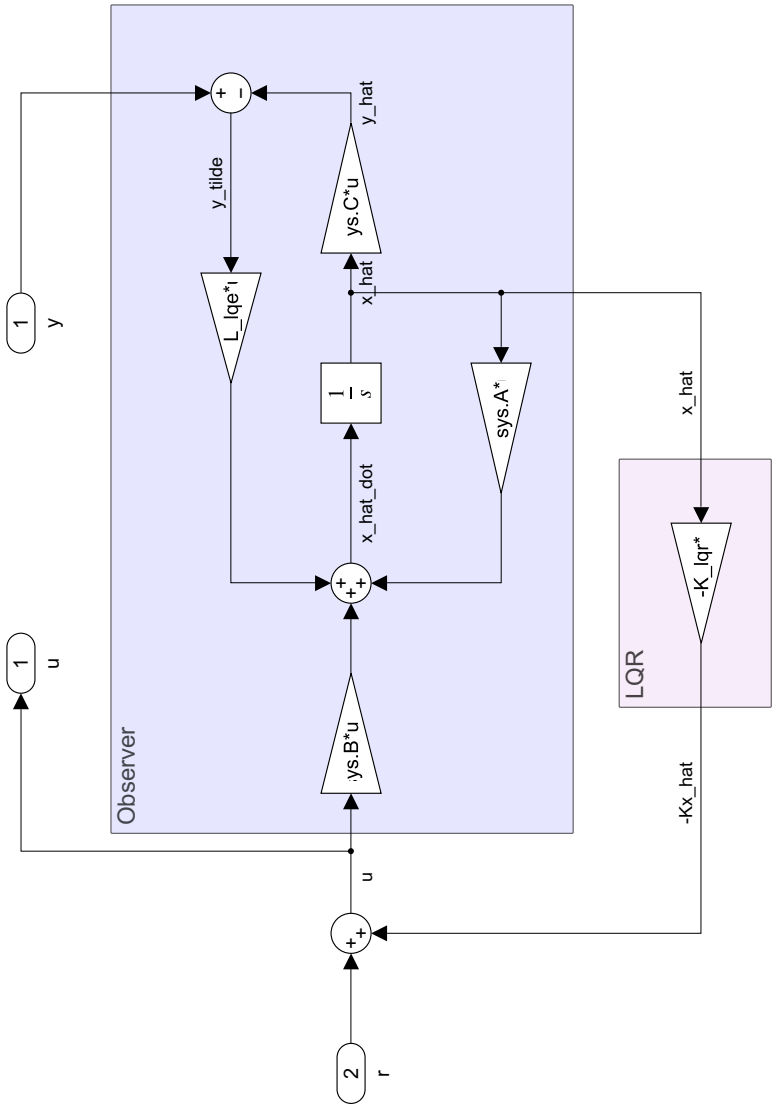

Figure B.12: Observer

B.4 Nonlinear Feedback controller

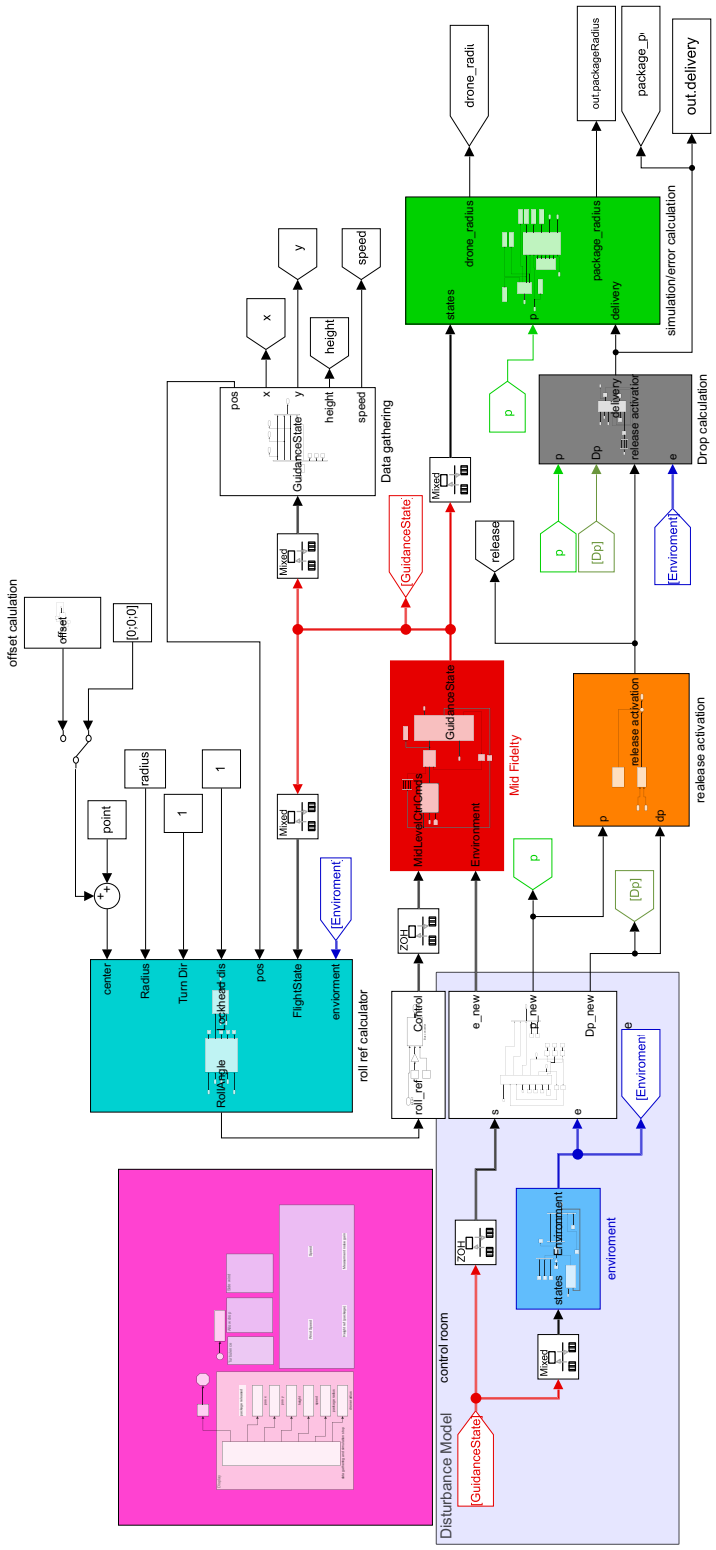

Figure B.13: Main for Nonlinear controller simulation

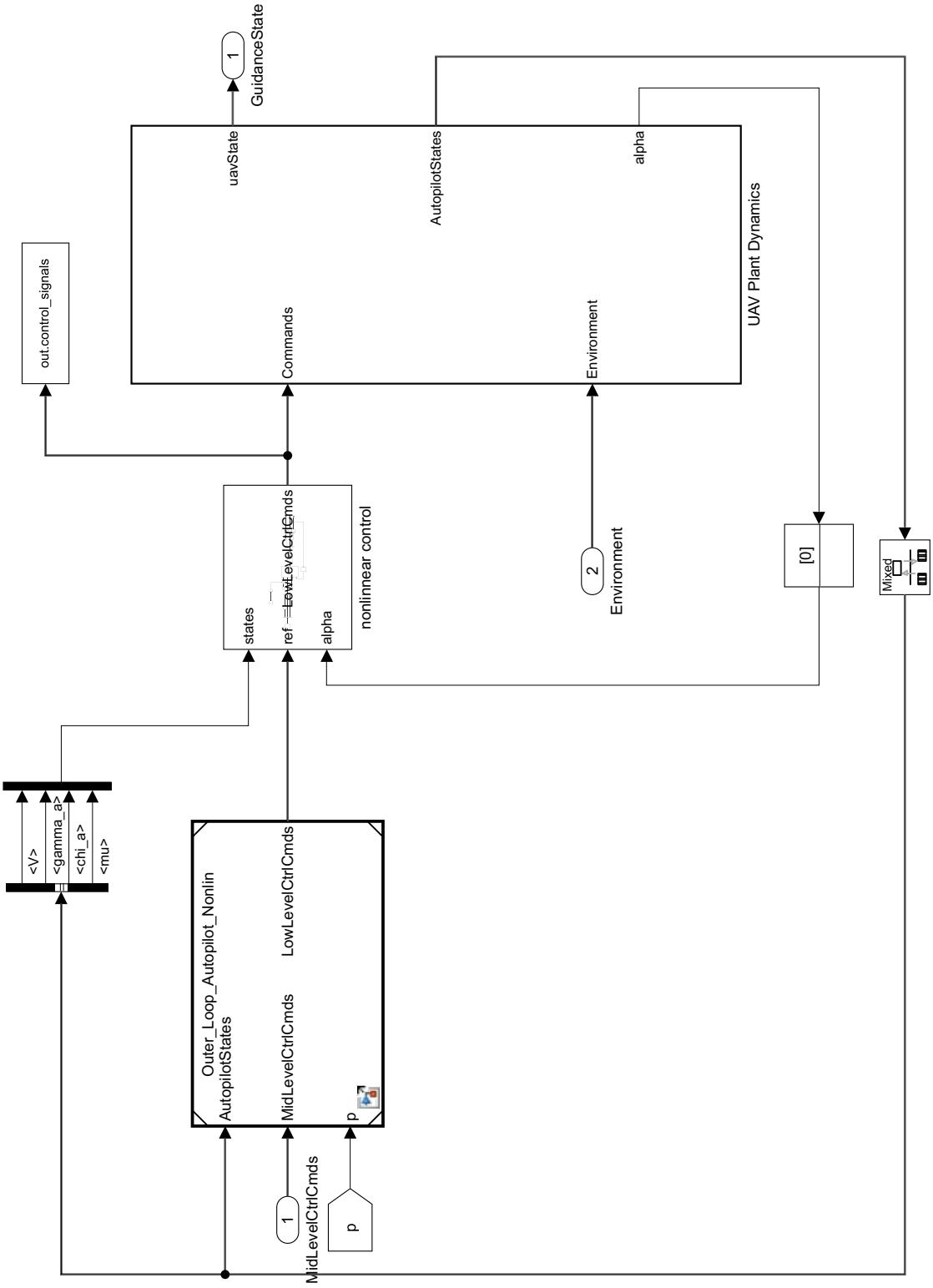

Figure B.14: Plant-controller loop for Nonlinear controller

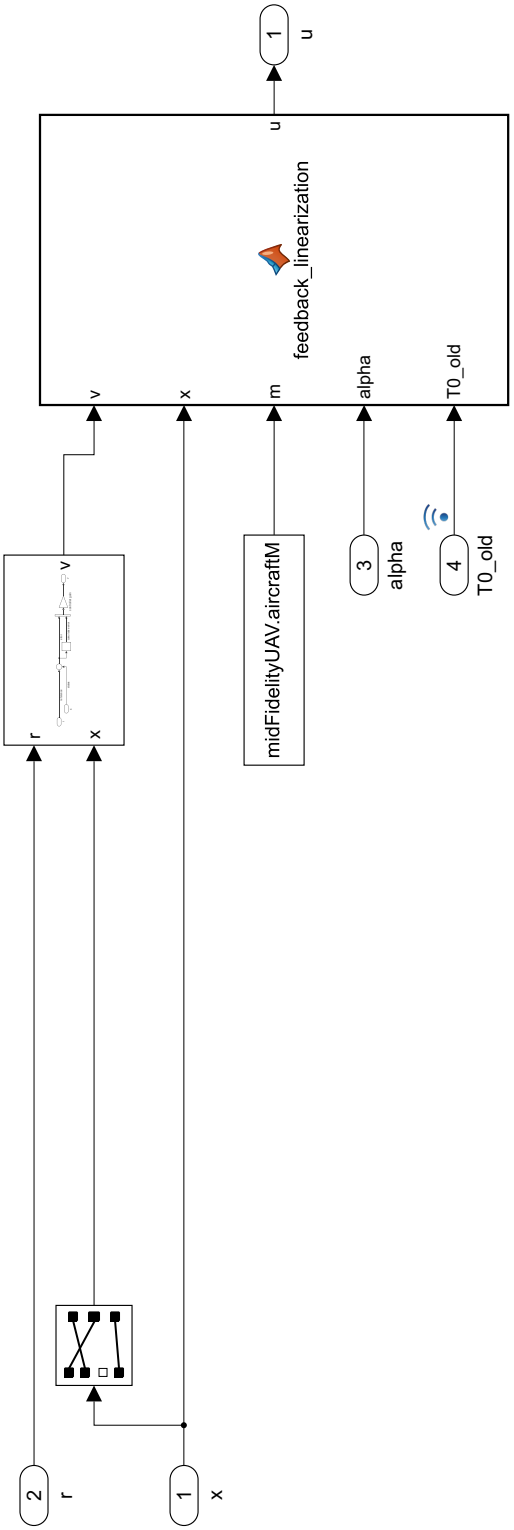

Figure B.15: Nonlinear controller

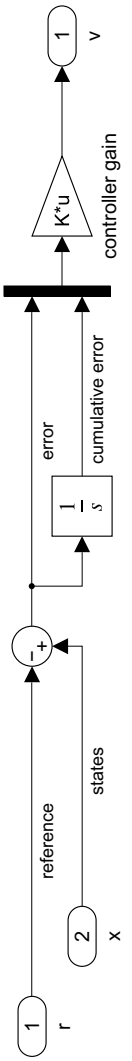

Figure B.16: Nonlinear controller subsystem

```
function u = feedback_linearization(v,x,m,alpha,T0_old)
u = v;
% r(1) = r(1) - x(2);

W = m*9.82;
L = 0.5*1.2250*x(1)^2*0.9820*(alpha*18.5000 + 0.3800);
D = 0.5*1.2250*x(1)^2*0.9820*(0.0220 +(alpha*18.5000 + 0.3800)^2/(6.9000*pi));
T0 = T0_old;
T = 9.8065*0.4536*(T0-(0.047*sqrt(T0)*x(1)));

a_gamma = ((0.5*1.2250*x(1)^2*0.9820*(-x(2)*18.5000 + 0.3800))*cos(x(4))-W*cos(x(2))-T*cos(x(4))*x(2))/(m*x(1));
b_gamma = ((0.5*1.2250*x(1)^2*0.9820*18.5000)*cos(x(4))+T*cos(x(4)))/(m*x(1));
u(1) =  (v(1) - a_gamma)/b_gamma;
%u(1) = v(1);


a_V = (- D - W*x(2))/m;
b_V =  9.8065*0.4536*(cos(alpha))/m;
u(2) =(v(2) - a_V)/b_V;

a_chi = 0;
b_chi = (L+T*sin(alpha))/(m*x(1)*cos(x(2)));
u(3) = ((v(3) - a_chi)/b_chi);
end
```

Figure B.17: Nonlinear Feedback linearization

C

Appendix: Figures

C.1 Plots from Package delivery test in turbulence

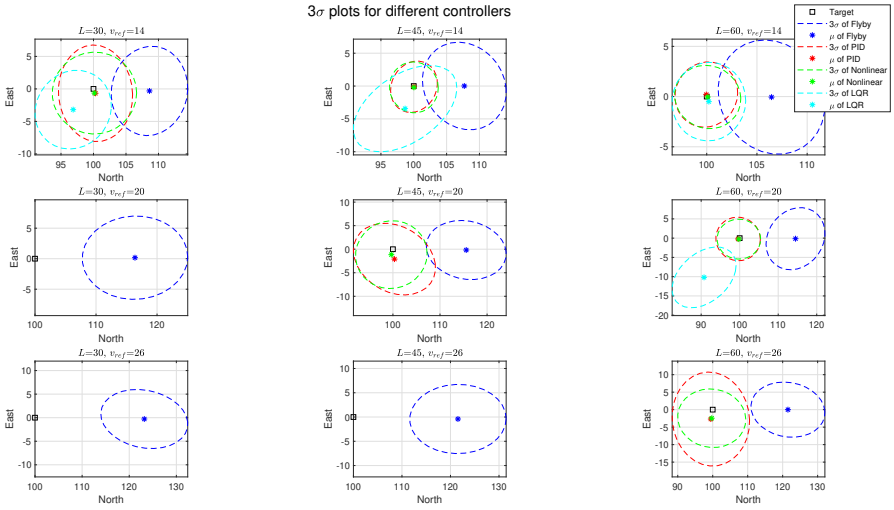

Figure C.1: 3σ plots for Package delivery test with 5m/s turbulence

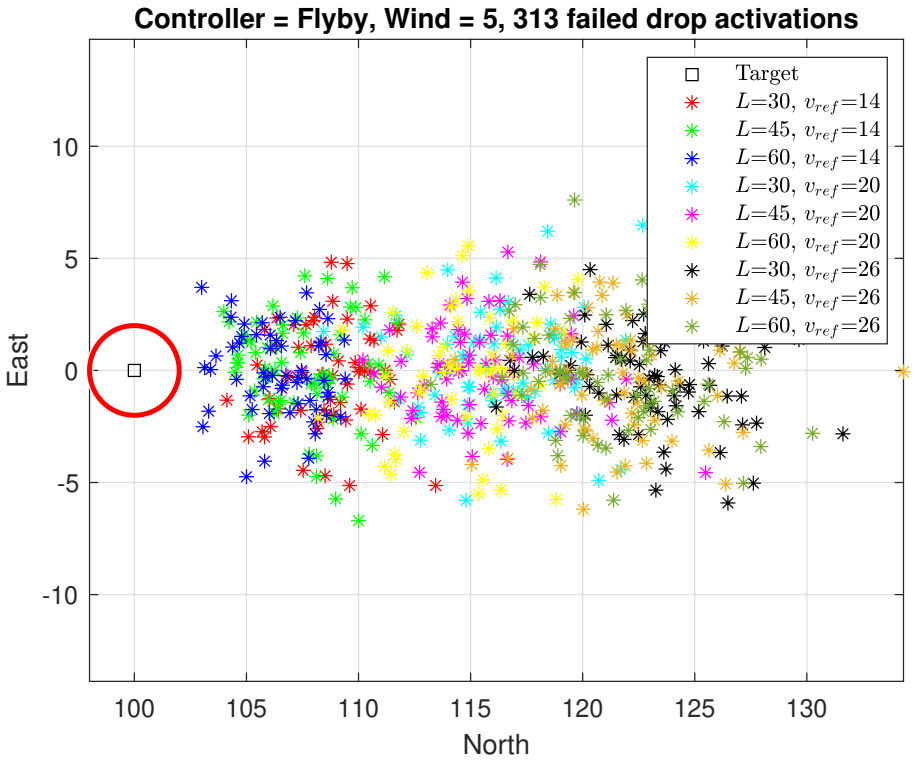

Figure C.2: Delivery position for straight flyby drop

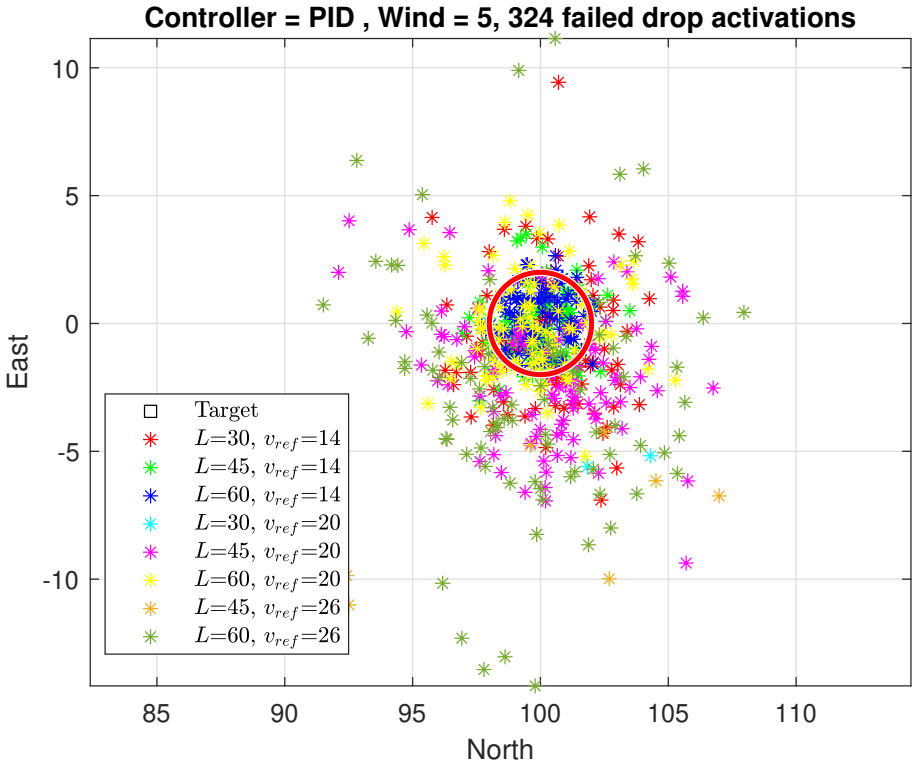

Figure C.3: Delivery position for PID

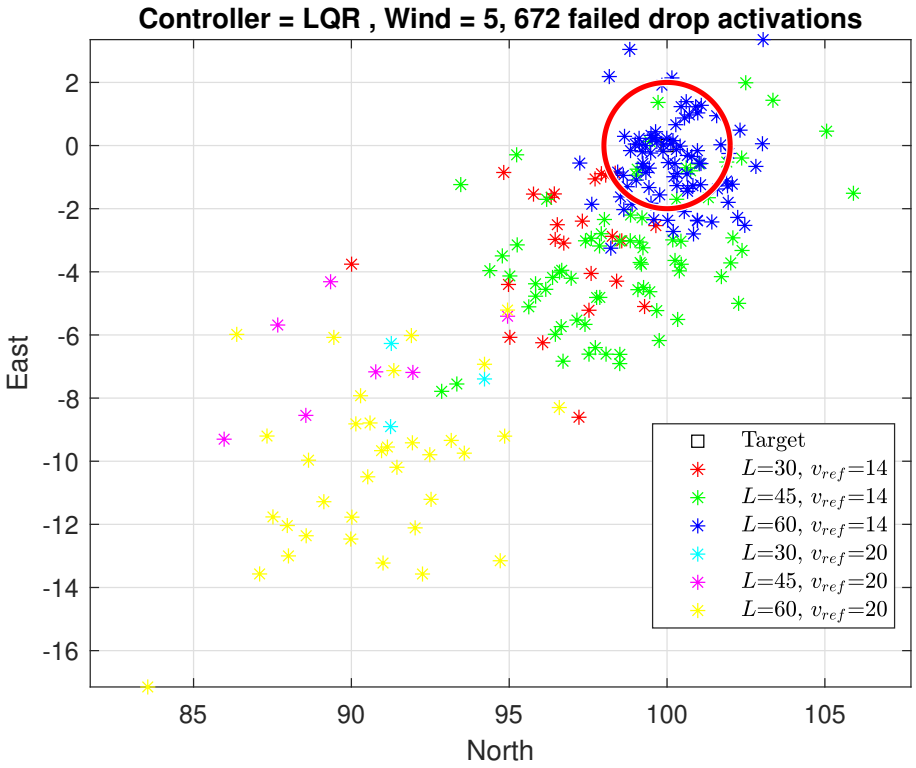

Figure C.4: Delivery position for LQR

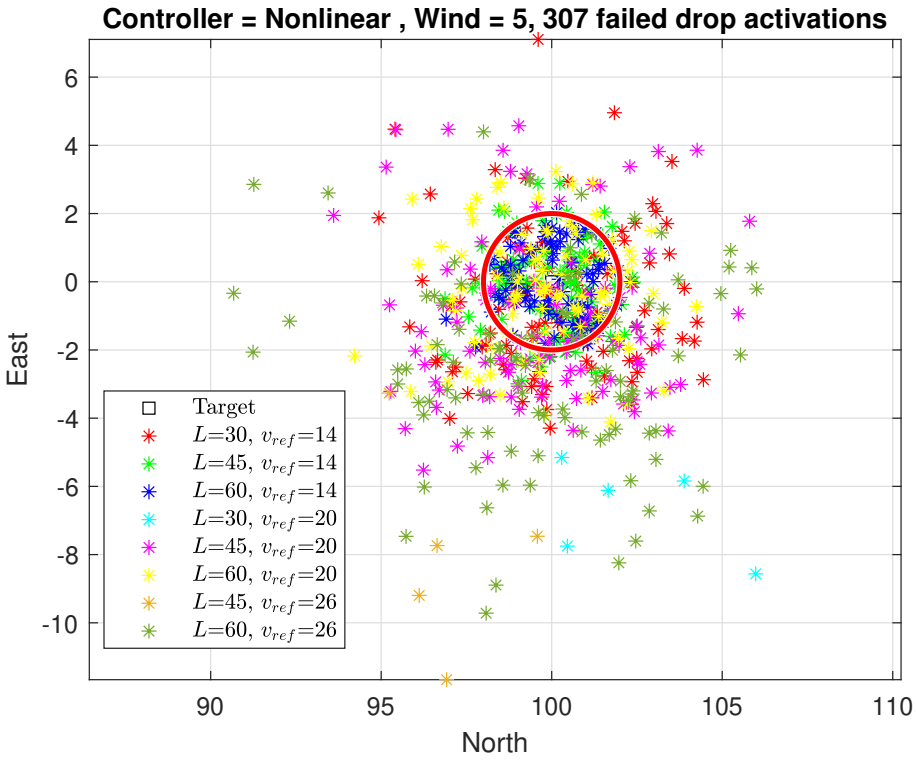

Figure C.5: Delivery position for Nonlinear feedback controller

C.2 Result plots from minimum and maximum package-radius in the wind test

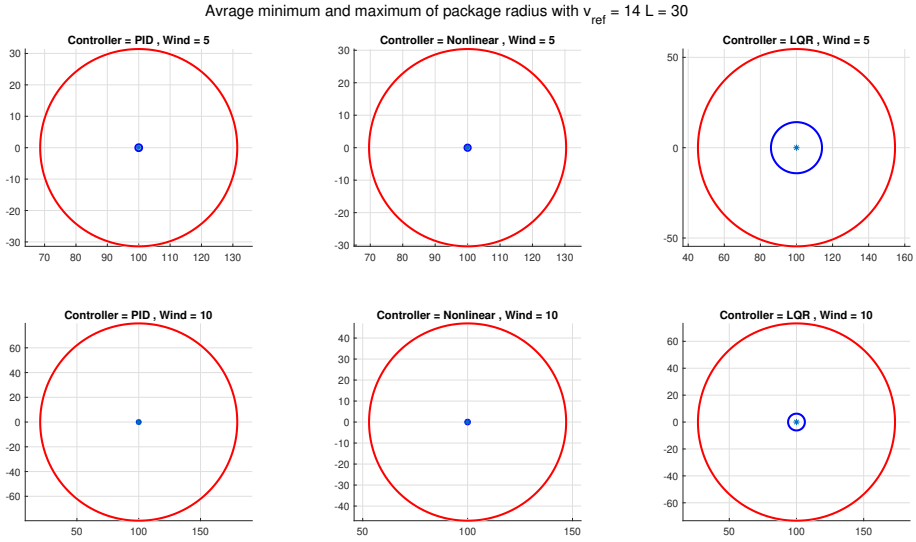

Figure C.6: Result average form minimum and maximum package-radius in wind test when L = 30 and $v_{ref} = 14$

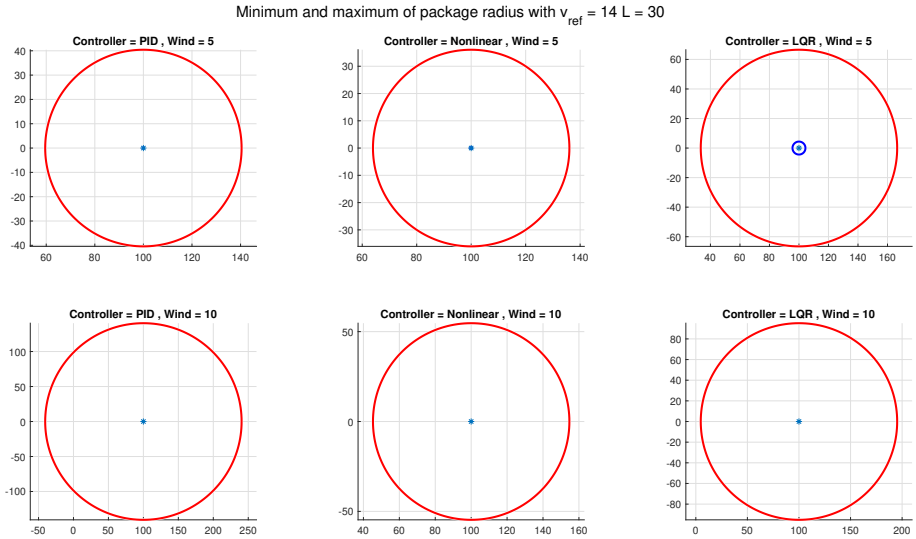

Figure C.7: Max and min value result form minimum and maximum package-radius in wind test when L = 30 and $v_{ref} = 14$

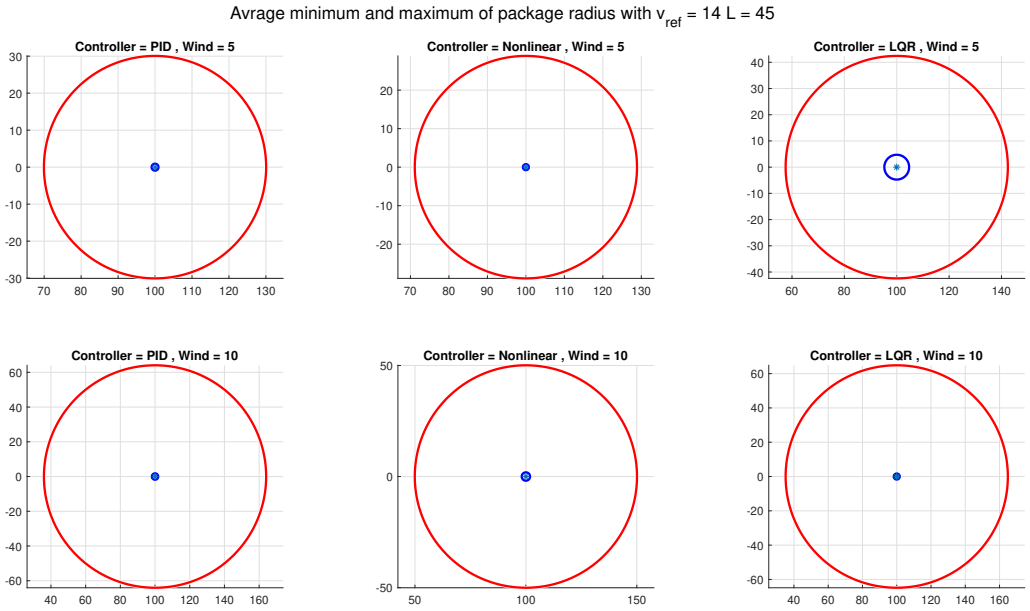

Figure C.8: Result average form minimum and maximum package-radius in wind test when L = 45 and $v_{ref} = 14$

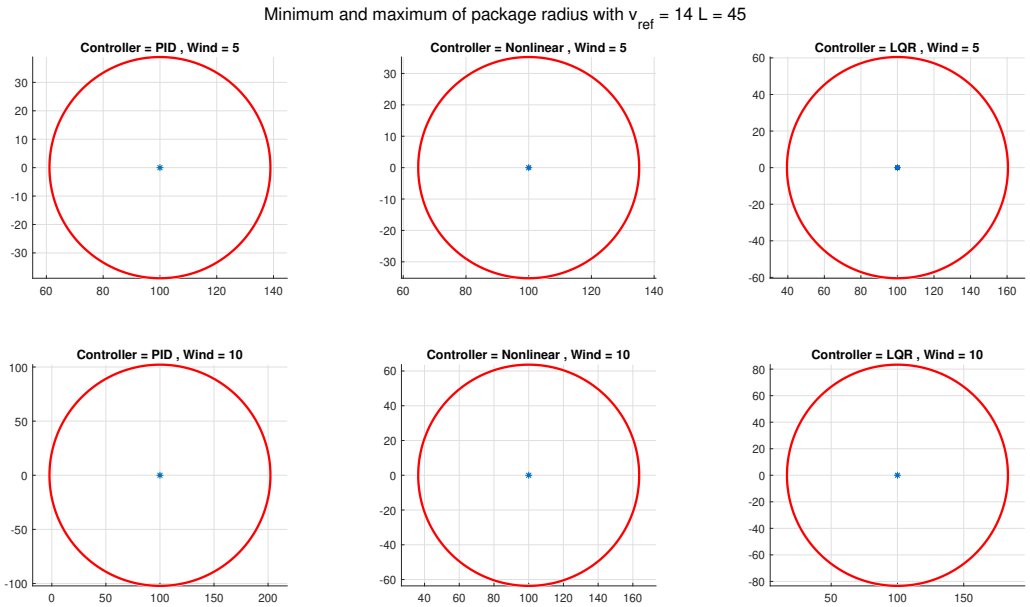

Figure C.9: Max and min value result form minimum and maximum package-radius in wind test when L = 45 and $v_{ref} = 14$

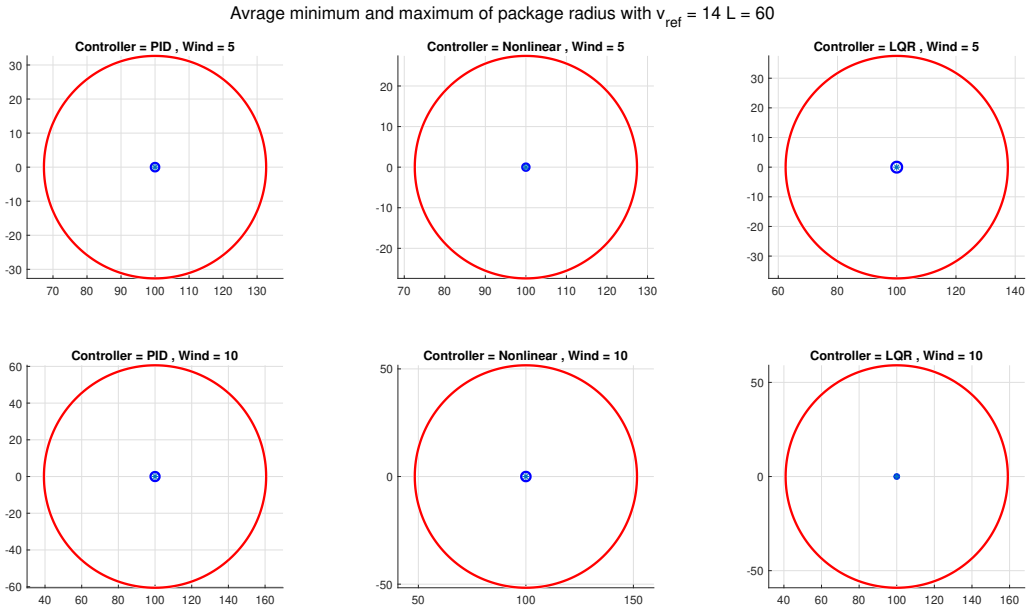

Figure C.10: Result average form minimum and maximum package-radius in wind test when L = 60 and $v_{ref} = 14$

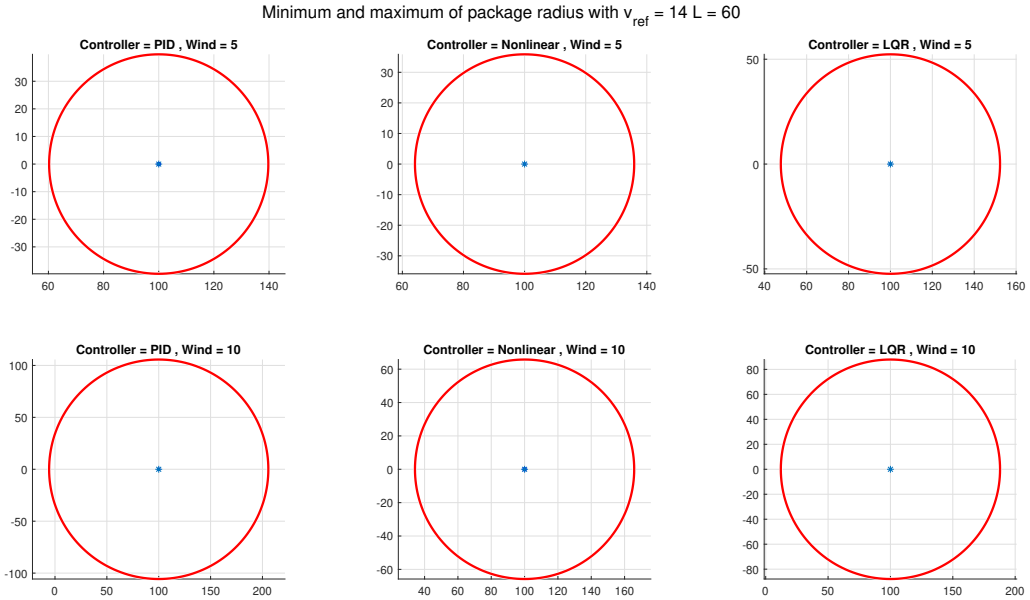

Figure C.11: Max and min value result form minimum and maximum package-radius in wind test when L = 60 and $v_{ref} = 14$

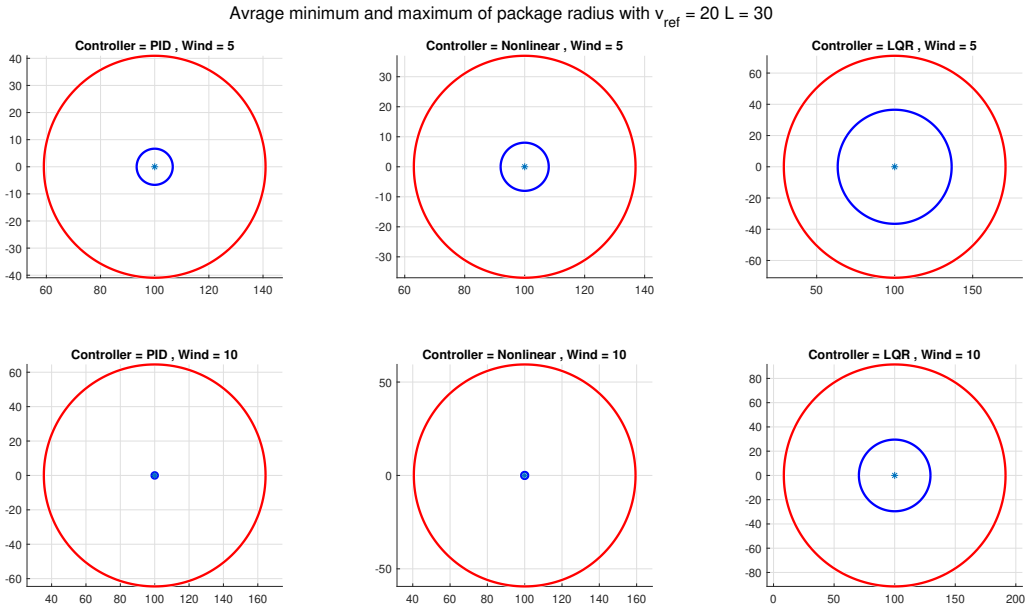

Figure C.12: Result average form minimum and maximum package-radius in wind test when L = 30 and $v_{ref} = 20$

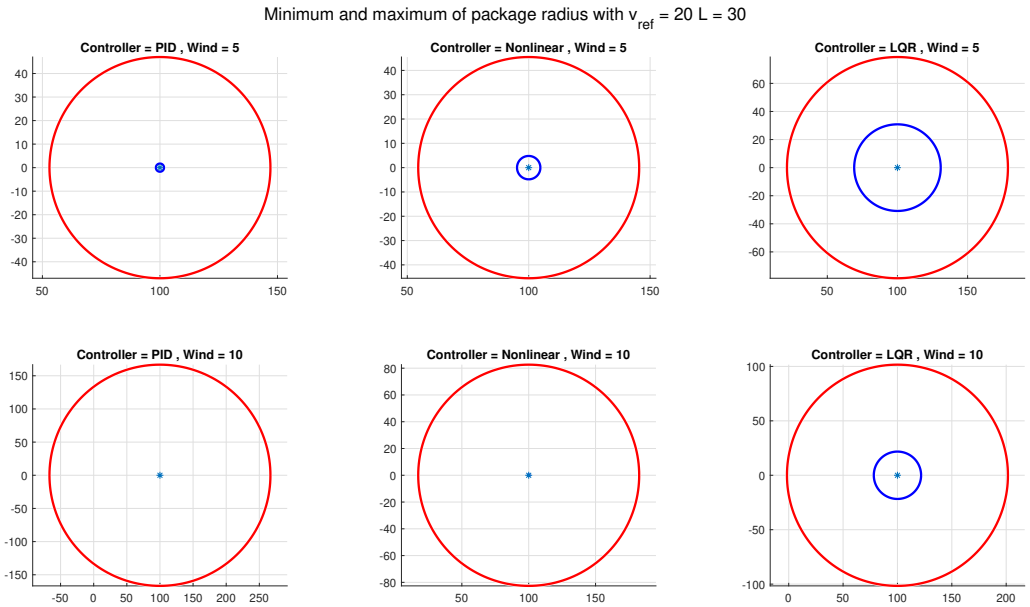

Figure C.13: Max and min value result form minimum and maximum package-radius in wind test when L = 30 and $v_{ref} = 20$

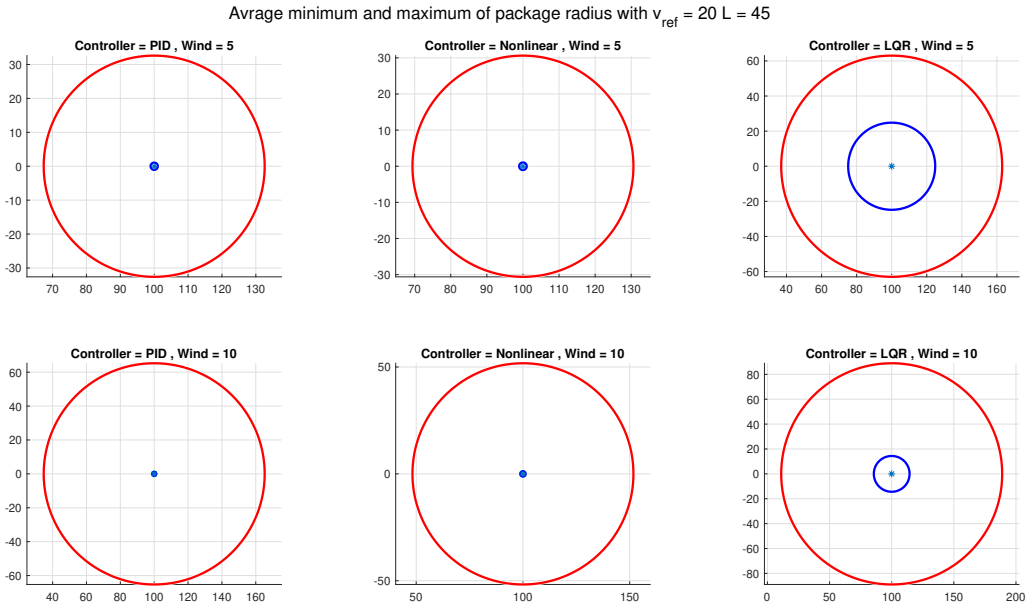

Figure C.14: Result average form minimum and maximum package-radius in wind test when L = 45 and $v_{ref} = 20$

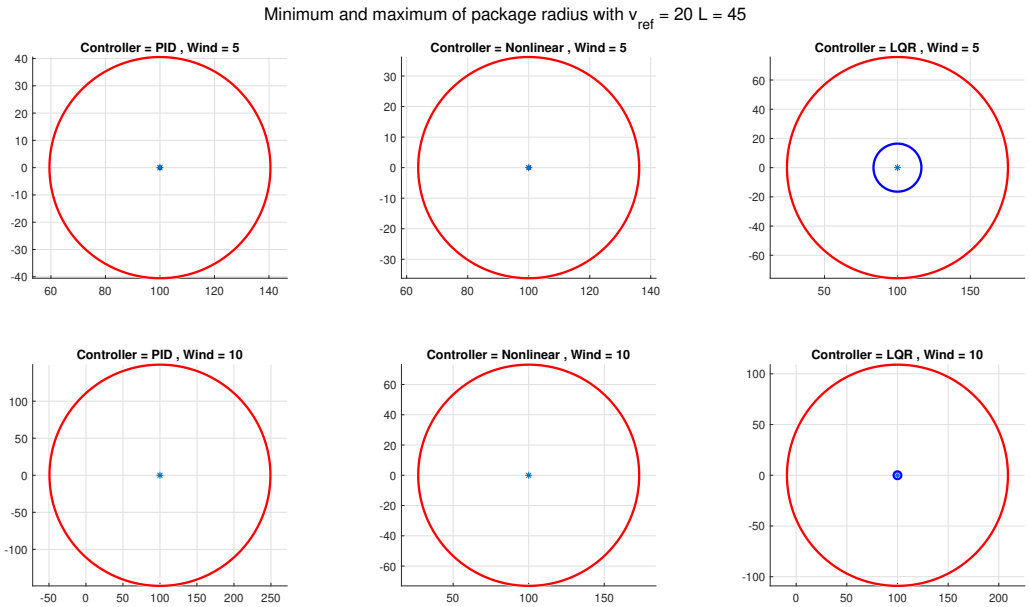

Figure C.15: Max and min value result form minimum and maximum package-radius in wind test when L = 45 and $v_{ref} = 20$

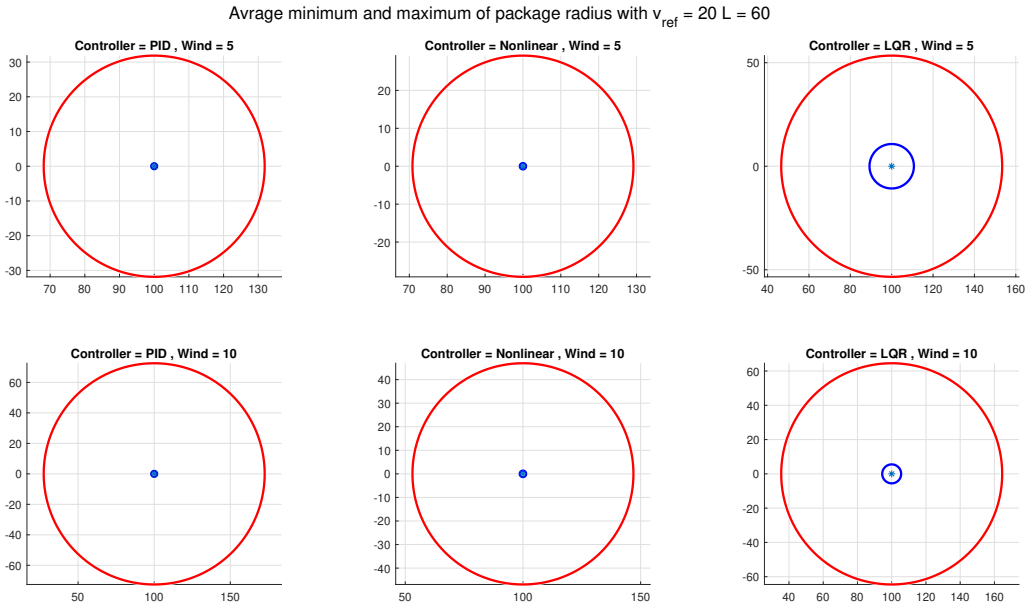

Figure C.16: Result average form minimum and maximum package-radius in wind test when L = 60 and $v_{ref} = 20$

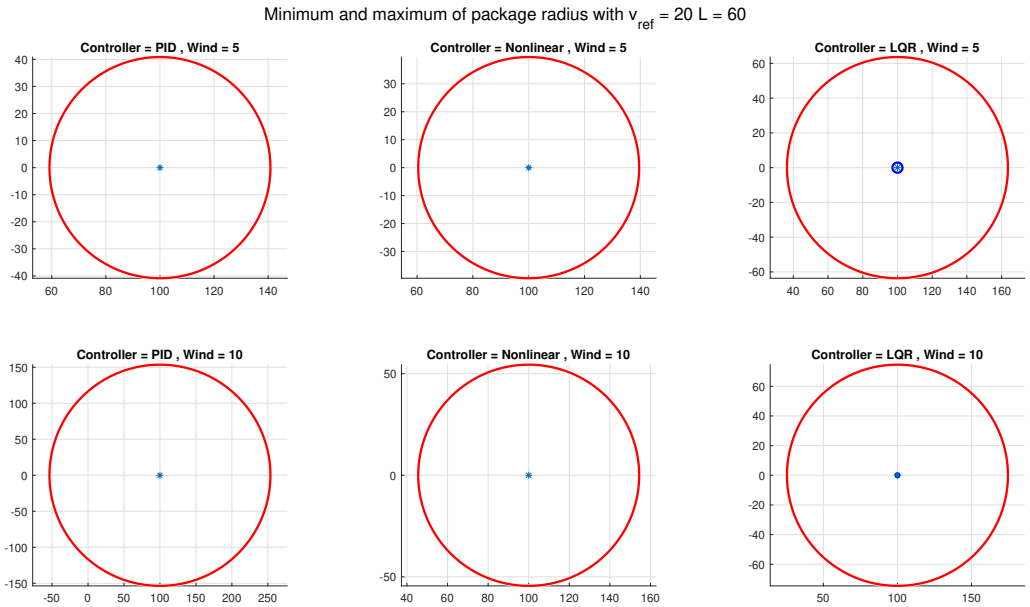

Figure C.17: Max and min value result form minimum and maximum package-radius in wind test when L = 60 and $v_{ref} = 20$

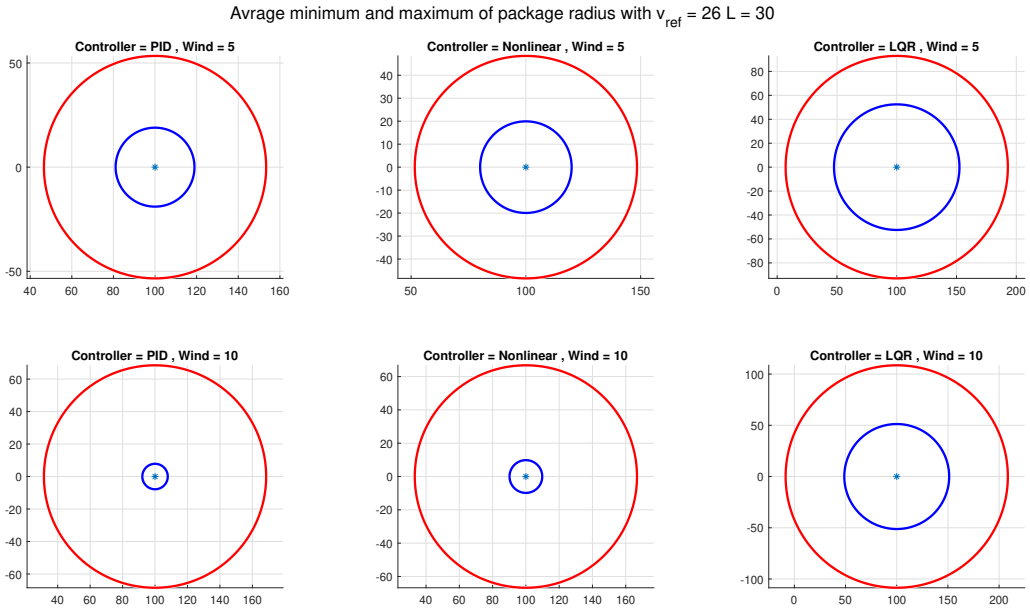

Figure C.18: Result average form minimum and maximum package-radius in wind test when L = 30 and $v_{ref} = 26$

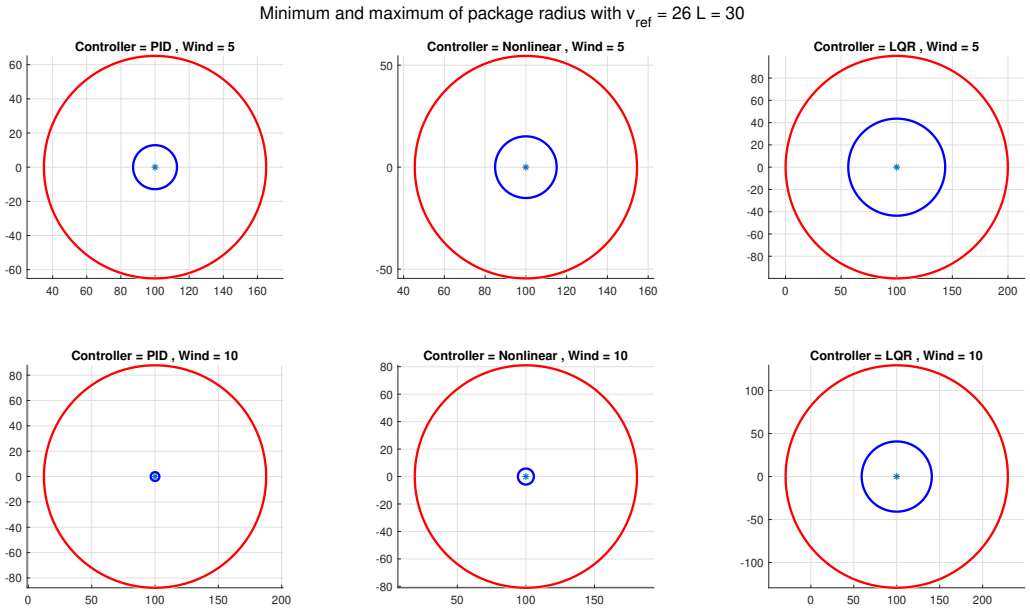

Figure C.19: Max and min value result form minimum and maximum package-radius in wind test when L = 30 and $v_{ref} = 26$

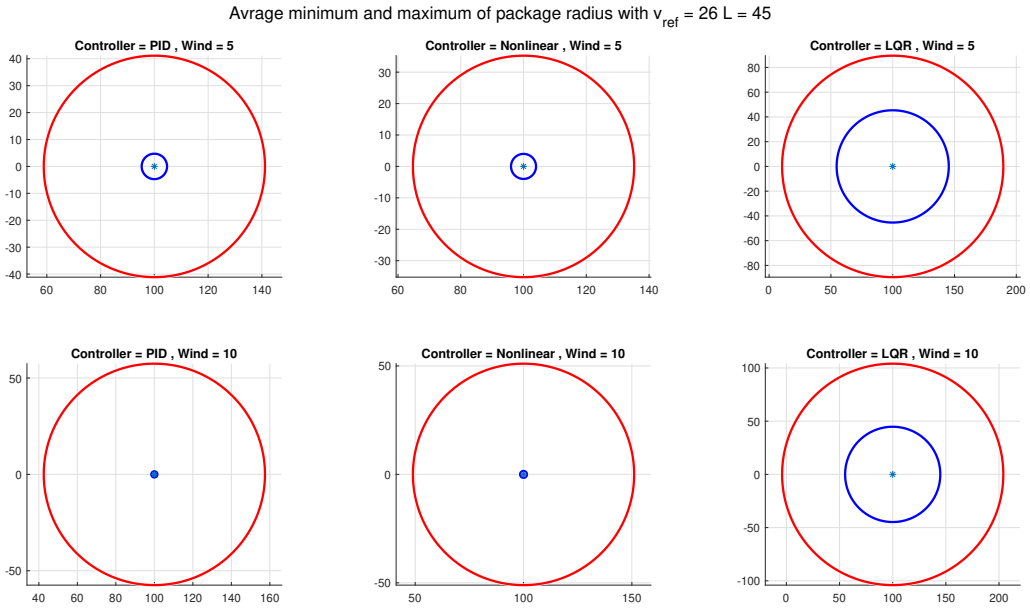

Figure C.20: Result average form minimum and maximum package-radius in wind test when L = 45 and $v_{ref} = 26$

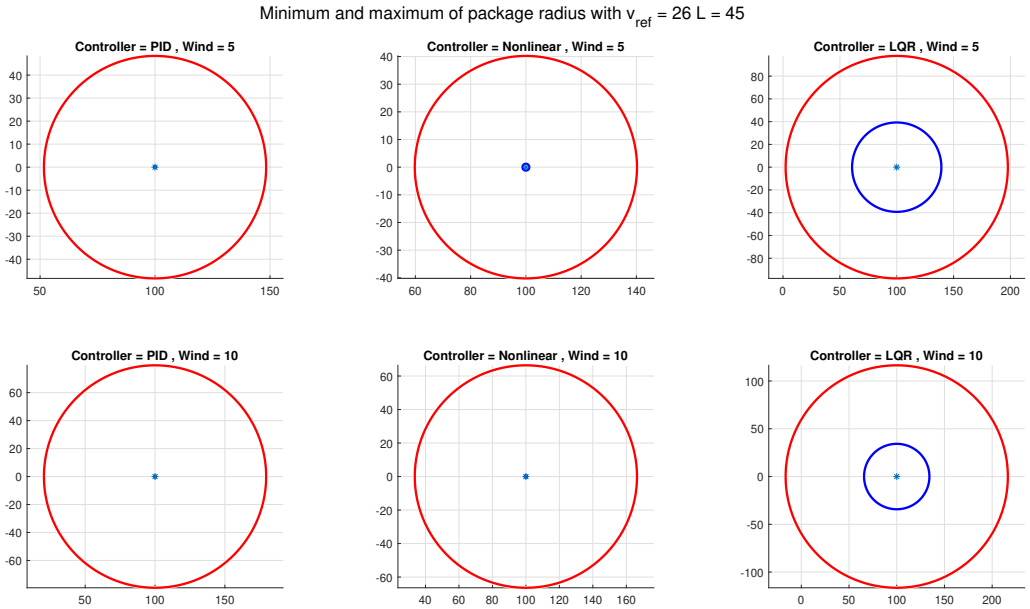

Figure C.21: Max and min value result form minimum and maximum package-radius in wind test when L = 45 and $v_{ref} = 26$

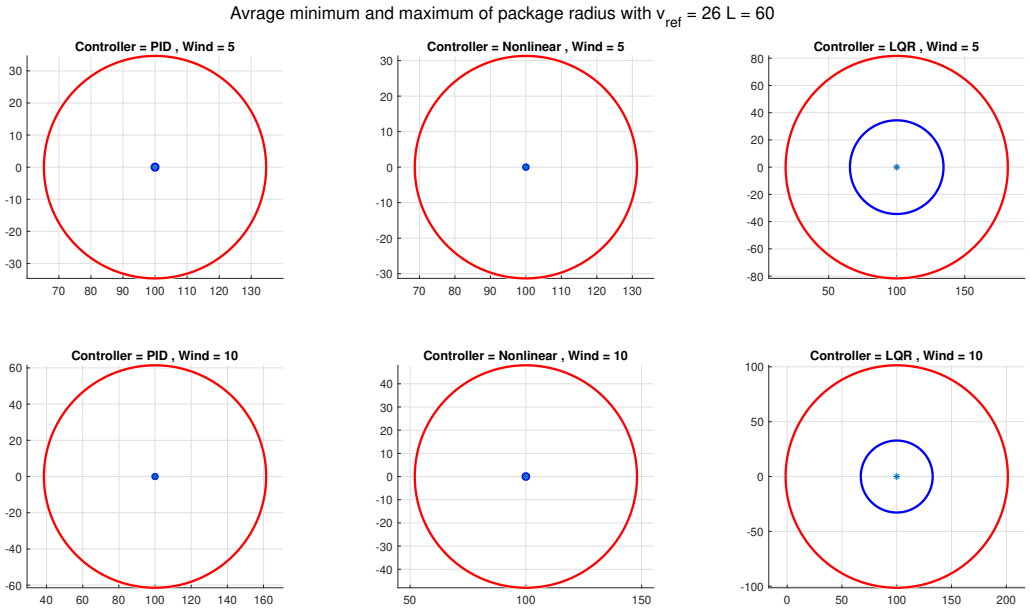

Figure C.22: Result average form minimum and maximum package-radius in wind test when L = 60 and $v_{ref} = 26$

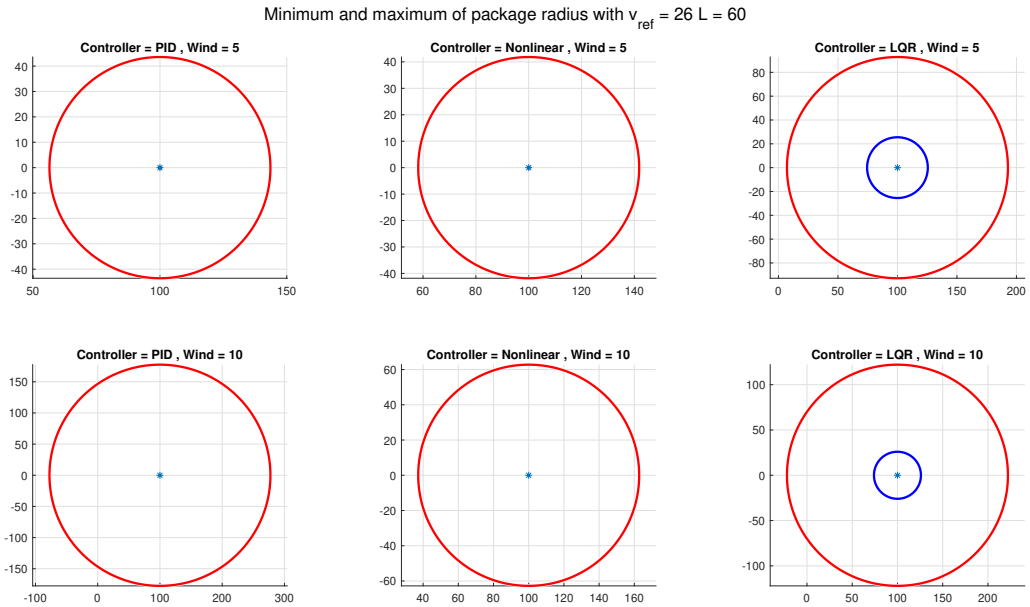

Figure C.23: Max and min value result form minimum and maximum package-radius in wind test when L = 60 and $v_{ref} = 26$

C.3 Figures for Drouge size

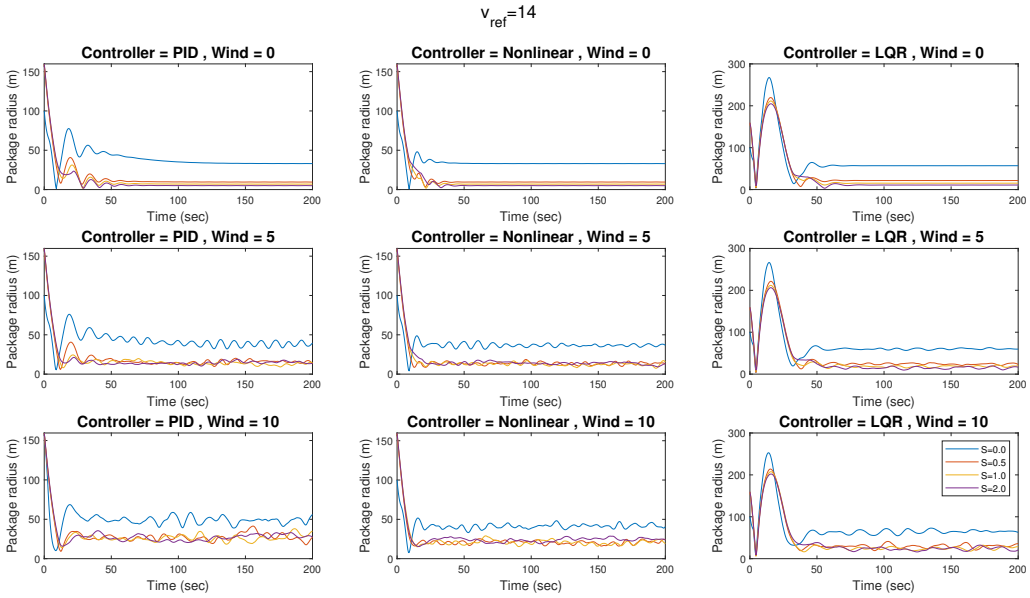

Figure C.24: Result from drouge test when $v_{ref} = 14$

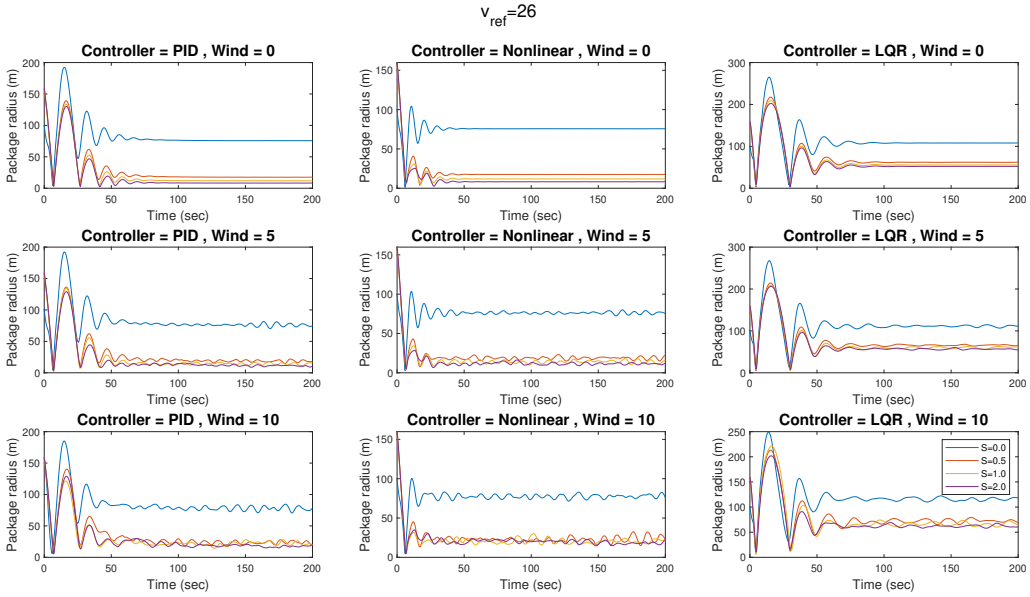

Figure C.25: Result from drouge test when $v_{ref} = 26$

C.4 Side wind plots

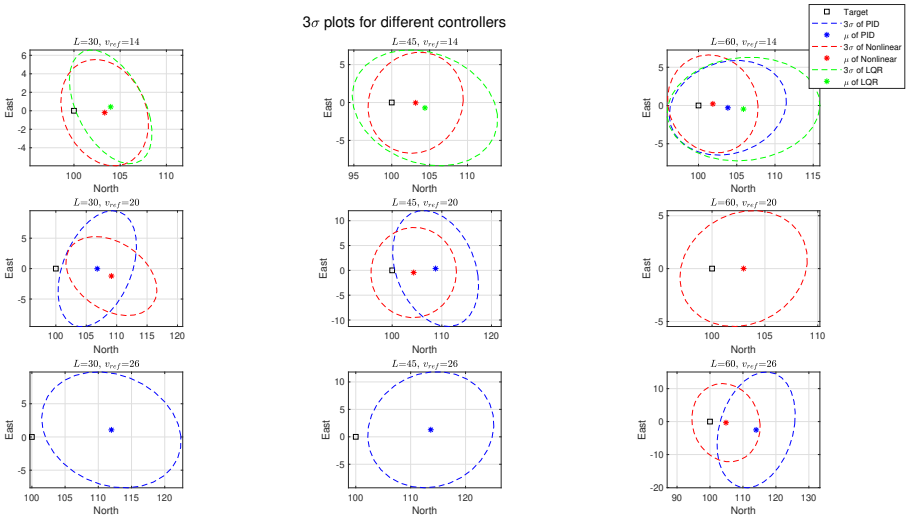

Figure C.26: 3σ plots from drop test with side wind

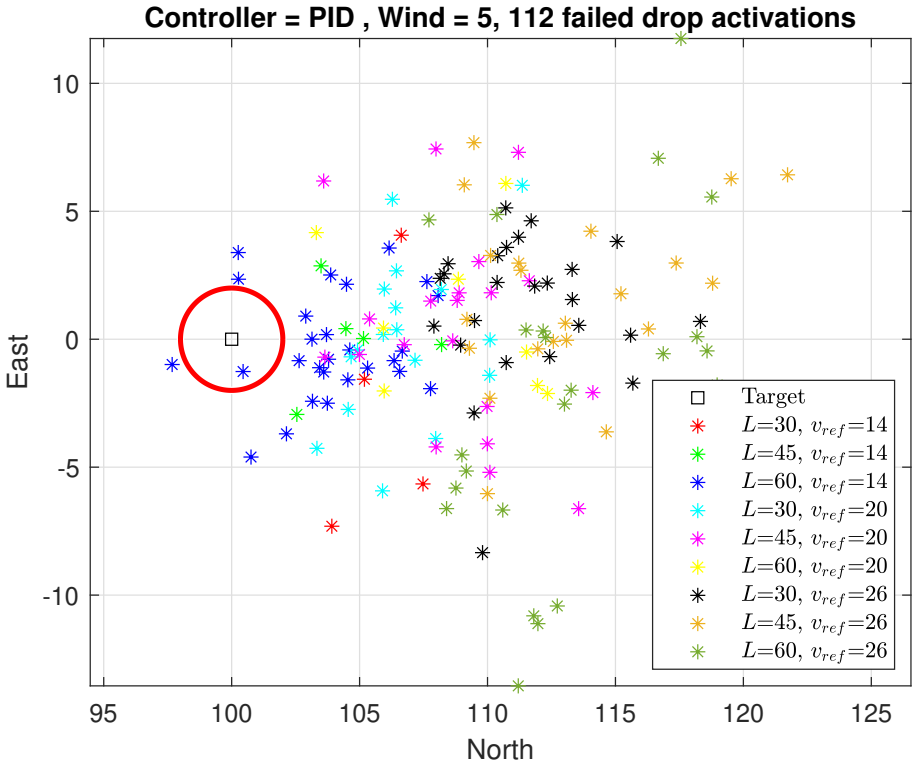

Figure C.27: Delivery position plot for PID in sidewind

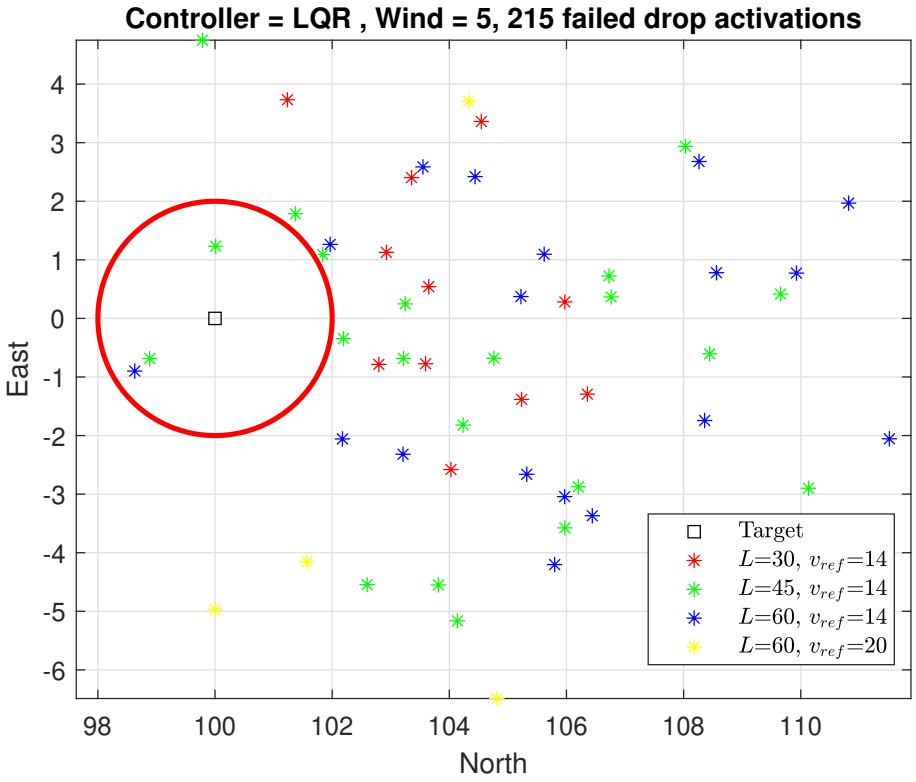

Figure C.28: Delivery position plot for LQR in sidewind

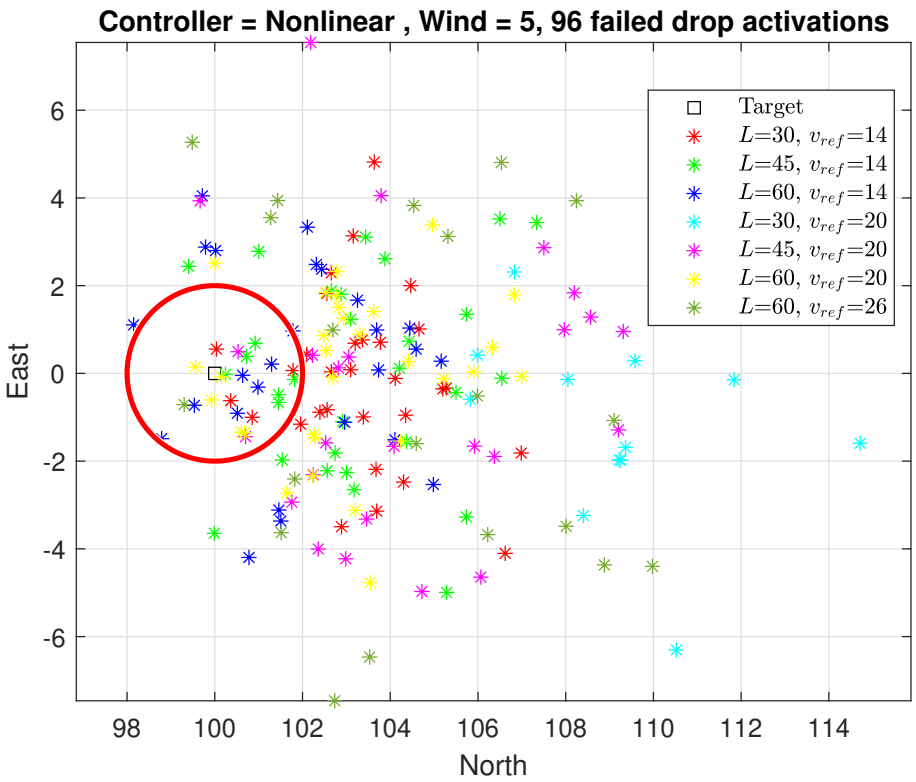

Figure C.29: Delivery position plot for Nonlinear controller in sidewind

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY