Robust Control and Learning for Autonomous Spacecraft Proximity Operations with Uncertainty

by

Charles E. Oestreich

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author . . . .
Department of Aeronautics and Astronautics
May 18, 2021

Certified by . . . .
Richard Linares
Charles Stark Draper Assistant Professor at MIT
Thesis Supervisor

Certified by . . . .
Ravi Gondhalekar
Senior Member of the Technical Staff at Draper
Thesis Supervisor

Accepted by . . . .
Zoltan Spakovszky
Chair, Graduate Program Committee

Robust Control and Learning for Autonomous Spacecraft Proximity Operations with Uncertainty

by

Charles E. Oestreich

Submitted to the Department of Aeronautics and Astronautics
on May 18, 2021, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

As the number of spacecraft and debris objects in orbit rapidly increases, active debris removal and satellite servicing efforts are becoming critical to maintain a safe and usable orbital environment. At the same time, future unmanned solar system exploration missions are targeting challenging destinations for scientific data collection. For practical realization of these technologies, the involved spacecraft must be highly autonomous and able to perform complex proximity operations maneuvers in a safe manner. This requires that the guidance and control system must reliably address inevitable sources of uncertainty while performing the maneuvers.

This thesis seeks to improve the flexibility and performance of autonomous spacecraft in uncertain scenarios by leveraging robust control theory and reinforcement learning. A novel algorithm, termed online tube-based model predictive control, is proposed and applied to a simulated mission involving the intercept of an tumbling target with unknown inertial properties. This algorithm demonstrates superior performance and exhibits less reliance on initial knowledge of the uncertainty when compared to standard robust control methods. Separately, reinforcement learning is utilized to develop a policy (to be employed as a feedback control law) for six-degree-of-freedom docking with a rotating target. The policy provides near-optimal performance in a simulated Apollo transposition and docking maneuver with uncertainty in the initial conditions. Both of these methods enhance the level of autonomy in their respective scenarios while also maintaining practical computational run-times. As such, this thesis represents an incremental step towards making missions based on highly autonomous proximity operations a reality.

Thesis Supervisor: Richard Linares
Title: Charles Stark Draper Assistant Professor at MIT

Thesis Supervisor: Ravi Gondhalekar
Title: Senior Member of the Technical Staff at Draper

Acknowledgments

This work was performed under funding from the Draper Fellow Program. The author is thankful for Draper's support and resources throughout this research effort.

I'd first like to thank my advisors Richard and Ravi. Richard, your optimism, big-picture thinking, and broad expertise have inspired me to think creatively about cutting-edge research ideas. Ravi, your precise technical knowledge, level-headed guidance, and tireless support have made this thesis a reality. You've instilled in me what it truly means to be a researcher.

Thanks to all of the friends and colleagues I've met over the past two years in Cambridge. Daniel, Adriana, Mario, and the rest of the ARCLab: the positive and relaxed lab culture you've all created has made the time enjoyable. I hope you didn't mind the loud ramblings on PPO and H-∞ control Daniel and I had in building 37. Thanks also to all of the GA^3 leaders, the sense of the community among the graduate students within the department is awesome. And of course, thanks to Connor (and Kiva!), Nico, Kathy, Cole, and the other MITOC community members that have made mountain adventures truly memorable.

I'd also like to thank Keenan, Tonio, Jess, Martin and the rest of the Astrobee team. Keenan, I've learned a lot from you on practical space robotics and I'm sure those late nights in the lab will pay off. I'd also like to thank Caroline and Roberto from DLR for their open sharing of results and data that are an integral part of this thesis.

Of course, I would absolutely not be here if it weren't for my family. Mom and Dad, thanks for raising me right and instilling a sense of drive in me that has gotten me through the past two years. Will, Elaine, and Kyle, it's the best time hanging out with you guys. Special extra thanks to Will for advice on graduate school, I definitely wouldn't be here without it.

Finally, thank you Morgan for being such an amazing partner. Running, coffee, music, and books are all just simply better when shared with you. You've made what should've been one of the worst years instead be one of the best.

Contents

List of Figures

List of Tables

Nomenclature

$A^{\pi}_{\boldsymbol{\eta}}$	advantage function
$\mathbf{A}$	state transition matrix
$\mathbf{A}^{*}_{\mathbf{w}}$	state transition matrix representation of the ARMA prediction model
$\mathbf{A}_{\mathcal{X}}, \mathbf{A}_{\mathcal{U}}$	polytopic state and control input constraint “A” matrices
$\mathcal{A}$	action space
a	autoregressive parameter value in ARMA model
$\mathbf{B}$	input distribution matrix
$\mathbf{b}_{\mathcal{X}}, \mathbf{b}_{\mathcal{U}}$	polytopic state and control input constraint “b” vectors
C	chaser body frame
c	moving average parameter value in ARMA model
d	successful dock reward term coefficient
$\mathcal{E}$	ellipsoidal set
$\mathbf{F}$	thrust command
$g(\mathbf{x})$	piecewise function for the successful dock reward
h	collision reward term coefficient
J	online tube-MPC performance cost metric

$J_{\boldsymbol{\theta}}$	PPO policy objective function
$\mathbf{J}$	inertia tensor
$\widehat{\mathbf{J}}$	estimated inertia tensor
$\widetilde{\mathbf{J}}$	inertia tensor estimate error
KL_{des}	desired KL-divergence value between policy updates
$\mathbf{K}$	LQR reference gain value for reinforcement learning reward function
$\mathbf{K}_A$	ARMA model parameter update gain
$\mathbf{K}_r$	exogenous input rejection gain
k	discrete time-step index
$L_{\boldsymbol{\eta}}$	PPO state-value loss function
$\mathbf{L}$	torque command
$\mathbf{M}$	matrix variable for mRPI ellipsoidal approximation
m	control input dimension
N	number of steps in the prediction horizon
n	state dimension
n_a	number of autoregressive parameters in the ARMA model
$\mathbf{P}$	quadratic terminal state cost matrix
$\mathbf{P}_e$	ellipsoid shape matrix
$\mathbf{p}_e$	ellipsoid center
$\mathcal{P}$	polyhedron set
$p_{\boldsymbol{\theta}}$	policy probability ratio

$p_{\mathbf{w}}$	probability that the estimated bounded set of exogenous inputs contains the true exogenous input
$\mathbf{Q}$	quadratic state error cost matrix
$\mathbf{Q}_a$	quadratic state error cost matrix (attitude state only)
$\mathbf{Q}_t$	quadratic state error cost matrix (translational state only)
$\mathbf{q}$	attitude quaternion
$\mathbf{q}_v$	vector quaternion component
q_w	scalar quaternion component
$\mathbf{R}$	quadratic control input cost matrix
$\mathbf{R}_Y^X$	rotation matrix; attitude of frame Y with respect to X
$\bar{\mathbf{R}}_Y^X$	predicted attitude via estimated inertia tensor
$\bar{\mathbf{r}}$	desired position
$\mathbf{r}$	position
$\mathbf{r}_p$	relative position between docking ports
$\mathbf{r}_{dc}, \mathbf{r}_{dt}$	position of docking port in the spacecraft body frame
r	reinforcement learning rewards
r_{col}	maximum possible distance for a collision
$\mathcal{S}$	state space
s	covariance scaling factor for the estimated set of exogenous inputs
T	target body frame
T_s	discrete sampling time

$\mathbb{T}$	set of all possible state-action pair trajectories
t_f	final trajectory time-step
$\bar{\mathcal{U}}$	shrunk control input constraints for tube-based MPC
$\mathbf{u}$	control input
$\mathcal{U}$	control input constraints
u_p	control limit magnitude
$V_{\boldsymbol{\eta}}^{\pi}$	state-value function
$\bar{\mathbf{v}}$	desired velocity
$\mathbf{v}$	velocity
$\mathbf{v}_p$	relative velocity between docking ports
W	world frame
$\mathbf{w}$	exogenous input, i.e., the uncertainty
$\mathbf{w}^*$	stack of predicted exogenous inputs
$\mathcal{W}$	bounded exogenous input set
$\widehat{\mathbf{w}}$	estimated exogenous input
$\widehat{\mathcal{W}}$	estimated bounded exogenous input set
$\widetilde{\mathbf{w}}$	exogenous input estimation error
$\bar{\mathcal{X}}$	shrunk state constraints for tube-based MPC
$\bar{\mathbf{x}}$	desired state
$\mathbf{x}$	state
$\mathbf{x}'$	translational component of the state

$\mathcal{X}$	state constraints
$\mathbf{Y}$	matrix variable for mRPI ellipsoidal approximation
y	generic ARMA model output
$\bar{\mathbf{z}}$	desired nominal state
$\mathbf{z}$	nominal state
$\mathcal{Z}$	approximate minimum robust positively invariant set, i.e., the “tube”
$\mathcal{Z}^*$	true minimum robust positively invariant set
α	design parameter for mRPI ellipsoidal approximation
$\boldsymbol{\alpha}$	attitude and angular velocity error vector
β	control limit allocation percentage for mRPI ellipsoidal approximation
$\beta_{\boldsymbol{\eta}}$	state-value function learning rate
$\beta_{\boldsymbol{\theta}}$	policy learning rate
ϵ	ARMA model prediction error caused by white noise
$\boldsymbol{\eta}$	reinforcement learning state-value function parameter vector
γ	reward discount factor
λ	forgetting factor for estimating the next exogenous input
λ_A	forgetting factor for updating the exogenous input prediction model
$\boldsymbol{\mu}_{\mathbf{w}}$	mean exogenous input
$\boldsymbol{\nu}$	nominal control input
$\boldsymbol{\Omega}$	angular velocity matrix for quaternion kinematics
$\bar{\boldsymbol{\omega}}$	predicted angular velocity via estimated inertia tensor

$\boldsymbol{\omega}$	angular velocity
$\boldsymbol{\phi}$	ARMA regression vector for recursive least-squares updates
ϕ	Euler angle for roll
$\pi_{\boldsymbol{\theta}}$	reinforcement learning policy
$\widehat{\pi}_{\boldsymbol{\theta}}$	reinforcement learning policy prior to a PPO update
ψ	Euler angle for yaw
ρ	spectral radius
$\boldsymbol{\Sigma}_{\mathbf{w}}$	exogenous input covariance
$\widehat{\boldsymbol{\Sigma}}_{\mathbf{w}}$	estimated exogenous input covariance
$\boldsymbol{\tau}$	trajectory of state-action pairs
$\boldsymbol{\theta}$	reinforcement learning policy parameter vector
$\boldsymbol{\theta}_A$	ARMA model parameter vector
$\boldsymbol{\theta}_{\mathbf{w}}$	full set of ARMA model parameters for exogenous input prediction
θ	Euler angle for pitch
ξ	PPO clipping parameter
ζ	sum of geometric progression

Acronyms

ARMA	autoregressive moving average
CSM	command service module
DAP	digital autopilot
DOF	degrees of freedom
ENTRY	orbital re-entry control system
ISS	International Space Station
LM	lunar module
LQR	linear quadratic regulator
MEV	Northrop Grumman's Mission Extension Vehicle
MPC	model predictive control
mRPI	minimum robust positively invariant set
OSAM-1	NASA's On-Orbit Servicing, Assembly, and Manufacturing-1 mission
PPO	proximal policy optimization

QCQP	quadratically constrained quadratic program
QP	quadratic program
RCS	reaction-control system
RL	reinforcement learning
RPI	robust positively invariant
SDP	semi-definite programming
SOCP	second-order cone programming
SPHERES	Synchronized Position Hold Engage and Reorient Experimental Satellites
TVC	thrust-vector-control system

Chapter 1

Introduction

Earth-orbiting spacecraft perform a variety of critical functions, such as providing reliable communication links or supporting climate science via remote sensing. As launch systems improve in both terms of efficiency and cost, it is expected that the number of objects in Earth orbit will continue to grow rapidly [64]. This in turn exacerbates already-crowded orbits and increases the risk of catastrophic collisions between satellites. To address these impending challenges, *active debris removal* and *on-orbit servicing* missions are seen as a critical aspect of future space operations as a whole [46].

Active debris removal missions involve a “chaser” spacecraft maneuvering to capture and de-orbit a debris object (the “target”), which could be a used rocket stage, defunct satellite, or any generic piece of “space junk”. The physical method of de-orbiting or capturing the target object can vary greatly depending on mission design [38]. However, nearly all mission designs require the chaser spacecraft to perform precise, close-range maneuvers that approach the target in order to prepare for capture or de-orbiting operations. These maneuvers are included under the general term of *proximity operations*, which also includes well-known cooperative docking maneuvers such as the cargo vehicles performing a rendezvous with the International Space Station [15].

On-orbit servicing missions aim to re-fuel, repair, or perform any other operations that assist and extend the lifespan of a currently operating satellite [16]. By extending the lifetime of existing satellites, on-orbit servicing diminishes the need of launching new replacement satellites and also limits the number of new space objects. Perhaps the most notable on-orbit servicing operations were carried out in the Hubble Space Telescope repair missions [26]. Using the Space Shuttle to perform proximity operations, astronauts were able to successfully repair the faulty mirror and restore nominal operations to the telescope.

Proximity operations also arise in many solar system exploration missions. This is most notable for asteroid exploration missions, where the spacecraft must perform precise orbit and landing maneuvers to gather useful scientific data [6]. These missions are especially challenging due to the highly uncertain environment of operations and significant time delays with Earth communication systems.

While proximity operations have a significant history in terms of manned missions, *autonomous* proximity operations have rarely been performed. Autonomous proximity operations that do not involve astronauts and minimize human operations in general are desirable for several key reasons: 1) robotic spacecraft are much less expensive than human-rated spacecraft, 2) less human oversight can offer more efficient and widespread servicing/debris removal capabilities, and 3) the risk that is associated with astronaut-based missions is eliminated. Full autonomy is sometimes a strict requirement for cases with large communication delays, such as a Mars landing. As such, there are many entities that are developing spacecraft and algorithms to enable effective, autonomous proximity operations. This thesis is focused on guidance and control algorithms that enable autonomous proximity operations to occur despite uncertainty in the particular mission scenario.

1.1 Background on Autonomous Proximity Operations

Regardless of the mission type, autonomous proximity operations generally involve a chaser spacecraft that performs precise maneuvers around the target. These maneuvers can include docking with the target, orbiting the target to inspect it, grappling the target with a robotic arm, etc. While the degree of human oversight can vary based on the specific mission, generally speaking, the chaser spacecraft must be able to autonomously navigate relative to the target and utilize guidance and control algorithms to safely perform the required maneuvers. This thesis only focuses on the guidance and control aspects of proximity operations; it is generally assumed that navigational state information on both the chaser and the target is readily available.

There are several autonomous on-orbit servicing missions currently being operated or planned for the near-future. Northrop Grumman's Mission Extension Vehicle (MEV) performed the first docking maneuver with a commercial satellite in 2020 [51]. This mission was performed to relocate the Intelsat 901 satellite, but was largely performed via tele-operation (i.e., humans in-the-loop). NASA's On-Orbit Servicing, Assembly, and Manufacturing-1 (OSAM-1) mission is planned to autonomously refuel a satellite in a low-Earth polar orbit [9]. While the spacecraft will perform some portion of maneuvers autonomously, the mission is once again designed in the framework of having humans in-the-loop. Similarly, the Robotic Servicing of Geosynchronous Satellites mission led by DARPA is planned to service multiple satellites located in the geostationary orbit [59].

There are fewer missions in development for active debris removal. The ESA is developing the RemoveDebris mission, which intends to demonstrate autonomous rendezvous and capture technologies needed for active debris removal [17]. This is part of a larger ESA effort in active debris removal called "e.Deorbit", which intends to eventually de-orbit Envisat, one of the largest, defunct satellites that poses a collision hazard in low-Earth orbit [61]. Active debris removal generally requires highly autonomous capabilities or long mission time-frames to deal with the uncertain properties of debris objects.

Apart from full missions in development, there are a wide range of research efforts to improve various phases of autonomous proximity operations. A selection of key research advancements in the field is covered in the literature review (Section 1.3). One of the most pressing challenges yet to be resolved is to safely and effectively address the inherent uncertainty present in many missions involving proximity operations. This challenge is currently handled by enabling humans to tele-operate or take control from the autonomous system if necessary in risky or uncertain situations. However, to truly enhance the active debris removal and on-orbit servicing missions, fully autonomous spacecraft are necessary. This thesis focuses on specific guidance and control algorithmic developments that contribute to making robotic spacecraft more reliable and autonomous despite inherent uncertainty.

1.2 Sources of Uncertainty

There are a wide range of factors that can potentially contribute to the overall uncertainty in an autonomous proximity operations mission. Generally speaking, sources of uncertainty can be classified as either *aleatoric* or *epistemic* [24]. The term aleatoric refers to unknown values that simply cannot be precisely determined by their very nature. On the other hand, the term epistemic refers to unknown values that *could* be known if more information was available in the mission scenario.

Autonomous spacecraft performing proximity operations must deal with both forms of uncertainty. Sensor noise (aleatoric) affects the accuracy of navigational state estimates during maneuvers. Similarly, thruster noise (also aleatoric) will produce slightly different chaser dynamics than expected. In general, aleatoric uncertainty is a stochastic process that is always present and can usually be practically handled via feedback control. However, in particular scenarios where the uncertainty may be significant or safety is paramount, the control system for performing maneuvers must be *robust* to the uncertainty, guaranteeing stability and constraint satisfaction.

Epistemic uncertainty is also common in proximity operations scenarios; however, it usually arises from sources specific to the mission scenario. Epistemic uncertainty is synonymous with *unmodeled dynamics*, in that there is a way to model the uncertainty, but its often too complex or simply requires information that is not available at hand. As such, epistemic uncertainty contains a deterministic but unknown component that is not covered by the nominal model the controller uses.

In future autonomous proximity operations, epistemic uncertainty sources will be highly significant. Examples include fuel slosh, flexible (i.e., non-rigid) dynamics, unknown mass/inertial properties of the target, solar radiation pressure disturbances while orbiting an asteroid, etc. Robust control techniques can address epistemic uncertainty in the same manner as aleatoric uncertainty, but do not consider potential performance gains that could be had by identifying characteristics or a dynamic model of the uncertainty. Additionally, *adaptive* controllers are specifically designed to alter control parameters based on the real, online dynamics experienced during the trajectory.

This thesis is generally focused on combining the benefits of robust control with online adjustments and learning to appropriately handle the uncertainties associated with fully autonomous proximity operations. The next section shares a selection of previous research in guidance and control techniques for autonomous proximity operations, including some works that directly address the challenge of uncertainty.

1.3 Literature Review

Autonomous proximity operations pose a number of guidance and control challenges [57]. Chief among these is reliably ensuring safe and successful maneuvers, as the missions typically have high risk and cost. Spacecraft maneuvering is also highly constrained, especially in terms of actuator capability since most spacecraft have a low thrust-to-mass ratio. Performing optimal maneuvering trajectories is also highly desirable, whether the objective is to minimize fuel usage or some other performance metric. If performing a docking or intercept maneuver, the target object may be tumbling, which in turn adds greater complexity to the trajectory. Finally, in addition to the numerous epistemic uncertainty examples covered above, autonomous fault tolerance is also desirable as it enhances the reliability of the chaser spacecraft.

Many research efforts have targeted one or several of these challenges. A selection of key works and their contributions are included below. Current flight approaches are discussed first, followed by more advanced techniques that are mostly untested onboard real spacecraft. Research works that directly address uncertainty in the context of proximity operations are given special attention.

1.3.1 Current Flight Approaches

Some of the earliest autonomous guidance and control technologies for proximity operations were implemented in the Apollo missions. The command service module (CSM) had three main automatic control systems: the orbital re-entry digital autopilot (ENTRY DAP), the reaction-control system digital autopilot (RCS DAP), and the thrust-vector-control digital autopilot (TVC DAP) [63]. All of these were largely based on phase-plane control logic. Phase-plane logic commands discrete on-off control inputs by defining switching curves in the 2-D space of the state error and its rate [57].

The ENTRY DAP performed the entire re-entry maneuver after jettison of the service module and used phase-plane logic controllers to command thrusters that provided the necessary lift to reach the landing zone. The RCS DAP also utilized phase-plane logic for controlling the CSM's attitude in most flight stages. Since the CSM had sixteen thrusters arranged in four quads around the spacecraft, the control system also required logic to translate the desired thrust/torque commanded by the controller into suitable individual thruster firings. The TVC DAP utilized phase-plane logic in order to control the CSM during main-engine burns, commanding gimbal servos of the main engine while also controlling the RCS thrusters. These three autonomous control systems were highly successful despite their relatively simple design; however, the ability of human intervention to correct for errors was always available and thus they do not constitute a fully autonomous system.

The most common autonomous proximity operations occurring in the present day are the rendezvous and docking maneuvers performed by visiting vehicles to the International Space Station (ISS). These spacecraft perform a series of phasing, far-range rendezvous, and close-range approach maneuvers before they reach a final berthing position in close proximity to the station [15]. From there, the spacecraft performs the final docking maneuver (as with the Soyuz spacecraft) or is captured by the ISS Canadarm (which is operated by astronauts onboard the station). The full pipeline of these maneuvers is meticulously planned in advance and has significant human oversight at each stage, with hold points in between stages to make manual corrections. Thus, the spacecraft can rely on simple control systems to follow the trajectory, as large sources of uncertainty are not handled by the spacecraft itself.

Current and near-future missions such as Northrop Grumman's MEV and NASA's OSAM-1 also have a similar framework of pre-planned trajectories followed by simple controllers with significant human oversight, thus making them, in general, low-autonomy proximity operations. Since space missions have high risk and cost associated with them, these incremental steps in on-orbit technology demonstrations are necessary and significant. Meanwhile, there have been many research efforts that propose guidance and control frameworks for highly autonomous proximity operations.

1.3.2 Trajectory Optimization and Tracking Control

Trajectory Optimization

Highly autonomous frameworks for proximity operations often split the guidance and control aspects into modular components. The guidance component typically consists of an optimization algorithm that provides the reference trajectory that minimizes a cost function, satisfies constraints, and is dynamically feasible. Since the optimization problem can often involve nonlinear dynamics, non-convex constraints, and a high-dimension state space, solving times can be significant (especially on low-performance spacecraft computing hardware). As such, the solving algorithms are typically implemented in an open-loop fashion, i.e., computed once using the available information and not updated or corrected by future information.

Trajectory optimization algorithms for guidance can vary depending on the specifically target mission scenario. Breger and How [10] developed a trajectory optimization method that enforced passive safety constraints (i.e., guaranteeing collision avoidance if the chaser loses actuation authority). Boyarko et al. [7] computed minimum-time and minimum-energy trajectories for a chaser spacecraft to rendezvous with a tumbling space object. Using a direct collocation method based on Gauss pseudospectral approach, trajectories were optimized while also enforcing collision avoidance. However, the computational time was significantly long and likely unsuitable for onboard spacecraft computing. Stoneman and Lampariello [58] utilized nonlinear optimization techniques and a look-up table framework to compute six-degree-of-freedom (6-DOF) optimal docking trajectories that guide the chaser to reach a synchronized mating point with a tumbling object. Trajectories were successfully computed for a variety of tumble types, and also enforced collision avoidance with respect to appendages on the target, such as antennae or solar panels. While the use of a look-up table algorithm produced reasonable computational times (roughly 10 seconds), re-planning the trajectory onboard an actual spacecraft as the trajectory progress was still infeasible. Jewison and Miller [23] directly addressed uncertainty in trajectories by probabilistically optimizing performance while balancing safety. Uncertainties in obstacle positions and the target spacecraft attitude were successfully dealt with via this method. Recently, Malyuta et al. [36] developed a 6-DOF trajectory optimization method for performing docking maneuvers. This method utilized successive convexification to simplify the nonlinear, non-convex problem and enable reasonable run-times (about 5 seconds). This work was also significant in its ability to address discrete control input variables and a number of realistic, challenging constraints.

Tracking Control

Once a trajectory has been computed, is then followed (“tracked”) by a controller operating in a closed-loop fashion (i.e., repeatedly computing new control inputs based on new state information numerous times per second). By operating in a closed-loop fashion, controllers are often able to deal with small levels of aleatoric or epistemic uncertainty, and can be usually designed with a simple dynamic model (often linear) for fast computation times.

There have been a number of research efforts to develop autonomous tracking control for proximity operations. Lee and Pernicka [32] developed optimal control methods for a simulated Space Shuttle rendezvous with the ISS. The state-dependent

Riccati equation and linear quadratic tracking controllers were successfully tested in the 6-DOF simulation. Jiang et al. [25] utilized adaptive control to provide actuator fault-tolerance in a 3-DOF (no attitude dynamics) docking scenario. Lampariello et al. [31] developed a visual servo controller for approaching and grasping a tumbling space object. The method's close link with the computer-vision based navigation system provided effective control in the extremely precise required maneuvering at the moment of capturing the object. This method was successfully validated on ground hardware that simulated the real spacecraft dynamics.

As an aside, note that many controller frameworks avoid addressing the full 6-DOF docking scenario (translational *and* attitude dynamics). This is largely due to the nonlinear attitude dynamics, which can significantly complicate controller design. As such, many standard tracking control frameworks for proximity operations often assume there is an existing attitude controller that can stabilize the spin of the chaser and achieve the correct attitude as it performs the translational maneuver. However, this can raise issues: on a real spacecraft (such as the Apollo CSM), the thrust and torque are naturally coupled due to the arrangement of individual thrusters around the spacecraft.

1.3.3 Model Predictive Control

Recent years have seen a significant interest in applying model predictive control (MPC) methods for autonomous proximity operations. MPC [28] solves a finite-horizon optimization problem in order to minimize a cost function and satisfy the constraints of the scenario. Using a model of the system, the controller can "predict" future state values throughout the horizon that are used in the optimization problem. The problem is re-solved at each time-step, which enables it to operate in closed-loop fashion as long as the prediction horizon is not too long and the optimization problem is relatively simple. This section is split into two parts. A selection of nominal MPC methods are covered first, which are not inherently robust to uncertainty (although nominal MPC is robust in a practical sense for small uncertainties). Then, a selection of robust MPC methods are covered, which seek to *guarantee* performance and constraint satisfaction despite the uncertainty present in the problem.

Nominal MPC methods

Park et al. [44] utilized MPC to perform rendezvous and docking of spacecraft as well as maneuvers to avoid debris objects. Constraints included a line-of-sight cone (to keep the docking port in the field of view of the chaser's sensors), an upper limit on the approach velocity, and thrust limits. This work was one of the earlier efforts to apply MPC to the spacecraft docking problem with its unique constraints. The successful performance in simulation demonstrated the potential of MPC to be used as a suitable controller for scenarios where there are complex constraints and performance must be optimized. MPC was used in a similar manner by Pong et al. [49] to deal with minor actuation faults during proximity operations while satisfying constraints. The method was successfully tested on the Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES) ground hardware testbed in 3-DOF scenarios. This points to the ability of MPC to sometimes handle minor uncertainties simply via its feedback nature.

Further emphasizing the ability of MPC to effectively handle constraints in a feedback manner, Jewison et al. [22] utilized MPC to avoid ellipsoidal obstacle constraints in a simulated spacecraft rendezvous scenario. Modeling the obstacles as ellipsoids creates non-convex constraints, which can pose challenges in running MPC in real-time. However, the sequential quadratic programming method successfully solved the relevant optimization problem in reasonable run-times for a slow-moving rendezvous maneuver.

Finally, Park et al. [45] implemented nonlinear MPC for docking with a rotating target and successfully implemented it on a ground hardware testbed. Once again, the nonlinear MPC in this scenario had to address non-convex collision avoidance constraints. The ability to solve nonlinear MPC problems in under 50 ms (as was observed in the ground hardware experiments) showcases the trend of increased computational power that makes complex controller designs more practically realizable on hardware.

Robust MPC methods

Nominal MPC does not explicitly guarantee satisfactory performance and constraint satisfaction in the presence of uncertainty, despite its feedback nature. Considering the uncertainty associated with autonomous proximity operations and the high consequences of violating constraints (such as collisions), robust MPC formulations are advantageous. One of the most common ways to provide robust MPC is through so-called "tube-based" methods [40]. Details on tube-based MPC theory are given in Chapter 2.

Mirshams and Khosrojerdi [41] implemented tube-based MPC for attitude control despite an underactuated system due to thrust not being available in one axis. The robust MPC framework was successfully able to de-tumble a simulated satellite despite uncertainty in its moment of inertia and low effectiveness in certain thrust axes. Mammarella et al. [37] also utilized tube-based MPC to provide robust performance in docking maneuvers despite uncertainty arising from errors in thruster specifications. This algorithm was successfully tested on a ground hardware testbed and had better performance than a nominal MPC controller. This demonstration also showed that tube-based MPC is practically effective in dealing with general aleatoric uncertainty sources associated with hardware testing. Galivan et al. [20] also addressed robust control for docking maneuvers, with simulated uncertainty arising from thruster noise, errors in the chaser's attitude, eccentric orbits, and rotating target objects. However, robustness was provided via chance constraints, which essentially guarantee satisfaction with a defined probability. Although probabilistic guarantees are less desirable than absolute guarantees in the context of spacecraft operations, this idea allows for the robust controller to be much more flexible in its characterization of the uncertainty.

Robust MPC formulations can also be utilized to track reference trajectories that have already been determined. As an example, Buckner and Lampariello [12] developed a tube-based MPC framework to track trajectories produced by the motion planner detailed in [58] in order to intercept a tumbling target. In this case, the uncertainty arises from estimation errors in the target's rotational state and inertial properties that cause it to tumble in a different manner than predicted. The robust control application of this thesis (Chapter 3) focuses on the same tracking problem, and seeks to make the standard tube-based MPC framework more flexible and practical for proximity operations while improving performance.

1.3.4 Reinforcement Learning

In recent years, there has been an explosion of machine learning applications for a variety of robotics tasks, especially with regards to computer vision tasks and relative navigation. This trend is largely driven by increased computational performance and memory, which have steadily made deep learning techniques more practical and effective. Learning-based techniques are also gaining in the ground of autonomous control, often in the context of reinforcement learning (RL) [60]. RL involves an autonomous agent learning a policy connecting states to actions (i.e., control inputs) in order to maximize a reward signal. Since the learning environment is essentially a black box, the resulting policy is ideally general, model-free, and robust to various sources of uncertainty. Moreover, by using rich deep neural networks to model the policy, several of the issues in standard control methods are avoided, such as challenges in dealing with nonlinear dynamics.

However, the extension of RL to spacecraft guidance, navigation, and control is still largely unexplored. Recent efforts include the application of the "reinforce" algorithm for asteroid mapping (Chan and Agha-mohammadi [13]) and the use of an RL actor-critic framework to widen the capabilities of the zero-effort-miss/zero-effort-velocity guidance algorithm, generalizing it for path constraints in near-rectilinear orbits (Scorsoglio et al. [54]). Broida and Linares [11] used proximal policy optimization (PPO) to accomplish 3-DOF rendezvous trajectories that exploit relative orbital dynamics. Likewise, Gaudet et al. [19, 18] used PPO for producing 6-DOF planetary landings and asteroid hovering maneuvers. The latter work utilized meta-learning, where the RL agent can adapt to a novel environment from learning on a wide range of possible environments. Implementing meta-learning via recurrent neural networks, the trained agent was able to adapt to unique asteroid dynamic environments and actuator faults. Hovell and Ulrich [21] recently presented a guidance policy for 3-DOF proximity operations using the "distributed distributional deep deterministic policy gradient" algorithm, testing it successfully in granite surface hardware experiments. The hardware implementation distinguishes this work from most RL research efforts that are limited to simulation.

However, several aspects of RL must be validated more strongly in order to render the method safely usable for autonomous proximity operations. The lack of guarantees on constraint satisfaction and stability, despite exceptional performance in practice, is undesirable for spacecraft operations. Thus, it is expected that future RL research involving autonomous proximity operations will focus on safety and constraint satisfaction guarantees, along with merging its benefits with standard control methods to develop highly capable hybrid methods.

1.4 Thesis Objectives and Outline

Having covered previous research in the areas of robust control and reinforcement learning for autonomous proximity operations, this section outlines the main purpose of this thesis and its research contributions. This thesis is focused on extending specific methods of robust control and reinforcement learning to make autonomous proximity operations more general and flexible for uncertain scenarios and improve performance in terms of the specified cost function. In doing so, this work incrementally improves the existing pool of guidance and control technologies for autonomous proximity operations, which in turn brings fully-fledged autonomous servicing and active debris removal spacecraft one step closer to reality.

In terms of robust control, this work seeks to extend standard tube-based MPC to develop an algorithm in what is termed as "online" tube-based MPC. This algorithm allows the robust "tube" to shrink and grow accordingly with measured uncertainty data as the trajectory progresses. In this way, online, tube-based MPC both a) removes the need for precise prior knowledge about the uncertainty characteristics, and b) improves performance for cases when standard tube-based MPC is over-conservative. Having developed the online, tube-based MPC algorithm, it is then applied to an autonomous proximity operations scenario involving a tumbling space object with uncertain inertial properties. The controller is able to successfully track intercept trajectories despite incorrect prior knowledge on the target's inertia tensor. Throughout numerous simulation runs, online tube-based MPC outperforms standard tube-based MPC.

With regards to reinforcement learning, this work utilizes PPO to develop a 6-DOF docking policy that is implemented as a feedback control law. The policy is robust to an range of uncertain initial conditions, successfully docking with a rotating target while preventing collisions and minimizing state error and control input costs. Once learned, the policy commands control inputs at a very fast rate, and thus is suitable for real-time implementation. To the best of the author's knowledge, this is the first application of RL for spacecraft docking involving a full 6-DOF dynamic environment. Results also include a comparison with standard trajectory optimization techniques in order to give discussion and general insight on the benefits and disadvantages of using RL for spacecraft proximity operations.

In summary, the contributions of this thesis are as follows:

1. Development of the "online" tube-based MPC algorithm, which enhances the flexibility and performance of tube-based MPC.

2. Application and validation of the online tube-based MPC algorithm in a simulated proximity operations scenario involving the intercept of a tumbling target with uncertain inertial properties.

3. Development and validation of a RL-based policy that is able to successfully dock with a rotating target in a simulated 6-DOF dynamic environment.

The thesis is structured as follows: Chapter 2 consists of the theoretical development of the online tube-based MPC algorithm. Chapter 3 outlines the tumbling target intercept problem and shares the results obtained by applying the online tube-based

MPC algorithm. Chapter 4 formulates the RL-based 6-DOF policy for docking with rotating targets and shares results obtained by applying the policy to the simulated Apollo transposition and docking maneuver. Finally, Chapter 5 shares conclusions and discussions regarding future work. Appendices cover a 2-D double integrator system for controller demonstrations, details on polytopic and ellipsoidal sets, and a formulation of the optimal control problems used in evaluating the RL-based policy's docking performance.

Chapter 2

Online, Tube-based Model Predictive Control

This thesis proposes online, tube-based MPC as the method of choice for robust control in autonomous proximity operations. This chapter focuses on the theoretical formulation of this control algorithm. A background on nominal (i.e., non-robust) MPC and standard tube-based MPC is covered first, before introducing the proposed, online, tube-based MPC algorithm. Throughout this chapter, a 2-D, double-integrator system is utilized to explain and demonstrate the properties of each control method. Details on the 2-D, double-integrator test cases are given in Appendix A.

2.1 Model Predictive Control

MPC is a commonly used strategy for constrained control systems [28]. An important advantage of MPC when compared to less advanced control methods, e.g., a linear quadratic regulator (LQR), is that it computes an online solution to a numerical optimization problem that allows for constraint satisfaction. This problem is solved for a finite horizon (termed the *prediction horizon*), where predicted states are obtained via the dynamic model of the system. Using a finite horizon permits fast solving times; this allows the controller to rapidly accept new state information and provide closed-loop control while enforcing state and control input constraints.

Optimization is implemented in the MPC framework using a cost function that balances tracking performance and controller efficiency. Thus, at each time-step, the constrained optimization problem is solved over the prediction horizon to determine the optimal sequence of control inputs $\mathbf{u}(0, 1, ..., N-1)$ where N is the number of time-steps in the prediction horizon. The first control input $\mathbf{u}_0$ is then applied to the system. Then, at the next time-step, the constrained optimization problem is re-solved based on the new current state, yielding an updated solution. By re-solving the optimization problem at each time-step based on the new state, MPC exhibits feedback control according to a so-called "receding horizon" control law.

Model predictive control utilizes the following form for the linear, time-invariant system dynamics:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state and $\mathbf{u} \in \mathbb{R}^m$ is the control input. The terms n and m refer to the state and control input dimensions, respectively. These dynamics are formulated in discrete time with a sample period T_s. The matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times m}$ are the state transition and input distribution matrices, respectively. To ensure that the optimized control sequence can be calculated efficiently, MPC cost functions are typically convex while the constraints are typically linear. In this work, the MPC optimization problem is formulated as a quadratic program (QP):

$$\min_{\mathbf{u},\ \mathbf{x}} \quad \sum_{i=k}^{k+N-1} \left(\left|\left|\mathbf{x}(i) - \bar{\mathbf{x}}(i)\right|\right|_{\mathbf{Q}}^2 + \left|\left|\mathbf{u}(i)\right|\right|_{\mathbf{R}}^2 \right) + \left|\left|\mathbf{x}(k+N) - \bar{\mathbf{x}}(k+N)\right|\right|_{\mathbf{P}}^2 \tag{2.2a}$$

$$\text{s.t.} \quad \mathbf{x}(i+1) = \mathbf{A}\mathbf{x}(i) + \mathbf{B}\mathbf{u}(i) \quad \forall i = k, ..., k+N-1 \tag{2.2b}$$

$$\mathbf{x}(i) \in \mathcal{X} \quad \forall i = k+1, ..., k+N \tag{2.2c}$$

$$\mathbf{u}(i) \in \mathcal{U} \quad \forall i = k, ..., k+N-1 \tag{2.2d}$$

where $\bar{\mathbf{x}}$ is the desired state, $\mathbf{Q} \in \mathbb{R}^{n\times n}, \mathbf{Q} \succeq \mathbf{0}$ is the cost matrix applied to tracking error, $\mathbf{R} \in \mathbb{R}^{m\times m}, \mathbf{R} \succ \mathbf{0}$ is the cost matrix applied to control inputs, and $\mathbf{P} \in \mathbb{R}^{n\times n}, \mathbf{P} \succ \mathbf{0}$ is the "terminal" tracking cost matrix applied to the final state of the prediction horizon. The cost notation $\left|\left|\mathbf{u}(i)\right|\right|_{\mathbf{R}}^2$ is equivalent to $\mathbf{u}(i)^\top \mathbf{R}\mathbf{u}(i)$. The

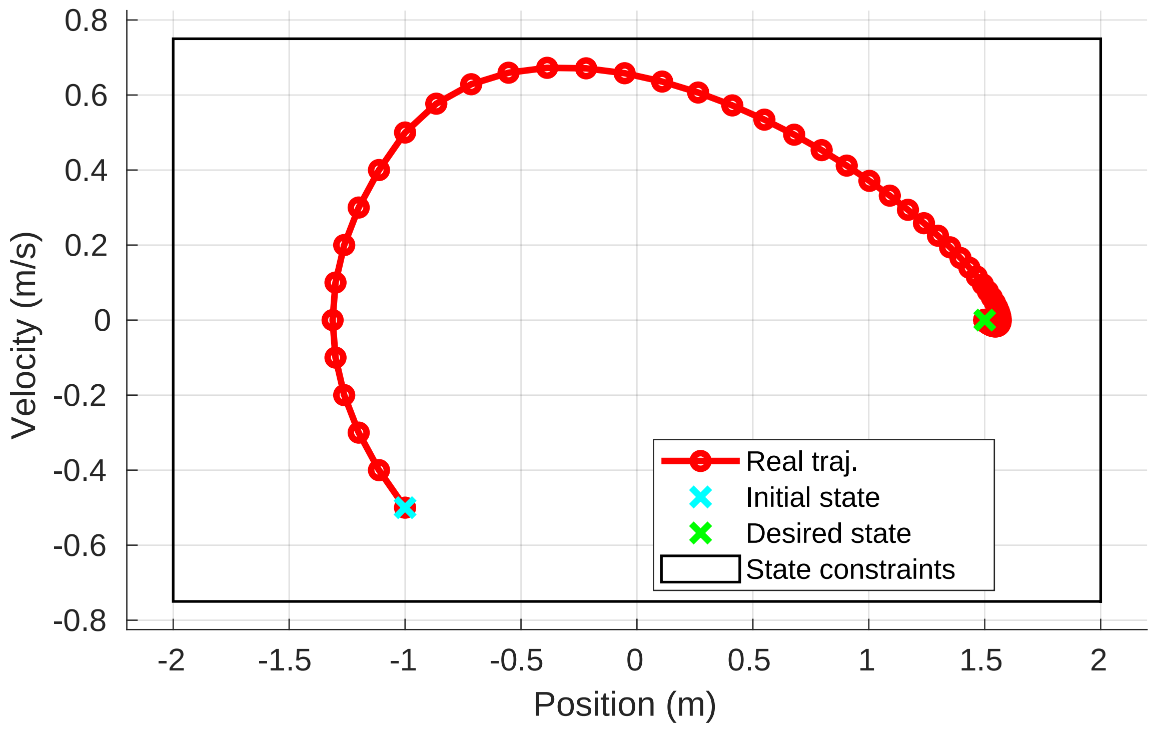


Figure 2-1: A trajectory produced via a model-predictive controller for the 2-D double-integrator system with no uncertainty. The red indicates the trajectory while the large box indicates the state constraints. Notice how the predictive capabilities of MPC enable constraint satisfaction in a feedback manner.

state and control input constraints are represented by $\mathcal{X}$ and $\mathcal{U}$ respectively. For the QP formulation, these constraints are polytopes and can be re-written as

$$\mathbf{x} \in \mathcal{X} := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}_{\mathcal{X}}\mathbf{x} \leq \mathbf{b}_{\mathcal{X}}\} \tag{2.3a}$$

$$\mathbf{u} \in \mathcal{U} := \{\mathbf{u} \in \mathbb{R}^m : \mathbf{A}_{\mathcal{U}}\mathbf{u} \leq \mathbf{b}_{\mathcal{U}}\} \tag{2.3b}$$

where $\mathbf{A}_{\mathcal{X}}$, $\mathbf{A}_{\mathcal{U}}$ are the "A" polytope matrices and $\mathbf{b}_{\mathcal{X}}$, $\mathbf{b}_{\mathcal{U}}$ are the "b" polytope vectors from (B.1). More information on polytope sets can be found in Appendix B.

The above QP formulation for MPC is applied to the 2-dimensional system defined in Appendix A. Figure 2-1 depicts a resulting trajectory produced by the controller. The prediction horizon was $N = 5$, while the cost matrices were $\mathbf{Q} = \mathbf{I}_2$ and $\mathbf{R} = 10$. The controller successfully reaches the goal state in an optimized manner while abiding by the position/velocity constraints. This is due to the constrained optimization problem in (2.2) being solved at each time-step using new state information.

However, if the system is uncertain, there is no guarantee for MPC that the unmodeled disturbances will not cause the controller to violate the constraints or become unstable. While the feedback nature of MPC can handle uncertainty in the unconstrained case, for controllers operating in critical scenarios (such as an autonomous spacecraft performing a docking maneuver), it is desirable to have guarantees in addressing the uncertainty. This points to the need of *robust* control methods, which, despite the uncertainty, can still guarantee constraint satisfaction and stability.

2.2 Standard Tube-based Model Predictive Control

Tube-based MPC is an effective way to guarantee stability and the satisfaction of constraints in the presence of uncertainty while still retaining nominal MPC optimization capabilities [40]. Standard tube-based MPC applies a feedback gain to counteract the uncertainty in the real system, which results in a robust positively invariant (RPI) set, i.e., the "tube". Positive invariance means that the state will stay within a bounded set as time progresses; the robust term denotes that positive invariance is achieved despite the presence of uncertainty. As such, the tube is guaranteed to contain any possible evolving trajectories that are disturbed by the unmodeled terms. The tube is also used to shrink the original constraints of the problem, which are then implemented in a standard MPC optimization problem to determine the nominal states and control inputs throughout the prediction horizon.

Standard tube-based MPC uses the following notation to address both the real and nominal dynamics of the system. The real system that includes the uncertainty is

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(k) \tag{2.4}$$

where $\mathbf{w} \in \mathbb{R}^n$ is an unknown but bounded term referred to as the *exogenous input* to the system. This term encapsulates the uncertainty of the real system. The *nominal dynamics* do not include the exogenous input:

$$\mathbf{z}(k+1) = \mathbf{A}\mathbf{z}(k) + \mathbf{B}\mathbf{v}(k) \tag{2.5}$$

where $\mathbf{z} \in \mathbb{R}^n$ is the nominal state and $\mathbf{v} \in \mathbb{R}^m$ is the nominal control input. Referring to the term $\mathbf{w}$ as "bounded" means it resides in the closed set $\mathcal{W}$, which contains all possible finite values of $\mathbf{w}$:

$$\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^n \tag{2.6a}$$

$$\max_{\mathbf{w} \in \mathcal{W}} |\mathbf{w}| < \infty \ . \tag{2.6b}$$

While the true set $\mathcal{W}$ typically does not have well-defined geometry, it is often approximated via polytopes or ellipsoids (Appendix B).

2.2.1 Exogenous Input Rejection

The tube-based MPC law for each time-step is written as

$$\mathbf{u}(k) = \boldsymbol{\nu}(k) + \mathbf{K}_r\big(\mathbf{x}(k) - \mathbf{z}(k)\big) \tag{2.7}$$

where $\mathbf{K}_r \in \mathbb{R}^{m \times n}$ is the exogenous input rejection feedback gain and $\boldsymbol{\nu}(k)$ and $\mathbf{z}(k)$ are the nominal initial control input and state, respectively, determined through MPC optimization. Essentially, the feedback gain $\mathbf{K}_r$ counteracts the difference between the real and nominal states of the trajectory. Certain gain values of $\mathbf{K}_r$ can result in an RPI set centered on the nominal state $\mathbf{z}(k)$ (see Section 2.2.2). This is critical for obtaining stability and constraint satisfaction guarantees despite the uncertainty in the system.

There are a number of methods to determine suitable gain values for $\mathbf{K}_r$. Perhaps the simplest is to calculate a stablizing gain matrix $\mathbf{K}_r$ via linear-quadratic-regulator (LQR) design, where

$$\rho(\mathbf{A} + \mathbf{B}\mathbf{K}_r) < 1 \tag{2.8}$$

with ρ signifying the spectral radius.

With tuning, this can achieve suitable results; however, a) it does not always guarantee the existence of an RPI set, and b), does not explicitly seek to minimize the size of the resulting set. Since a smaller RPI set reduces the difference between the real and nominal states and limits the degree of constraint shrinking (see section 2.2.3), it is generally favorable to seek gain values for $\mathbf{K}_r$ that minimize the resulting RPI set. However, more aggressive gain values for $\mathbf{K}_r$ that reduce the size of the RPI set also result in a greater degree of constraint shrinking on the control inputs. Practically speaking, more of the limited control authority is transferred to the exogenous input rejection part of the problem, and less is available for the MPC optimization step. Thus, care must be taken to choose a gain for $\mathbf{K}_r$ that does not result in an empty shrunk constraint for the control inputs.

2.2.2 Minimum Robust Positively Invariant Set

The minimum RPI (mRPI) set $\mathcal{Z}^*$ is the smallest set that contains all trajectories that result from applying the feedback gain $\mathbf{K}_r$ to the difference between the real and nominal states:

$$(\mathbf{A} + \mathbf{B}\mathbf{K}_r)\mathbf{x} + \mathbf{w} \in \mathcal{Z}^*, \quad \forall\, \mathbf{x} \in \mathcal{Z}^*, \mathbf{w} \in \mathcal{W} \ . \tag{2.9}$$

In other words, the set $\mathcal{Z}^*$ contains all future states $\mathbf{x}$ despite the exogenous inputs $\mathbf{w}$ as long as the initial state $\mathbf{x}(0)$ is also contained in $\mathcal{Z}^*$. Note that the true minimal set $\mathcal{Z}^*$ is often difficult to determine; thus, an approximate set $\mathcal{Z}$ is usually found instead and used as the tube set in tube-based MPC. Also note that the tube set $\mathcal{Z}$ is directly influenced by the definition of the uncertainty bounds, $\mathcal{W}$. Thus, it is important to have accurate knowledge of the bounds of $\mathbf{w}$ in order to properly define $\mathcal{W}$ and compute $\mathcal{Z}$.

Given the uncertainty set $\mathcal{W}$ and the dynamic matrices $\mathbf{A}$ and $\mathbf{B}$, there are several algorithms to calculate an exogenous input rejection gain $\mathbf{K}_r$ that results in the tube set $\mathcal{Z}$. These algorithms differ depending on the parameterization of the uncertainty set $\mathcal{W}$ and the desired parameterization of the tube $\mathcal{Z}$. Limón et al. [33] present an effective strategy to find the gain matrix $\mathbf{K}_r$ that minimizes the tube $\mathcal{Z}$ if $\mathcal{W}$ and $\mathcal{Z}$ are assumed to be parameterized as polytopes. Based on linear matrix inequalities (LMIs) [8] and convex optimization, this method optimizes the gain matrix $\mathbf{K}_r$ while ensuring the shrunk control input constraint is not empty. However, the actual resulting definition of the tube $\mathcal{Z}$ must still be approximated separately. As such, Raković et al. [50] present a straightforward algorithm based on the gain matrix $\mathbf{K}_r$ to provide a close, polytope approximation to $\mathcal{Z}^*$ while still guaranteeing RPI properties.

For reasons detailed in Section 2.3, the chosen parameterizations of $\mathcal{W}$ and $\mathcal{Z}$ are ellipsoids for this thesis. The method developed by Polyak et al. [48] is utilized to find the combination of the gain matrix $\mathbf{K}_r$ and the tube $\mathcal{Z}$ that results in the minimal ellipsoidal approximation to $\mathcal{Z}^*$. Also based on LMIs, this convex optimization problem is formulated as

$$\min_{\mathbf{P}_{\mathcal{Z}},\, \mathbf{Y},\, \mathbf{M}} \quad \operatorname{trace}\big(\mathbf{P}_{\mathcal{Z}}\big) \tag{2.10a}$$

$$\text{s.t.} \quad \frac{1}{\alpha}\Big(\mathbf{A}\mathbf{P}_{\mathcal{Z}}\mathbf{A}^\top + \mathbf{B}\mathbf{Y}\mathbf{A}^\top + \mathbf{A}\mathbf{Y}^\top\mathbf{B}^\top + \mathbf{B}\mathbf{M}\mathbf{B}^\top\Big) - \mathbf{P}_{\mathcal{Z}} + \frac{1}{1-\alpha}\mathbf{P}_{\mathcal{W}} = \mathbf{0} \tag{2.10b}$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{Y} \\ \mathbf{Y}^\top & \mathbf{P}_{\mathcal{Z}} \end{bmatrix} \geq \mathbf{0} \tag{2.10c}$$

$$\begin{bmatrix} \mathbf{P}_{\mathcal{Z}} & \mathbf{Y}^\top \\ \mathbf{Y} & (\beta u_p)^2\mathbf{I}_m \end{bmatrix} \geq \mathbf{0} \tag{2.10d}$$

where $\mathbf{P}_{\mathcal{Z}}$ is the resulting ellipsoidal shape matrix of the tube set $\mathcal{Z}$, $\mathbf{P}_{\mathcal{W}}$ is the ellipsoidal shape matrix of the exogenous input set $\mathcal{W}$, and $\mathbf{M} \in \mathbb{R}^{m\times m}$ and $\mathbf{Y} \in \mathbb{R}^{n\times m}$ are additional optimization variables. The terms $\alpha \in (0,1)$ and $\beta \in (0,1)$ are design parameters, with β being applied to allocate a certain percentage of the defined control limit magnitude u_p in minimizing $\mathcal{Z}$. Note that using the trace of the ellipsoidal shape matrix retains convexity of the optimization problem. As such, this problem is efficiently solved using semi-definite programming (SDP) techniques.

The corresponding exogenous input rejection gain $\mathbf{K}_r$ is then simply calculated as

$$\mathbf{K}_r = \mathbf{Y}^{-1}\mathbf{P}_{\mathcal{Z}}. \tag{2.11}$$

Figure 2-2 depicts an ellipsoidal tube $\mathcal{Z}$ that approximates the mRPI for a given system. Note that, since the initial state started within the tube, all future states remain in the tube despite being perturbed by bounded values of the exogenous input

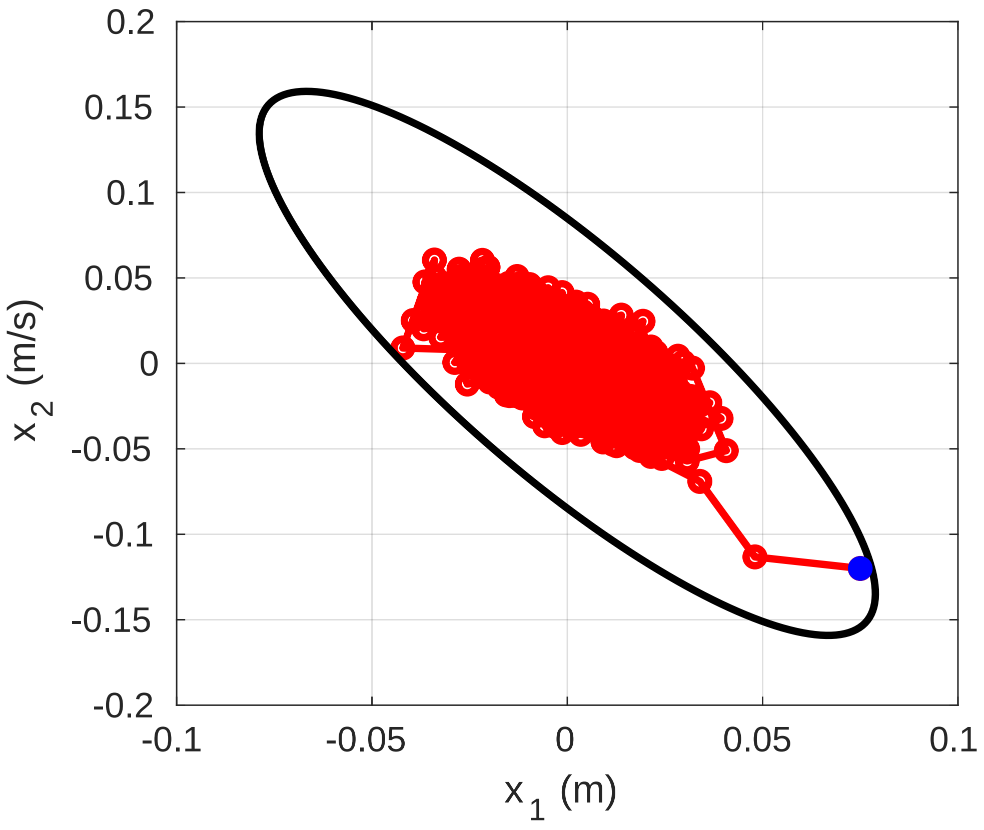


Figure 2-2: An example ellipsoidal tube that approximates the minimum robust positively invariant set for a double-integrator system. All future states (red) stay within the tube despite the perturbations caused by the exogenous input **w** since the initial state (blue) was within the tube.

w. This concept serves as the source of robustness for tube-based MPC.

2.2.3 Constraint Shrinking

After the tube set $\mathcal{Z}$ has been calculated, the original constraints of the problem can be shrunk in order to retain a constraint satisfaction guarantee despite the uncertainty. These shrunk constraints are utilized in the subsequent nominal MPC optimization problem, which does not consider the uncertainty.

The shrunk constraints are simply obtained via the Pontryagin set difference:

$$\bar{\mathcal{X}} = \mathcal{X} \ominus \mathcal{Z} \tag{2.12a}$$

$$\bar{\mathcal{U}} = \mathcal{U} \ominus \mathbf{K}_r \mathcal{Z} \tag{2.12b}$$

where $\ominus$ indicates the Pontryagin difference, and $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$ indicate the shrunk state and control input constraints, respectively. As mentioned before, it is clear that more aggressive selections of the exogenous input rejection gain $\mathbf{K}_r$ result in more shrinking of the control input constraint. Thus, there is a trade-off between robustness (more aggressive gain $\mathbf{K}_r$) and optimal controller performance (more control input authority allocated for the nominal MPC problem).

If the original constraints $\mathcal{X}$ and $\mathcal{U}$ are modeled via polytopes, the resulting shrunk constraints $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$ are also polytopes sets if the tube $\mathcal{Z}$ is either a polytope or ellipsoid. Since the parameterization of the tube $\mathcal{Z}$ in this work is chosen to be an ellipsoid, the Pontryagin set difference between a polytope and ellipsoid is explicitly stated for the state constraint:

$$\mathbf{A}_{\bar{\mathcal{X}}} = \mathbf{A}_{\mathcal{X}} \tag{2.13a}$$

$$b_{\bar{\mathcal{X}}_i} = b_{\mathcal{X}_i} - \sqrt{\mathbf{P}_{\mathcal{Z}}\mathbf{a}_{\mathcal{X}_i}^\top \cdot \mathbf{a}_{\mathcal{X}_i}^\top} \tag{2.13b}$$

where $b_{\mathcal{X}_i}$ is the i-th element of the vector $\mathbf{b}_{\mathcal{X}}$ and $\mathbf{a}_{\mathcal{X}_i}$ is the i-th row of the matrix $\mathbf{A}_{\mathcal{X}}$. Figure 2-3 depicts the original and shrunk constraints the example 2-D problem.

2.2.4 Nominal Model Predictive Control Problem

Having defined the tube $\mathcal{Z}$ and the shrunk constraints $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$, it is now possible to formulate an MPC optimization problem that guarantees robust constraint satisfaction despite only using the nominal system dynamics without uncertainty. The result of the nominal MPC optimization problem is the nominal control input $\boldsymbol{\nu}$ as well as the nominal initial state $\mathbf{z}$. Together, these terms are used in (2.7) to determine the final robust control input that is applied to the real system.

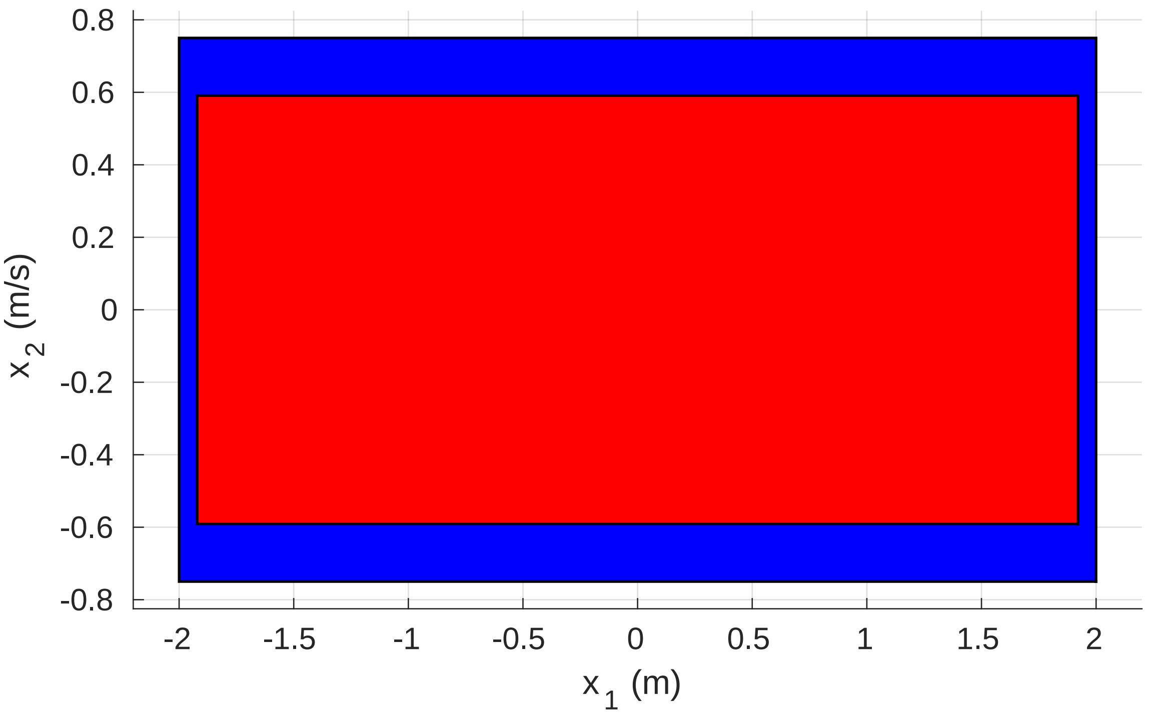


Figure 2-3: An example of state constraint shrinking for tube-based MPC on the 2-D system. The outer, blue-colored box (polytope) is the original state constraint, while the inner, red-colored box is the shrunk constraint that ensures constraint satisfaction despite the exogenous inputs.

The nominal MPC problem for standard, tube-based MPC is formulated as

$$\min_{\boldsymbol{\nu},\ \mathbf{z}} \quad \sum_{i=k}^{k+N-1} \left(\left|\left|\mathbf{z}(i) - \bar{\mathbf{x}}(i)\right|\right|_{\mathbf{Q}}^2 + \left|\left|\mathbf{v}(i)\right|\right|_{\mathbf{R}}^2 \right) + \left|\left|\mathbf{z}(k+N) - \bar{\mathbf{x}}(k+N)\right|\right|_{\mathbf{P}}^2 \tag{2.14a}$$

$$\text{s.t.} \quad \mathbf{z}(i+1) = \mathbf{A}\mathbf{z}(i) + \mathbf{B}\mathbf{v}(i) \ \ \forall i = k, ..., k+N-1 \tag{2.14b}$$

$$\mathbf{z}(i) \in \bar{\mathcal{X}} \ \ \forall i = k, ..., k+N \tag{2.14c}$$

$$\mathbf{v}(i) \in \bar{\mathcal{U}} \ \ \forall i = k, ..., k+N-1 \tag{2.14d}$$

$$\mathbf{z}(k) \in \mathbf{x}(k) \oplus -\mathcal{Z} \tag{2.14e}$$

where $\oplus$ indicates the Minkowski sum.

There are several important differences between the above problem and (2.2). For one, the initial state of the trajectory throughout the (nominal) prediction horizon is now a decision variable $\mathbf{z}(k)$. The tube $\mathcal{Z}$ is centered at this nominal initial state; thus (2.14e) requires that the real initial state $\mathbf{x}(k)$ is encapsulated within the tube. By requiring $\mathbf{x}(k)$ to be inside the tube centered at $\mathbf{z}(k)$, the robust positively invariant properties shown in Figure 2-2 are achieved and all future states $\mathbf{x}$ will continue to remain in the tube, thus guaranteeing stability. Furthermore, since the tube is contained in the original constraints by restricting the nominal state $\mathbf{z}$ to be within the shrunk constraints, it also guarantees robust constraint satisfaction despite the uncertainty.

Practically speaking, choosing an ellipsoidal form of $\mathcal{Z}$ creates a quadratic constraint for (2.14e)

$$\big(\mathbf{x}(k) - \mathbf{z}(k)\big)^\top \mathbf{P}_{\mathcal{Z}} \big(\mathbf{x}(k) - \mathbf{z}(k)\big) \leq 1 \ . \tag{2.15}$$

This makes the MPC problem a quadratically constrained quadratic program (QCQP), which is slightly more complex than the QP formulation in (2.2). However, the problem is still convex and can be easily handled via second-order cone programming (SOCP) methods.

2.2.5 Application to the 2-D Double-Integrator System

The above formulation of standard, tube-based MPC was applied to the 2-D double-integrator system outlined in Appendix B. Figure 2-4 depicts a resulting trajectory produced by the controller for a case when the exogenous input $\mathbf{w}$ was sampled using the standard deviation values of $\sigma_r = 7.5 \times 10^{-3}$, $\sigma_v = 7.5 \times 10^{-3}$, and a mean value of $\boldsymbol{\mu}_{\mathbf{w}} = [0, 0]^\top$. The prediction horizon was $N = 5$, while the cost matrices were set as $\mathbf{Q} = \mathbf{I}_2$ and $\mathbf{R} = 10$. The $\mathbf{P}$ matrix was obtained via LQR design using $\mathbf{Q}$ and $\mathbf{R}$. The exogenous input rejection gain $\mathbf{K}_r$ and tube $\mathcal{Z}$ were determined via (2.10), yielding a gain of $\mathbf{K}_r = [-7.04, -4.54]$.

Notice that, despite the perturbations caused by the exogenous input $\mathbf{w}$, the real state stays within the tube and thus within the original constraints of the problem. The exogenous input rejection gain $\mathbf{K}_r$ also effectively counteracts $\mathbf{w}$, driving the real state to the desired final state near the end of the trajectory.

However, there are important drawbacks to the standard, tube-based MPC formulation. Since the exogenous input rejection gain and tube set are both calculated

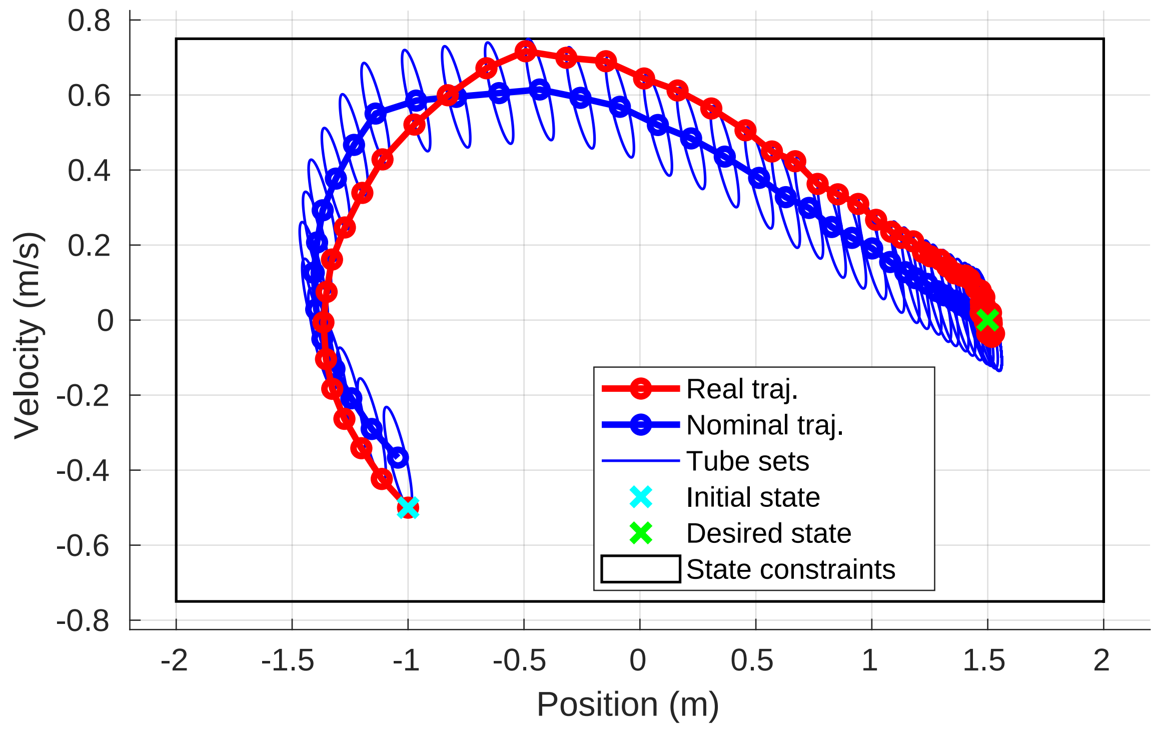


Figure 2-4: An successful trajectory produced by the standard, tube-based MPC for the disturbed 2-D double-integrator system.

offline prior to online implementation, it is critically reliant on an accurate bounding set $\mathcal{W}$ of the exogenous inputs. If the estimated bounding set $\widehat{\mathcal{W}}$ is significantly conservative and the level of uncertainty is actually lower, the standard tube-based MPC misses out on performance gains since it allocates too much control authority to the non-optimized exogenous input rejection gain $\mathbf{K}_r$.

On the other hand, a more catastrophic result occurs when the approximated bounding set of the exogenous inputs $\widehat{\mathcal{W}}$ is *less* conservative than the true set of exogenous inputs. This issue can arise when there is a greater-than-expected variance or a significant bias in the exogenous inputs. This results in the standard tube-MPC calculating a tube set that is not sufficient for the actual level of uncertainty. Thus, the robust guarantees of stability and constraint satisfaction are gone.

Consider the following scenario in the 2-D double-integrator problem where the *real* exogenous inputs are sampled using standard deviations of $\sigma_r = 5 \times 10^{-3}$, $\sigma_v = 5 \times 10^{-3}$ and the mean $\boldsymbol{\mu}_{\mathbf{w}} = [0, 0.04]^\top$ while the *estimated* exogenous input standard deviations are $\hat{\sigma}_r = 5 \times 10^{-3}$, $\hat{\sigma}_v = 5 \times 10^{-3}$, and the mean $\hat{\boldsymbol{\mu}}_{\mathbf{w}} = \mathbf{0}$. As shown in

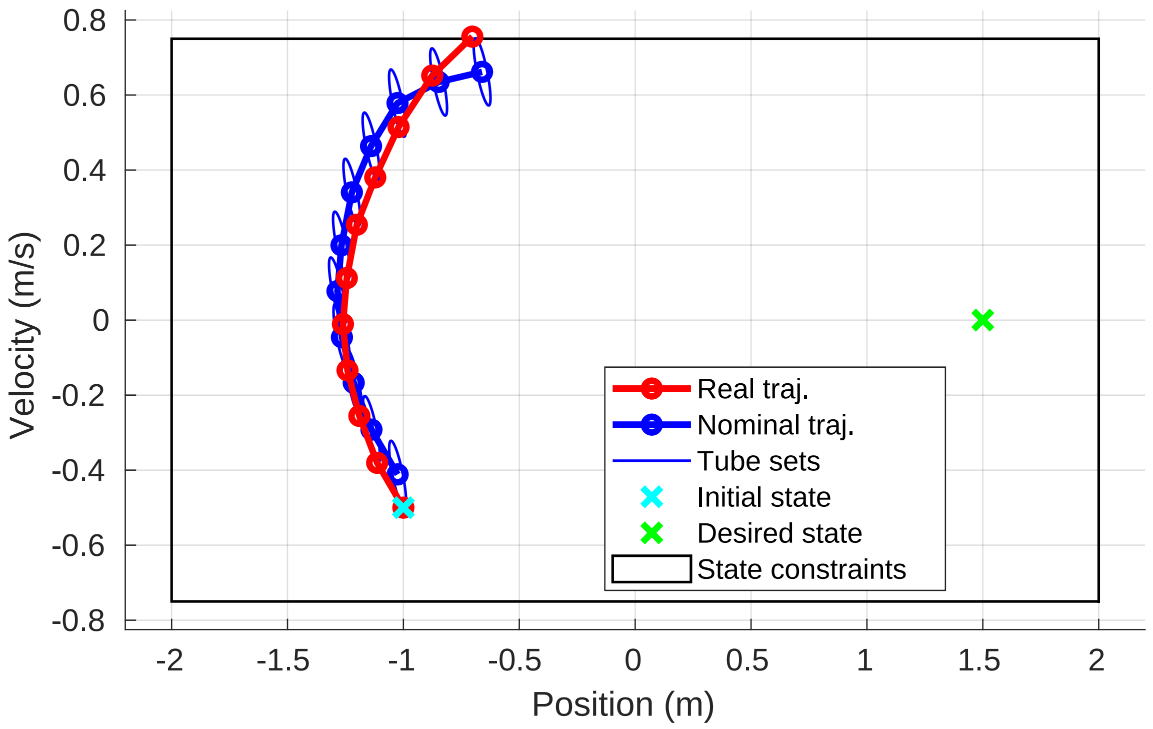


Figure 2-5: A failed trajectory produced by the standard tube-based MPC for the disturbed 2-D double-integrator system. Since the initialized uncertainty bounds did not accommodate a strong bias in the velocity exogenous input component, the velocity constraint was violated.

Figure 2-5, the tube is not accurate and the strong velocity bias in the exogenous inputs forces the real state outside the constraints. Thus, the standard tube-MPC failed the trajectory since it relied completely on pre-determined information about the exogenous input $\mathbf{w}$, and did not attempt to infer its characteristics online.

2.3 Online Tube-based Model Predictive Control

An *online* tube-based MPC algorithm is developed to address the aforementioned issues of standard tube-based MPC. This algorithm is one of the main contributions of this thesis. The idea is to constantly update the approximated tube set throughout the trajectory using measured exogenous input data. This allows for the estimated bounds on the exogenous inputs $\widehat{\mathcal{W}}$ to be formulated via real, measured data instead of using a pre-determined value as in standard tube-based MPC. The current value of the exogenous input $\mathbf{w}$ is also estimated in this process and fed into the nominal

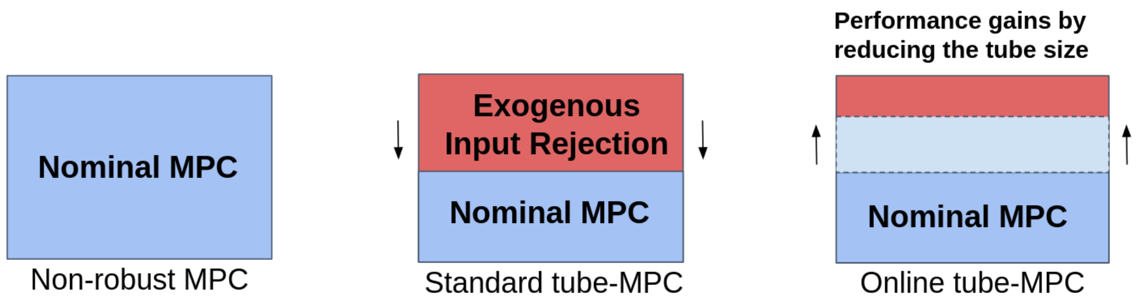


Figure 2-6: An illustration of how standard tube-based MPC strips away part of the control authority for exogenous input rejection, and how online tube-based MPC can re-allocate control authority to the nominal MPC optimization step if the uncertainty is relatively low.

model. Furthermore, if the exogenous inputs are found to be generally non-stochastic (i.e., low amounts of noise), predictions of their future values can be obtained via an identified system model. This allows the dynamic nature of the exogenous inputs to be incorporated into the prediction horizon of the nominal MPC optimization step.

By adjusting the tube online based on measured data, the two previously mentioned issues of standard tube-based MPC can be addressed. The first issue of having an over-conservative estimate of $\mathcal{W}$ and losing out on a potential increase in performance is clearly addressed by online tube-based MPC; the tube will be shrunk over time as the measured exogenous input data indicates a smaller set of exogenous inputs $\mathcal{W}$ when compared to its initial estimate $\widehat{\mathcal{W}}$. As such, a greater portion of the finite control authority will be allocated for the nominal MPC optimization step, which generally increases performance in minimizing the cost function (Figure 2-6). This is especially true in such cases where the exogenous input $\mathbf{w}$ is a fairly large but constant or highly deterministic term. In this case, the tube is shrinks to a small set centered around the estimated value of $\mathbf{w}$. Since the online estimates of $\mathbf{w}$ are incorporated into the nominal model, the nominal MPC handles most of the control input needed to address the exogenous inputs.

The second, more serious, issue of having an initial estimate $\widehat{\mathcal{W}}$ that is too small to encapsulate the true possible exogenous input values $\mathbf{w}$ is also addressed via online tube-based MPC. The controller still lacks robustness early on in the trajectory since there is not yet enough data on the exogenous inputs to converge to a correct estimate of their bounds. Thus, care should be taken if the initial conditions of the trajectory are close to the constraint boundaries. But after a sufficient amount of data has been collected, the calculated tube will grow in size based on the up-to-date estimates of the exogenous input bounds $\widehat{\mathcal{W}}$. Since the estimation of $\widehat{\mathcal{W}}$ is dynamic, online tube-based MPC is eventually able to provide robustness despite a poor initial estimate of the exogenous input bounds. This allows for greater flexibility in applying robust control to problems when the characteristics of the exogenous inputs $\mathbf{w}$ are (initially) largely unknown.

Of course, re-calculating the tube based on updated estimates of $\widehat{\mathcal{W}}$ raises questions on the computational feasibility of the overall algorithm. This is why ellipsoidal sets are chosen for describing the tube set $\mathcal{Z}$ and exogenous input set $\mathcal{W}$. The number of parameters for an ellipsoidal set is fixed for a given state dimension n. This contrasts with polytopes, which scale poorly with increasing values of n and can often be very complex depending on the method used to approximate the tube $\mathcal{Z}$. Furthermore, the optimization problem to approximate an ellipsoidal tube set (2.10) is convex and can be solved efficiently. Finally, ellipsoidal descriptions of the exogenous input set $\mathcal{W}$ naturally correspond to the covariance obtained in estimating the exogenous input values $\mathbf{w}$. As such, for practical implementation, ellipsoidal representations of the tube $\mathcal{Z}$ and exogenous input set $\mathcal{W}$ are the most suitable for online tube-based MPC.

Generally, the online tube-based MPC algorithm follows three major steps for each time-step k:

1. Compute a control input $\mathbf{u}(k)$ via the nominal MPC problem and the exogenous input rejection gain $\mathbf{K}_r(k)$.

2. Estimate the next exogenous input $\mathbf{w}(k+1)$ and its covariance, which in turn yields an updated tube and shrunk constraints.

3. If applicable, update the parameters of the identified dynamic model that predicts future values of the exogenous inputs $\mathbf{w}$ throughout the MPC prediction horizon.

These steps are explained in more detail below.

2.3.1 Control Input Determination

The control input $\mathbf{u}(k)$ is determined in a similar manner to standard tube-based MPC. However, there are a few modifications to the nominal MPC optimization problem. The estimate of the exogenous input at time-step k, denoted as $\widehat{\mathbf{w}}(k)$, is included as an initial condition. This allows for an improved nominal dynamic model for the first step in the prediction horizon k:

$$\mathbf{z}(k+1) = \mathbf{A}\mathbf{z}(k) + \mathbf{B}\mathbf{v}(k) + \widehat{\mathbf{w}}(k) \tag{2.16}$$

This means that the tube only has to encapsulate the error in estimating $\mathbf{w}(k)$: $\widetilde{\mathbf{w}}(k) \triangleq \mathbf{w}(k) - \widehat{\mathbf{w}}(k)$.

Furthermore, if predictions of future exogenous inputs via an identified model are enabled, we can include them as part of the state vector throughout the entire prediction horizon. In this work, an autoregressive moving average (ARMA) model structure is used for prediction. Via the ARMA formulation, the linear dynamic model to predict future exogenous inputs is defined as

$$\mathbf{w}^*(k+1) = \mathbf{A}^*_{\mathbf{w}}(k)\mathbf{w}^*(k) \tag{2.17}$$

where $\mathbf{w}^*(k) \in \mathbb{R}^{n(n_a)}$ is a stack of n_a past measured/predicted values of $\mathbf{w}$ and n_a is the number of autoregressive parameters used in the ARMA model formulation (i.e., the model order). The matrix $\mathbf{A}^*_{\mathbf{w}}(k) \in \mathbb{R}^{n(n_a)\times n(n_a)}$ is the state transition matrix representation of the identified ARMA model. More details on the ARMA system identification method for predicting future exogenous inputs are included in Section 2.3.3.

The extended state vector that is modeled via the nominal dynamics can now be written as $\left[\mathbf{z}(k), \mathbf{w}^*(k)\right]^\top$. The resulting nominal MPC problem for online, tube-based

MPC is now formulated as:

$$\min_{\boldsymbol{\nu},\ \mathbf{z}} \sum_{i=k}^{k+N-1} \Big(\big|\big|\mathbf{z}(i) - \bar{\mathbf{x}}(i)\big|\big|_{\mathbf{Q}}^{2} + \big|\big|\boldsymbol{\nu}(i)\big|\big|_{\mathbf{R}}^{2} \Big) + \big|\big|\mathbf{z}(k+N) - \bar{\mathbf{x}}(k+N)\big|\big|_{\mathbf{P}}^{2} \tag{2.18a}$$

$$\text{s.t.} \quad \begin{bmatrix} \mathbf{z}(i+1) \\ \mathbf{w}^{*}(i+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & [\mathbf{I}_n,\ \mathbf{0}] \\ \mathbf{0} & \mathbf{A}^{*}_{\mathbf{w}}(k) \end{bmatrix} \begin{bmatrix} \mathbf{z}(i) \\ \mathbf{w}^{*}(i) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \boldsymbol{\nu}(i) \ \ \forall i = k, ..., k+N-1 \tag{2.18b}$$

$$\mathbf{z}(i) \in \bar{\mathcal{X}}(k) \ \ \forall i = k, ..., k+N \tag{2.18c}$$

$$\boldsymbol{\nu}(i) \in \bar{\mathcal{U}}(k) \ \ \forall i = k, ..., k+N-1 \tag{2.18d}$$

$$\mathbf{z}(k) \in \mathbf{x}(k) \oplus -\mathcal{Z}(k) \tag{2.18e}$$

$$\mathbf{w}^{*}(k) = \big[\widehat{\mathbf{w}}(k), \widehat{\mathbf{w}}(k-1),\ ...\ \widehat{\mathbf{w}}(k-n_a+1)\big]^{\top} \tag{2.18f}$$

Notice that the sets $\mathcal{Z}(k)$, $\bar{\mathcal{X}}(k)$, and $\bar{\mathcal{U}}(k)$ vary depending on the time-step k: this is because the tube is constantly being updated based on the latest measured exogenous inputs $\mathbf{w}$. The identified exogenous input dynamic model $\mathbf{A}^{*}_{\mathbf{w}}(k)$ is also updated at each time-step based on this data. This problem is still a QCQP, and can be solved via SOCP methods.

The final control input value applied to the real system, $\mathbf{u}(k)$, is calculated via the control law

$$\mathbf{u}(k) = \mathbf{v}(k) + \mathbf{K}_r(k)\big(\mathbf{x}(k) - \mathbf{z}(k)\big) \tag{2.19}$$

Notice that the only difference when compared to (2.7) is that $\mathbf{K}_r(k)$ now varies from time-step to time-step since the tube is being updated.

2.3.2 Exogenous Input Estimation and Tube Updates

Estimating the Next Exogenous Input

Once the control input has been applied to the system, the exogenous input for time-step k can be measured as the difference from the real state at $k+1$ and the nominally expected state:

$$\mathbf{w}(k) = \mathbf{x}(k+1) - \mathbf{A}\mathbf{x}(k) - \mathbf{B}\mathbf{u}(k). \tag{2.20}$$

Since the exogenous input at time-step k is only measurable at time-step $k+1$, it is imperative to estimate its next value, denoted as $\widehat{\mathbf{w}}(k+1)$. This allows for the control input at step $k+1$ to utilize the estimated exogenous input in the nominal model.

A simple forgetting factor algorithm, based on the work of [20], is implemented to estimate the next value $\widehat{\mathbf{w}}(k+1)$. The estimate covariance $\widehat{\mathbf{\Sigma}}_{\mathbf{w}}(k+1)$ is also provided in this algorithm. This covariance is directly used for determining a suitable set $\widehat{\mathcal{W}}(k+1)$ that best encapsulates the possible values of the exogenous input $\mathbf{w}(k+1)$. Defining the recursive sum of a geometric progression as

$$\zeta(k+1) \triangleq \sum_{i=0}^{k} e^{(\lambda-1)(k+1-i)} = \frac{e^{(\lambda-1)}\left[1-e^{(\lambda-1)(k+1)}\right]}{1-e^{(\lambda-1)}} \ , \tag{2.21}$$

we can obtain the following estimates for the next exogenous input $\widehat{\mathbf{w}}(k+1)$ and the covariance $\widehat{\mathbf{\Sigma}}_{\mathbf{w}}(k+1)$:

$$\widehat{\mathbf{w}}(k+1) = \frac{e^{(\lambda-1)}}{\zeta(k+1)}\big(\zeta(k)\widehat{\mathbf{w}}(k) + \mathbf{w}(k)\big) \tag{2.22a}$$

$$\widehat{\mathbf{\Sigma}}_{\mathbf{w}}(k+1) = \frac{e^{(\lambda-1)}}{\zeta(k+1)}\Big(\zeta(k)\widehat{\mathbf{\Sigma}}_{\mathbf{w}}(k) + \big(\mathbf{w}(k)-\widehat{\mathbf{w}}(k)\big)\big(\mathbf{w}(k)-\widehat{\mathbf{w}}(k)\big)^{\top}\Big) \ . \tag{2.22b}$$

The λ term (i.e., the forgetting factor) is a constant design parameter in the range $(0,1)$. Values of λ that approach 1 result in more weighting of past measured data in the estimate. Lower values of λ produce more dynamic estimates that largely rely on only the past few measurements of the exogenous input $\mathbf{w}$. Thus, there is a trade-off between utilizing more data for estimates and quickly capturing dynamic patterns of the exogenous input. Figure 2-7 shows the application of this estimation algorithm for a component of the exogenous input vector. Note that the estimation algorithm is able to quickly adjust the estimate of the true exogenous input and encapsulate it via the 3-σ bounds.

Estimating the Set of Exogenous Inputs via the Covariance

The estimated covariance allows us to define a ellipsoidal set $\widehat{\mathcal{W}}(k+1)$ that provides the best estimate of the bounds on the exogenous input $\mathbf{w}(k+1)$. A constant scaling

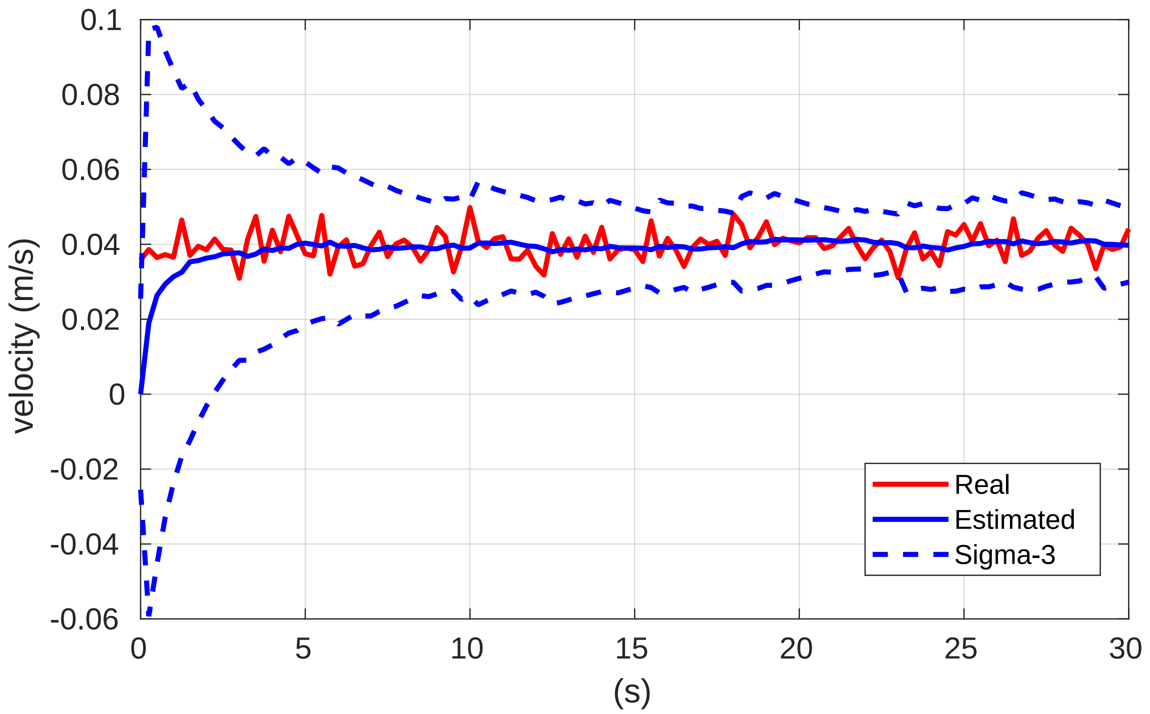


Figure 2-7: Estimation results by applying the forgetting factor algorithm to a noisy component of the exogenous input (in this case, the velocity component for the 2-D double-integrator system).

factor s is applied to the covariance that results in an ellipsoid $\widehat{\mathcal{W}}(k+1)$ that encapsulates $\mathbf{w}(k+1)$ with a probability $p_{\mathbf{w}}$. The probability $p_{\mathbf{w}}$ is chosen as a design parameter and thus yields the corresponding scaling factor s. This process of scaling the covariance ellipsoid to produce the estimated set of exogenous inputs $\widehat{\mathcal{W}}(k+1)$ is formally defined as

$$P(\chi^2(n) \leq s) = p_{\mathbf{w}} \tag{2.23a}$$

$$\widehat{\mathcal{W}}(k+1) : \big(\mathbf{w}(k+1) - \widehat{\mathbf{w}}(k+1)\big)^\top \big(s\widehat{\mathbf{\Sigma}}_{\mathbf{w}}(k+1)\big)^{-1}\big(\mathbf{w}(k+1) - \widehat{\mathbf{w}}(k+1)\big) \leq 1 \tag{2.23b}$$

where $\chi^2(n)$ is the chi-square distribution with the number of degrees of freedom equal to the state dimension. For instance, if $n = 2$ and $p_{\mathbf{w}} = 0.99$, the scaling factor $s = 9.2103$. Figure 2-8 depicts several ellipsoidal sets $\widehat{\mathcal{W}}(k+1)$ for increasing values of $p_{\mathbf{w}}$.

Having obtained an updated exogenous input set $\widehat{\mathcal{W}}(k+1)$, it is straight-forward

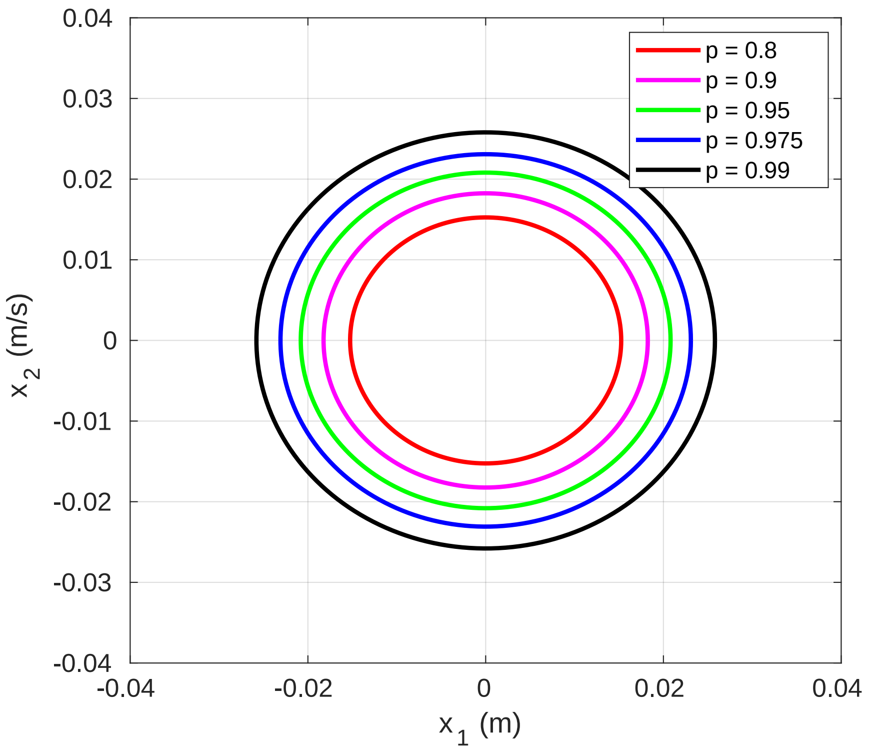


Figure 2-8: An depiction of increasingly larger definitions of $\widehat{\mathcal{W}}(k+1)$ based on increasing probabilities that the set contains the possible values of $\mathbf{w}(k+1)$. The probability levels are set as design parameters for the online tube-based MPC algorithm.

to compute an updated tube set $\mathcal{Z}(k+1)$ and exogenous input rejection gain $\mathbf{K}_r(k+1)$ using (2.10) and (2.11). The constraints are also shrunk again via (2.12) using the updated tube $\mathcal{Z}(k+1)$. Together, these steps ensure robustness for the next time-step.

2.3.3 Exogenous Input Prediction

If it is inferred that the nature of the exogenous inputs exhibits dynamic patterns and is not purely stochastic, an identified model can be utilized to predict their future values. In this work, the linear dynamic model is formulated as an ARMA model [34].

Autoregressive-Moving-Average Model Definition

An ARMA model consists of the parameter vector $\boldsymbol{\theta}_A \in \mathbb{R}^{n_a+n_c}$:

$$\boldsymbol{\theta}_A = [a_1, a_2, ..., a_{n_a}, c_1, c_2, ..., c_{n_c}]^\top. \tag{2.24}$$

where n_a is the number of poles and n_c is the number of coefficients used in modeling the error terms during identification (thus accounting for noise in the data). The output $y(k+1)$ of an ARMA model for a single-output system is then obtained as

$$y(k+1) = -a_1 y(k) - ... - a_{n_a} y(k - n_a + 1) + c_1 \epsilon(k) + ... + c_{n_c} \epsilon(k - n_c + 1) \tag{2.25}$$

where $\epsilon(k)$ is equal to the prediction error caused by white noise entering the system. It is clear that there are two distinct parts to (2.25): the first part is the autoregressive part, which essentially regresses the past values of y to yield the prediction of $y(k+1)$ using the parameters $a_1, ..., a_{n_a}$. The second part models the current noise in the data as a moving, weighted average of past error values using the parameters $c_1, ..., c_{n_c}$. The number of parameters n_a and n_c are tunable design values: greater values utilize more past measurements, but also induce more model complexity (which in turn requires a larger dataset for system identification convergence).

Predicting Exogenous Inputs via ARMA Models

In the context of predicting future exogenous inputs for online, tube-based MPC, the model structure defined in (2.25) is applied to predict each element of the exogenous input vector $\mathbf{w}$ (thus there is an ARMA model per element of $\mathbf{w}$). In a state transition formulation, the state is defined as a stack of past predicted exogenous input values:

$$\mathbf{w}^*(k) \in \mathbb{R}^{n(n_a)} = \left[\widehat{\mathbf{w}}(k)^\top, \widehat{\mathbf{w}}(k-1)^\top, \ \dots \ , \widehat{\mathbf{w}}(k - n_a + 1)^\top\right]^\top . \tag{2.26}$$

Since the error terms $\epsilon(k)$ cannot be propagated throughout the MPC prediction horizon (there is no available real data to measure errors), only the autoregressive parameters of the ARMA models are used in the state transition matrix. Before defining the full state transition matrix $\mathbf{A}^*_{\mathbf{w}}(k) \in \mathbb{R}^{n(n_a) \times n(n_a)}$, smaller $n \times n$ matrices $\mathbf{A}^*(k),\ \mathbf{A}^*(k-1),\ \dots, \mathbf{A}^*(k-n_a+1)$ are defined as

$$\mathbf{A}^*(k) \triangleq \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{21} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n1} \end{bmatrix},\ \mathbf{A}^*(k-1) \triangleq \begin{bmatrix} a_{12} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n2} \end{bmatrix},\ \cdots \tag{2.27}$$

where a_{11} corresponds to the ARMA a_1 model parameter for the first element of $\mathbf{w}$, a_{21} corresponds to the ARMA a_1 model parameter for the second element of $\mathbf{w}$, and so on. This finally allows us to define the full state transition matrix used in predicting future values of $\mathbf{w}$:

$$\mathbf{A}^*_{\mathbf{w}}(k) = \begin{bmatrix} \mathbf{A}^*(k) & \mathbf{A}^*(k-1) & \cdots & \mathbf{A}^*(k-n_a+2) & \mathbf{A}^*(k-n_a+1) \\ \mathbf{I}_n & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdot & \mathbf{I}_n & \mathbf{0} \end{bmatrix} \tag{2.28}$$

This state transition matrix corresponds to the one used in equations (2.17) and (2.18).

Parameter Updates for the Identified Model

Predicting future values of the exogenous input $\mathbf{w}$ in this way results in a stack of parameter vectors $\boldsymbol{\theta}_{\mathbf{w}} \in \mathbb{R}^{n(n_a+n_c)}$ for each ARMA model that is dedicated to each element of $\mathbf{w}$. These parameters are updated at each time-step via a recursive least-squares algorithm [34]. Summarizing the recursive least-squares parameter update for one of the ARMA models $\boldsymbol{\theta}_A$ that predicts an element w of the $\mathbf{w}$ vector, we define the regression vector as

$$\boldsymbol{\phi}(k) \triangleq [w(k-1),\ \dots\ w(k-n_a), \epsilon(k-1),\ \dots\ \epsilon(k-n_c)]^\top . \tag{2.29}$$

We can then outline the rest of the recursive least-squares algorithm that yields the updated parameter vector $\boldsymbol{\theta}_A(k+1)$:

$$\mathbf{K}_A(k) = \frac{\mathbf{P}_A(k)}{\lambda_A + \boldsymbol{\phi}^\top(k)\mathbf{P}_A(k)\boldsymbol{\phi}(k)}\boldsymbol{\phi}(k) \tag{2.30a}$$

$$\boldsymbol{\theta}_A(k+1) = \boldsymbol{\theta}(k) + \mathbf{K}_A(k)\Big[w(k) - \boldsymbol{\phi}^\top(k)\boldsymbol{\theta}_A(k)\Big] \tag{2.30b}$$

$$\mathbf{P}_A(k+1) = \frac{1}{\lambda_A}\Big(\mathbf{P}_A(k) - \mathbf{K}_A(k)\boldsymbol{\phi}^\top(k)\mathbf{P}_A(k)\Big) \tag{2.30c}$$

where $\mathbf{K}_A \in \mathbb{R}^{n_a+n_c}$ is the parameter update gain, $\mathbf{P}_A \in \mathbb{R}^{(n_a+n_c)\times(n_a+n_c)}$ is the error covariance in parameter estimation, and λ_A is a forgetting factor term used similarly to the λ term in (2.22).

Demonstration for a Deterministic Exogenous Input Pattern

Figure 2-9 depicts an application of this online system identification method using $n_a = 2$ and $n_c = 1$ for an exogenous input that exhibits an oscillating dynamic pattern. The ability to predict future values of $\mathbf{w}$ solely based on measured data allows the exogenous input to be incorporated into nominal MPC model throughout the entire prediction horizon. This allows for potential performance gains since the MPC optimization step more accurately models the evolution of the real system.

However, the above method of online system identification can yield poor results if the exogenous input follows an almost entirely stochastic pattern, as it would require many data points to uncover any sort of deterministic, dynamic pattern. The identified model's performance is also poor in these cases since the estimated exogenous input values $\widehat{\mathbf{w}}$ used to initialize the sequence of predicted values in the MPC optimization step can be significantly noisy. Thus, it is suggested that if the magnitude of the covariance matrix $\widehat{\boldsymbol{\Sigma}}_{\mathbf{w}}(k)$ is above a design threshold (indicating significantly noisy exogenous input data), the exogenous input prediction should be disabled by setting $\mathbf{A}^*_{\mathbf{w}} = \mathbf{0}$ for all time-steps past k in the MPC prediction horizon.

It is also noted that the ARMA formulation for performing recursive, online system identification is a fairly general approach to the problem. This gives an advantage of

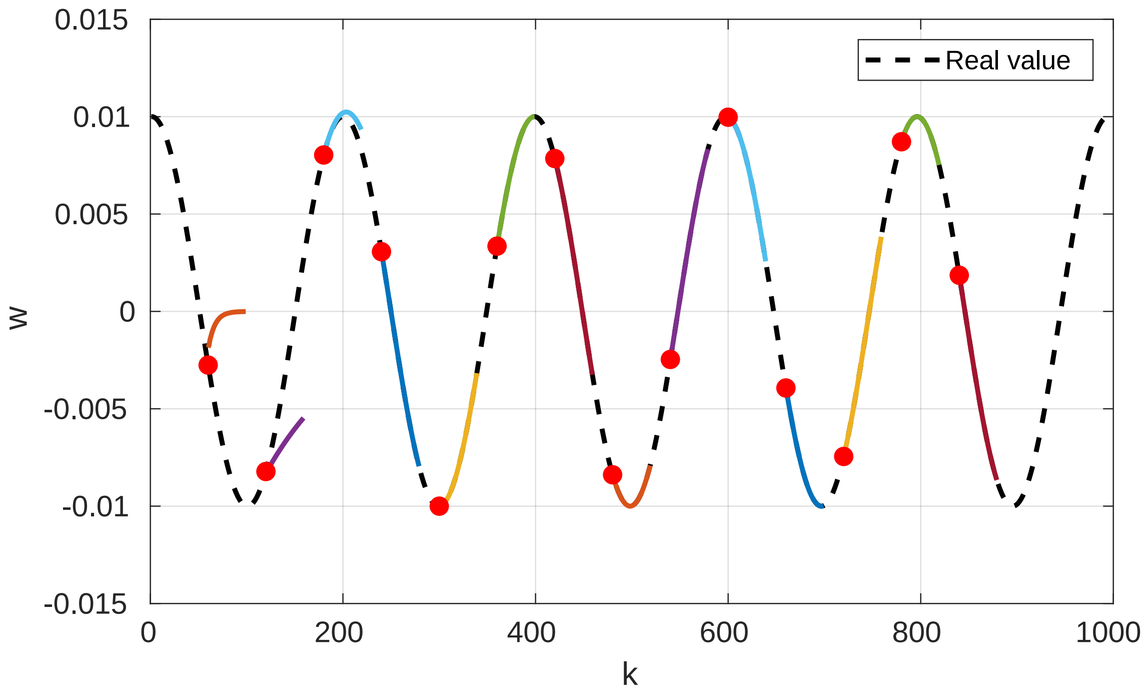


Figure 2-9: Applying the ARMA-based, online system identification method to predict future values of the exogenous input. In this case, the exogenous input (dashed line) follows a distinct oscillating pattern; the predictions throughout the MPC horizon are given by colored curves at selected intervals.

being flexible for many different exogenous input scenarios. However, this also means it requires a sizeable amount of data points before the identified model parameters properly converge. Thus, if there are unique characteristics known about the exogenous inputs in a particular application of online tube-based MPC, it is potentially advantageous to investigate more specific model structures that are better tuned for the system at hand.

2.3.4 Application to the 2-D Double-Integrator System

The online, tube-based MPC algorithm is applied to the 2-D double-integrator system defined in Appendix B. Figure 2-10 depicts a resulting trajectory produced by the controller for a case when the exogenous inputs $\mathbf{w}$ are sampled using $\sigma_r = 4 \times 10^{-3}$, $\sigma_v = 4 \times 10^{-3}$ and $\boldsymbol{\mu}_\mathbf{w} = \mathbf{0}$. However, the initial estimate of the exogenous input set was obtained using $\hat{\sigma}_r = 8.5 \times 10^{-3}$, $\hat{\sigma}_v = 8.5 \times 10^{-3}$, and $\hat{\boldsymbol{\mu}}_\mathbf{w} = \mathbf{0}$ making the set $\widehat{\mathcal{W}}(0)$ unnecessarily conservative. The prediction horizon was $N = 5$, while the cost

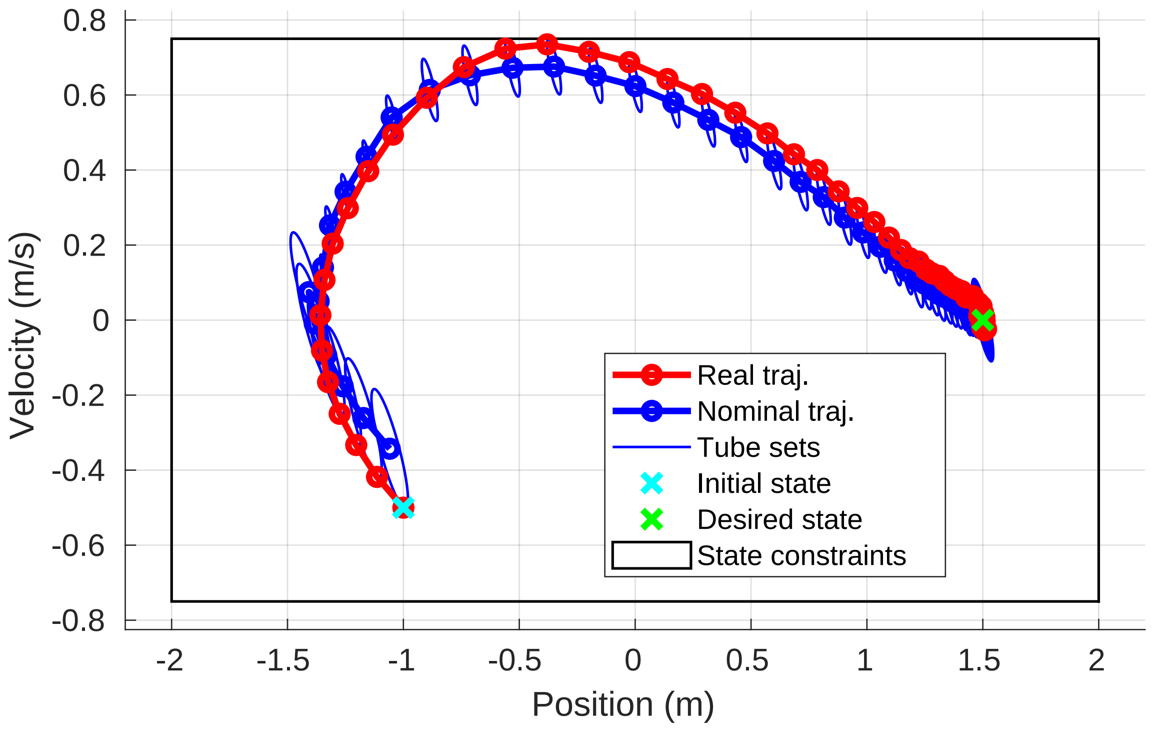


Figure 2-10: A trajectory produced by the online tube-based controller when the initial estimate of the exogenous input set is over-conservative. Notice that the tube shrinks immediately after the 5-step warm-up period in accordance with the measured exogenous input data. By shrinking the tube, more control authority is allocated to the nominal MPC optimization step and thus performance is increased.

matrices were set as $\mathbf{Q} = \mathbf{I}_2$ and $\mathbf{R} = 10$. The $\mathbf{P}$ matrix was obtained via LQR design using $\mathbf{Q}$ and $\mathbf{R}$. Exogenous input estimation parameters were set as $\lambda = 0.875$ and $p_{\mathbf{w}} = 0.99$. Tube updates did not take place until a estimation warm-up period of 5 time-steps was completed. Note that the exogenous input prediction was disabled in this case since a), there are a very small amount of available data points for system ID in this toy problem, and b), the exogenous input is purely stochastic. As shown in the trajectory, the tube size shrinks as the algorithm adjusts its estimate of exogenous input set $\widehat{\mathcal{W}}(k)$ based on the measured exogenous input data. This naturally allows for more optimization in the nominal MPC problem and produces better performance when compared to standard tube-based MPC.

Figure 2-11 shows another trajectory produced by the online, tube-based MPC in the same case where the standard tube-based MPC failed. The online tube-based MPC is able to successfully estimate the exogenous input and adjust the tube size

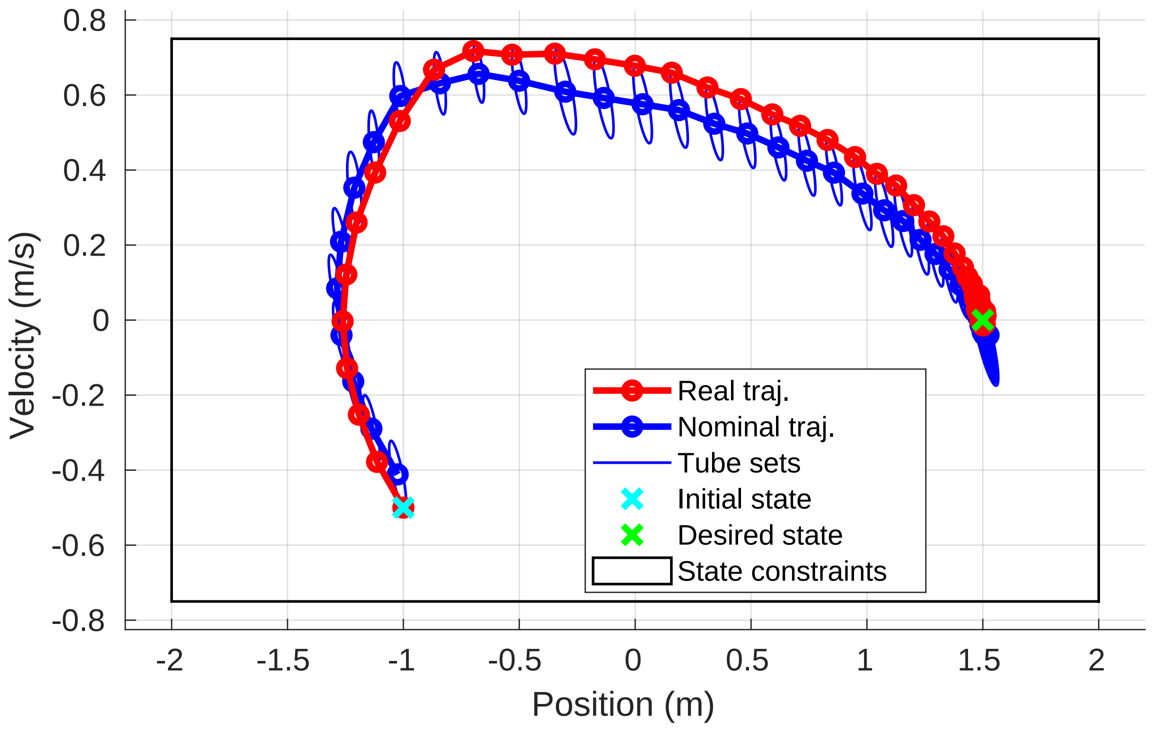


Figure 2-11: A successful trajectory produced by the online tube-based MPC for the disturbed 2-D double-integrator system. In the same exogenous input case where the standard tube-based MPC failed (Figure 2-5), the online tube-based MPC was able to adjust the estimate of the exogenous input and the tube size to accommodate a strong velocity component bias that was not included in the initial estimate of the exogenous input set.

accordingly to accommodate the strong velocity component bias, thus reclaiming robust properties. Notice that the nominal states near the end of the trajectory have a negative velocity value; this is because the controller knows there is a strong positive velocity bias acting on the real state.

Chapter 3

Robust Control for Intercepting a Tumbling Target

This chapter focuses on the application of the aforementioned robust control methods to intercept a tumbling target with unknown inertial properties. First, an overview of the problem including its its dynamics, constraints, and uncertainty sources is provided. Then, the simulation environment for the maneuver is presented, and a complete controller framework is formulated to apply online, tube-based MPC. Finally, the results obtained by applying the controller to the tumbling target problem are shared and discussed.

3.1 Problem Definition

This section provides context on the tumbling object intercept problem and defines the relevant dynamics and constraints. A discussion of how uncertainty enters the system is also included.

3.1.1 Mission Overview

Defunct satellites or debris objects often exhibit uncontrolled, tumbling motion due to the accrual of disturbances such as drag, gravity gradient, or solar radiation pressure.

Figure 3-1: An illustration of Envisat, a large, defunct satellite that is exhibiting a uncontrolled tumble while residing in a high-density LEO orbit. Image credit: European Space Agency.

A notable example is Envisat [29] (Figure 3-1), whose orbit and large size poses a risk for collisions. As such, autonomous on-orbit servicing or active debris removal efforts must be prepared to intercept tumbling targets whose inertial properties may be unknown.

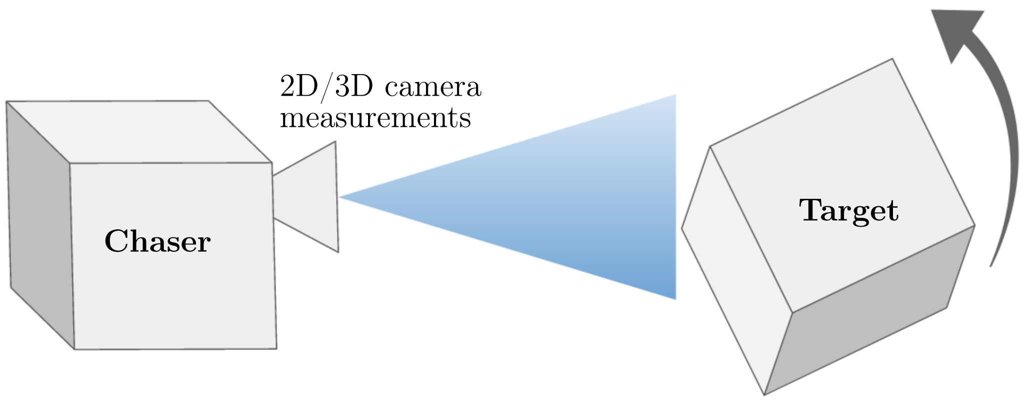
A typical pipeline for intercepting a tumbling target follows three distinct phases (Figure 3-2):

1. Estimation of the target's rotational state and inertial properties.

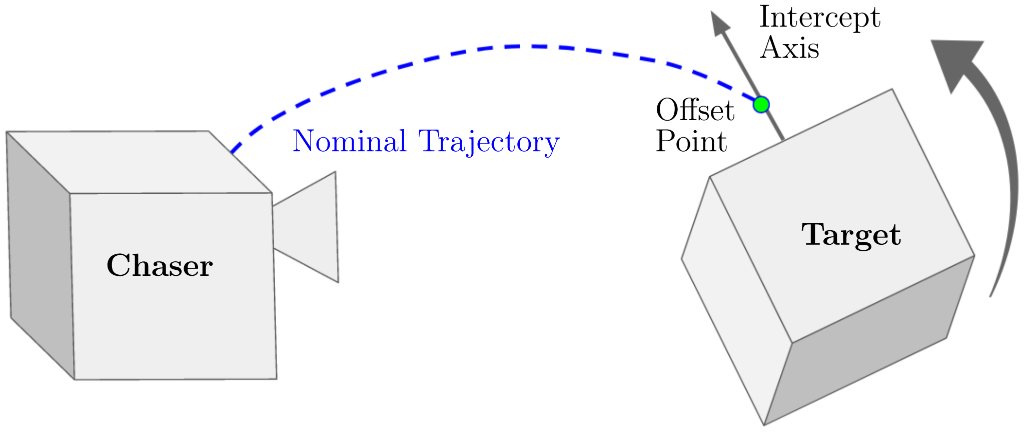
2. Motion planning to determine an optimal, collision-free intercept trajectory.

3. Tracking control to follow the collision-free trajectory despite errors in the estimation and modeling of the target.

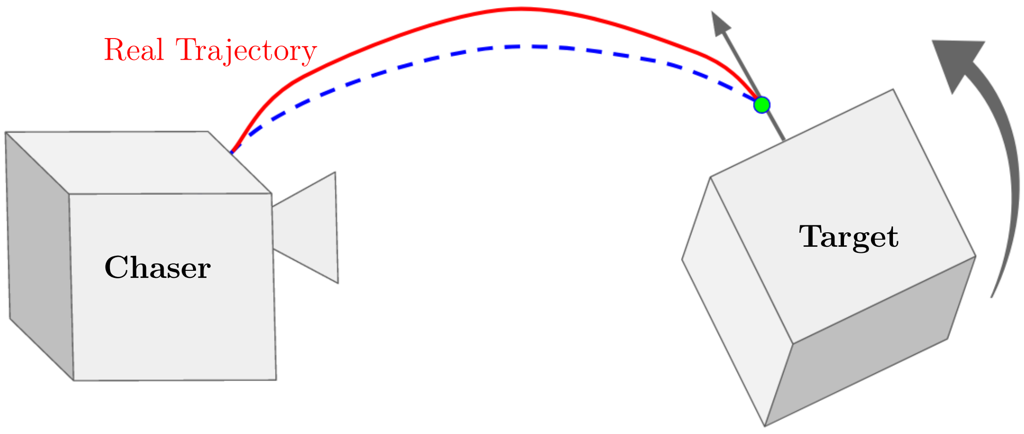
Various frameworks exist for estimating the target's state and inertial properties, including filtering-based [4] and simultaneous localization and mapping approaches



(a) Phase 1: Estimation of the target's rotational state and inertial properties.



(b) Phase 2: Planning the nominal intercept trajectory.



(c) Phase 3: Tracking the real trajectory that accounts for estimation errors.

Figure 3-2: A depiction of the mission phases required to autonomously intercept an unknown, uncontrolled tumbling target.

[62], [55], [14]. However, the estimation of the target's inertia tensor is a challenging task due to measurement error and the potential complexity of the target's tumbling motion. Thus, there will always be some degree of error in the estimated target's inertia tensor versus its true value.

The motion planning step uses the estimated target state and inertia tensor to predict its motion and determine an optimal, collision-free trajectory for the chaser to follow and intercept the target. To successfully intercept the target, the chaser must reach a desired offset distance in the target body frame and synchronize its motion with the target's tumble. At this point, a robotic manipulator could reach out and grapple the target. Collision avoidance and motion synchronization is defined in the body frame of the target; thus, the resulting motion plan itself is defined in the target's body frame as well.

This is a complex motion planning problem due to the 6-DOF, nonlinear dynamics and non-convex collision avoidance constraints. For instance, in cases such as Envisat, there are large solar panels and antennae that produce time-varying collision zones. Thus, optimized trajectories can significantly differ depending on the target's inertia tensor and determined initial rotational state. A state-of-the-art motion planner for this task is developed in [58], which utilizes nonlinear optimization techniques and a look-up-table to provide the optimal, collision-free trajectory in the target's body frame.

The main focus of this thesis is on the tracking control step, where the chaser actually attempts to follows the nominal trajectory produced by the motion planner and intercept the target. Since there will be error present in the target's estimated inertia tensor, the predicted motion used by the motion planner in creating the nominal trajectory will differ from the real tumbling motion of the target. As such, the nominal relationship between the inertial and target body frames is disturbed, and the controller must be robust to this uncertainty in order to successfully avoid collisions and synchronize the motion at the desired offset point. In [12], a standard tube-based MPC was developed for this task. The focus of this chapter is on advancing this controller framework to adapt to the measured uncertainty online in order

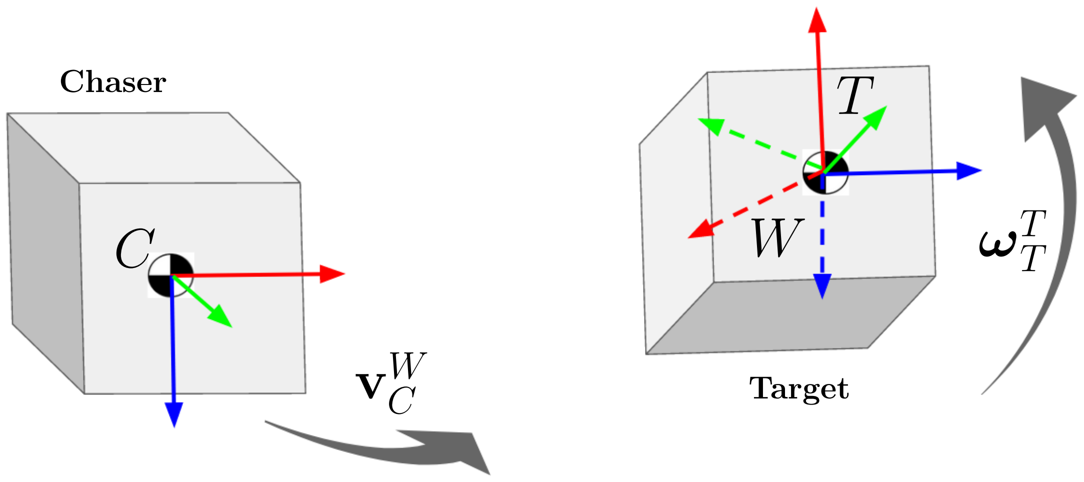


Figure 3-3: An overview of the relevant coordinate frames and state variables for intercepting a tumbling target. Red/green/blue indicate the x/y/z axes, respectively.

to increase performance and expand the flexibility of the overall approach. The next section outlines the dynamics involved for the tumbling object intercept problem.

3.1.2 Dynamics and Constraints

Coordinate Frames

Figure 3-3 depicts the relevant coordinate frames for this problem. The W frame represents the world (i.e., inertial) frame and is situated at the target's center of mass. The T frame represents the target's body frame, whose origin is also located at the target's center of mass. The C frame represents the chaser's body frame, whose origin is located at the chaser's center of mass. The goal of the controller is to bring the C frame to the stationary offset point in the T frame.

State Definitions

The scope of this work only focuses on controlling the chaser's 3-DOF translational motion to track the desired trajectory; it is assumed that there is a sufficient attitude controller to keep the target in the chaser's field of view and synchronize the relative attitude. This allows the tracking controller to be formulated with a linear model, which in turn provides a straightforward path to robustness guarantees. As such, the chaser's state is defined as

$$\mathbf{x}_C^W \triangleq \left[\left(\mathbf{r}_C^W\right)^\top, \left(\mathbf{v}_C^W\right)^\top\right]^\top \in \mathbb{R}^6 \tag{3.1}$$

where $\mathbf{r}_C^W \in \mathbb{R}^3$ is the chaser's position with respect to the W frame and $\mathbf{v}_C^W \in \mathbb{R}^3$ is the chaser's velocity with respect to the W frame.

The target is assumed to be stationary with respect to translation at the origin of the W frame. Thus, the only relevant target state variables are its attitude with respect to the W frame $\mathbf{R}_T^W \in SO(3)$ and its angular velocity with respect to the T frame $\boldsymbol{\omega}_T^T \in \mathbb{R}^3$ where $SO(3)$ is the special group of orthogonal 3×3 matrices that describe 3D rotations. Attitude can also be described via unit quaternions $\mathbf{q} = \left[\left(\mathbf{q}_v\right)^\top, q_w\right]^\top$ where $\mathbf{q}_v = \left[q_x, q_y, q_z\right]^\top \in \mathbb{R}^3$ is the vector part and $q_w \in \mathbb{R}$ is the scalar part. Quaternions offer a simpler way to model the target's tumbling dynamics. Conversions from rotation matrices to quaternions are performed by

$$\mathbf{q} = \begin{bmatrix} \frac{1}{4q_w}(R_{32} - R_{23}) \\ \frac{1}{4q_w}(R_{13} - R_{31}) \\ \frac{1}{4q_w}(R_{21} - R_{12}) \\ \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \end{bmatrix} \tag{3.2}$$

and vice versa

$$\mathbf{R} = \left(q_w^2 - \mathbf{q}_v^\top \mathbf{q}_v\right) \mathbf{I}_3 + 2\mathbf{q}_v \mathbf{q}_v^\top + 2q_w \left[\mathbf{q}_v\right]_\times \tag{3.3}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix and $\left[\mathbf{p}\right]_\times$ is the cross-product matrix operator for any vector $\mathbf{p} = [p_x, p_y, p_z]^\top \in \mathbb{R}^3$:

$$\left[\mathbf{p}\right]_\times \triangleq \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}. \tag{3.4}$$

Alternate formulations of (3.2) are available if $q_w \approx 0$ [56].

Dynamics

We assume that the distance between the chaser and target at the beginning of the intercept maneuver is relatively small and orbital effects are inconsequential. Thus, we avoid the usage of the Hill-Clohessy-Wiltshire relative orbital dynamics and instead adopt a simple double-integrator model:

$$\dot{\mathbf{r}}_C^W = \mathbf{v}_C^W \tag{3.5a}$$

$$\dot{\mathbf{v}}_C^W = \frac{\mathbf{F}}{m} \tag{3.5b}$$

where $\mathbf{F} \in \mathbb{R}^3$ is the applied thrust in each direction of the W frame and $m \in \mathbb{R}$ is the chaser's mass.

Since this is a linear system, it can be written in the state-space form (continuous time)

$$\dot{\mathbf{x}}_C^W = \mathbf{A}\mathbf{x}_C^W + \mathbf{B}\mathbf{u} \tag{3.6}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} \tag{3.7}$$

and $\mathbf{u} \triangleq \frac{\mathbf{F}}{m} \in \mathbb{R}^3$ is the control input to the system.

The target is assumed to be a rigid body and follow torque-free tumbling dynamics. The tumbling dynamics are modeled through Euler's equations:

$$\dot{\mathbf{q}}_T^W = \frac{1}{2}\boldsymbol{\Omega}\Big(\boldsymbol{\omega}_T^T\Big)\mathbf{q}_T^W \tag{3.8a}$$

$$\dot{\boldsymbol{\omega}}_T^T = \mathbf{J}_T^{-1}\Big(-\boldsymbol{\omega}_T^T \times \mathbf{J}_T\boldsymbol{\omega}_T^T\Big) \tag{3.8b}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) \triangleq \begin{bmatrix} 0 & -\boldsymbol{\omega}^\top \\ \boldsymbol{\omega} & -\Big[\boldsymbol{\omega}\Big]_\times \end{bmatrix} \tag{3.9}$$

and $\mathbf{J}_T \in \mathbb{R}^{3\times 3}$ is the target's inertia tensor.

Constraints

As is typical of most autonomous spacecraft maneuvers, the chaser must satisfy constraints on the state and control inputs. For the tumbling object intercept maneuver, the collision constraint is already handled by the nominal reference trajectory in the target body frame that is produced by the motion planner. Thus, the only constraints the controller must consider while operating online are limits on the position, velocity, and control inputs.

The position constraint essentially forms a box in which the maneuver must stay. In practice, this keep-in-box is large; thus the position constraint is rarely active and generally does not affect controller performance. The velocity constraint limits the maximum chaser velocity in each axis to a set value. Depending on the context of the specific mission, this constraint could be used to both a) reduce the overall risk of the maneuver and b) assist the relative navigation algorithm. Finally, the control input constraint represents the maximum available thrust in each axis for the chaser. This particular constraint typically has the most influence on the controller's performance as it directly inhibits the chasers ability to track the nominal reference trajectory if the target is spinning at a considerable rate.

3.1.3 Uncertainty Modeling

The source of uncertainty in this problem arises from errors in the estimated target's inertia tensor $\widehat{\mathbf{J}}_T$ when compared to the true value $\mathbf{J}_T$. Based on (3.8), the value of $\mathbf{J}_T$ directly influences the target's tumbling motion, which means that the predicted motion used to plan the reference trajectory will differ from the real motion of the intercept maneuver.

Therefore, the controller must react to changes in the W frame state that maintain the collision-free reference trajectory within the target body frame. The reference trajectory in the target's body frame is denoted as

$$\bar{\mathbf{x}}_C^T(k = 0, 1, ..., t_f) \triangleq \left[\bar{\mathbf{r}}_C^T(k = 0, 1, ..., t_f)^\top, \bar{\mathbf{v}}_C^T(k = 0, 1, ..., t_f)^\top\right]^\top \tag{3.10}$$

where $\bar{\mathbf{r}}_C^T$ and $\bar{\mathbf{v}}_C^T$ indicate the desired position and velocity, k indicates the time-step of each trajectory setpoint, and t_f indicates the final time-step. At each time-step k, this result is transformed into the W frame using the current estimate of the target's attitude $\mathbf{R}_T^W(k)$ and angular velocity $\boldsymbol{\omega}_T^T(k)$:

$$\bar{\mathbf{x}}_C^W(k) \triangleq \left[\bar{\mathbf{r}}_C^W(k)^\top, \bar{\mathbf{v}}_C^W(k)^\top\right]^\top \tag{3.11a}$$

$$\bar{\mathbf{r}}_C^W(k) = \mathbf{R}_T^W(k)\ \bar{\mathbf{r}}_C^T(k) \tag{3.11b}$$

$$\bar{\mathbf{v}}_C^W(k) = \mathbf{R}_T^W(k)\ \bar{\mathbf{v}}_C^T(k) - \boldsymbol{\omega}_T^W(k) \times \left(\mathbf{R}_T^W(k)\right)^\top\ \bar{\mathbf{r}}_C^W(k) \tag{3.11c}$$

where $\boldsymbol{\omega}_T^W = \mathbf{R}_T^W \boldsymbol{\omega}_T^T$ is the target's angular velocity expressed in the W frame. Transforming the reference trajectory setpoint into the W frame is necessary since the dynamic model used by the controller (3.6) is also formulated in the W frame.

The values of $\mathbf{R}_T^W(k)$ and $\boldsymbol{\omega}_T^T(k)$ are used along with the estimated inertia tensor $\widehat{\mathbf{J}}_T$ to propagate the target dynamics (3.8). This results in the expected target state $\bar{\mathbf{R}}_T^W(k+1)$ and $\bar{\boldsymbol{\omega}}_T^T(k+1)$. These values, along with the next reference trajectory setpoint $\bar{\mathbf{x}}_C^T(k+1)$, are used to obtain the nominally expected desired state for the controller, termed $\bar{\mathbf{z}}_C^W(k+1)$, in the same manner as in (3.11).

However, the real state of the target at time-step $k+1$ will differ from the expected values due to the error between $\widehat{\mathbf{J}}_T$ and the true inertia tensor $\mathbf{J}_T$. This results in an actual desired state in the W frame based on the real target attitude and angular velocity, termed $\bar{\mathbf{x}}_C^W(k+1)$. Thus, there is a difference between the nominal and real states the controller must track in the W frame in order to maintain the collision-free reference trajectory in the T frame. This divergence occurs over the time-step k and is modeled as an uncertain term since its value cannot be measured until time-step $k+1$ when a new target state estimate is available:

$$\mathbf{w}_C^W(k) \triangleq \bar{\mathbf{x}}_C^W(k+1) - \bar{\mathbf{z}}_C^W(k+1). \tag{3.12}$$

This uncertain term is modeled as an additive, exogenous input to the system to fit the robust control framework outlined in Chapter 2. The controller must handle

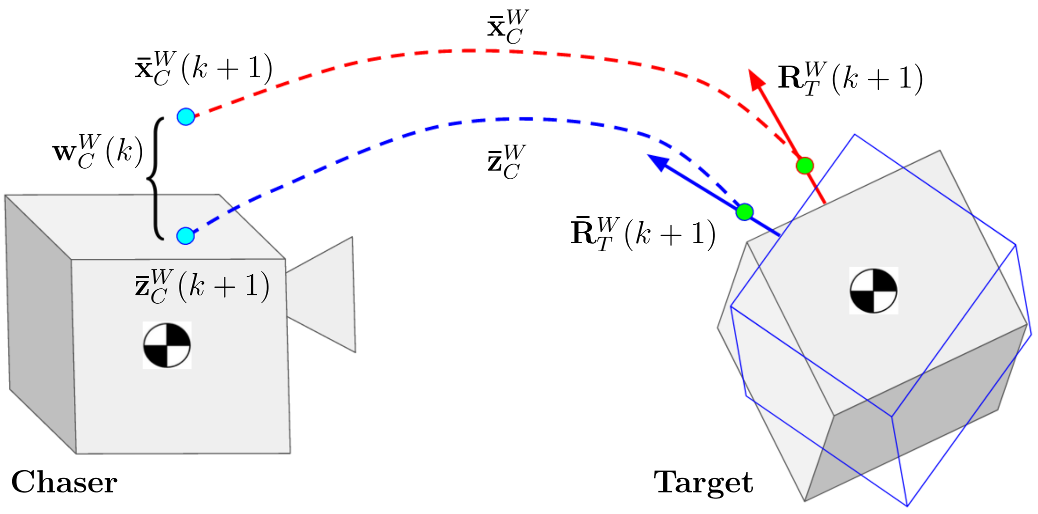


Figure 3-4: A depiction of how errors in estimating the target's inertia tensor creates uncertainty in the world frame as the chaser tries to maintain the reference trajectory in the target's body frame.

this exogenous input to the system at time-step k in order to track the correct, actual desired states $\bar{\mathbf{x}}_C^W$ and maintain the collision-free trajectory in the T frame that properly intercepts the target. Figure 3-4 provides a visualization of how the trajectory uncertainty in the W frame arises due to errors in estimating $\widehat{\mathbf{J}}_T$.

3.2 Simulation Environment

The following section formulates the simulation environment used for modeling the tumbling target intercept scenario. First, the spacecraft configurations are given. Then, the nominal trajectory provided by the motion planner for the scenario is presented and discussed. Finally, the methods used to simulate target inertia tensor estimate errors are shared.

3.2.1 Spacecraft Configurations

The simulation scenario begins at the end of the motion planning phase; the target's inertial properties have been estimated and the nominal trajectory has been produced by the motion planner. It is assumed that target attitude and angular velocity estimates are continuously available. The chaser starts from a resting state situated 3

Table 3.1: Initial conditions for simulating the intercept of a tumbling target.

Parameter	Value	Units
$\mathbf{r}_C^W(0)$	$[0, -3, 0]$	m
$\mathbf{v}_C^W(0)$	$[0, 0, 0]$	m/s
$\mathbf{q}_T^W(0)$	$[0.622, 0.307, -0.449, -0.564]$	-
$\boldsymbol{\omega}_T^T(0)$	$[0, 3.53, 3.53]$	deg/s

meters away from the target. The target's initial attitude is arbitrary, since the chaser could finish the estimation and motion planning phases at any particular moment in the target's tumbling motion pattern. The full list of initial state conditions for the chaser and target in are given in Table 3.1.

The target's angular velocity is initialized with $\boldsymbol{\omega}_T^T(0) = [0, 3.53, 3.53]$ deg/s. This is chosen to roughly match the expected tumbling rates of the Envisat spacecraft. The target's inertia tensor is also set to match Envisat's inertia tensor:

$$\mathbf{J}_T = \begin{bmatrix} 17023.3 & 397.1 & -2171.4 \\ 397.1 & 124825.7 & 344.2 \\ -2171.4 & 344.2 & 129112.2 \end{bmatrix} \text{kg}\cdot\text{m}^2 \tag{3.13}$$

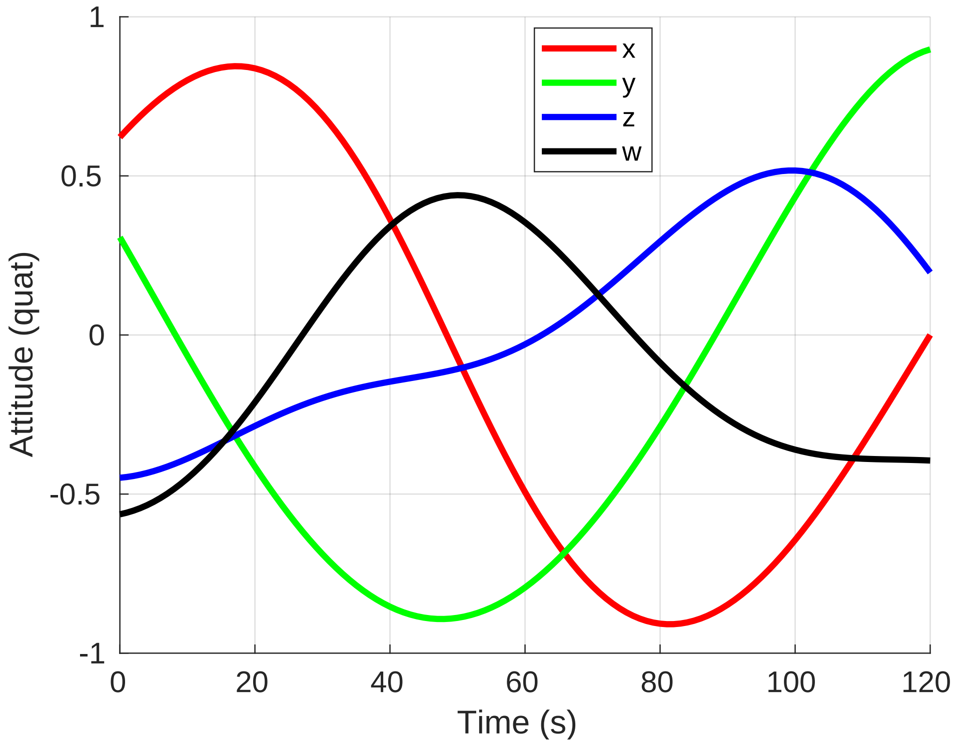
Despite the relatively slow tumbling rate, the initial angular velocity vector evolves over time in a nonlinear fashion according to (3.8). This is due to the inertia tensor being tri-axial and asymmetric. In simulation, the dynamics are propagated via 4th-order Runge Kutta numerical integration. Figure 3-5 shows the resulting tumbling trajectory for the target arising from the initial conditions and Envisat's inertia tensor.

The chaser's state is constrained within position and velocity box constraints. The minimum/maximum position values are ±8 meters in each axis. This creates a cube within which the trajectory must take place. The velocity in each axis is constrained to a maximum magnitude of 0.1 m/s. The chaser's control inputs (accelerations) are limited to a maximum magnitude of 0.075 m/s^2. This is the most critical constraint as the online, tube-based MPC design must appropriately allocate this finite control authority to both the nominal MPC optimization and the exogenous input rejection aspects. Both the state and control input constraints are formulated as polytopes (2.3), yielding the sets $\mathcal{X}$ and $\mathcal{U}$.

3.2.2 Nominal Trajectory

A nominal trajectory to intercept the target (given its estimated inertia tensor) is provided by the motion planning phase. The optimal trajectory reaches the desired intercept point with the correct rates to synchronize with the target's rotational motion while avoiding collisions with target spacecraft appendages. The simulated target spacecraft appendages are shown in Figure 3-6. The cost function is a combination of minimizing control input usage as well as maximizing the time to a collision if the chaser loses actuation capabilities. This optimization problem is formulated as a nonlinear boundary value problem, where trajectory solutions are parameterized as clamped, nonuniform B-splines. More details on the motion planner and its methods can be found in [58].

Utilizing the initial conditions from Table 3.1 and the estimated value of the target's inertia tensor $\widehat{\mathbf{J}}_T$ as inputs, the motion planner's solution yields the desired intercept trajectory in the target's body frame $\bar{\mathbf{x}}_C^T(k = 0, 1, ..., t_f)$. Note that the nominal trajectory has a fixed final time of 120 seconds to reach the intercept point of 0.5 m along the target's x-axis. Using up-to-date estimates of the target's rotational state enables the nominal trajectory to be transformed into the inertial frame ($\bar{\mathbf{x}}_C^W$) for robust tracking control (3.11). Figures 3-7 and 3-8 show the resulting nominal trajectory in both the target body and inertial frames, respectively, assuming perfect estimation of $\widehat{\mathbf{J}}_T$. However, as discussed earlier, errors in $\widehat{\mathbf{J}}_T$ cause divergence between the predicted motion used by the motion planner and the real motion. Since the motion planner is not capable of re-planning in real-time, the robust tracking controller must account for the exogenous inputs $\mathbf{w}_C^W$ that affect the trajectory in the inertial frame (3.12).



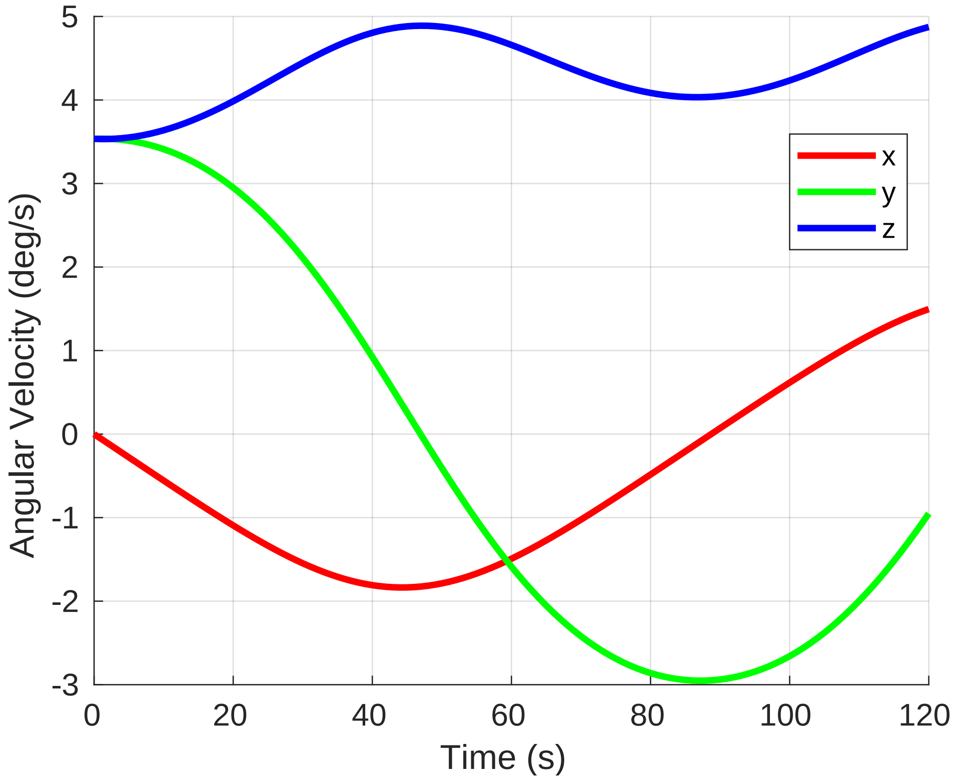


Figure 3-5: The resulting target tumbling trajectory over 120 seconds, produced by the initial conditions in Table 3.1 and Envisat's inertia tensor.

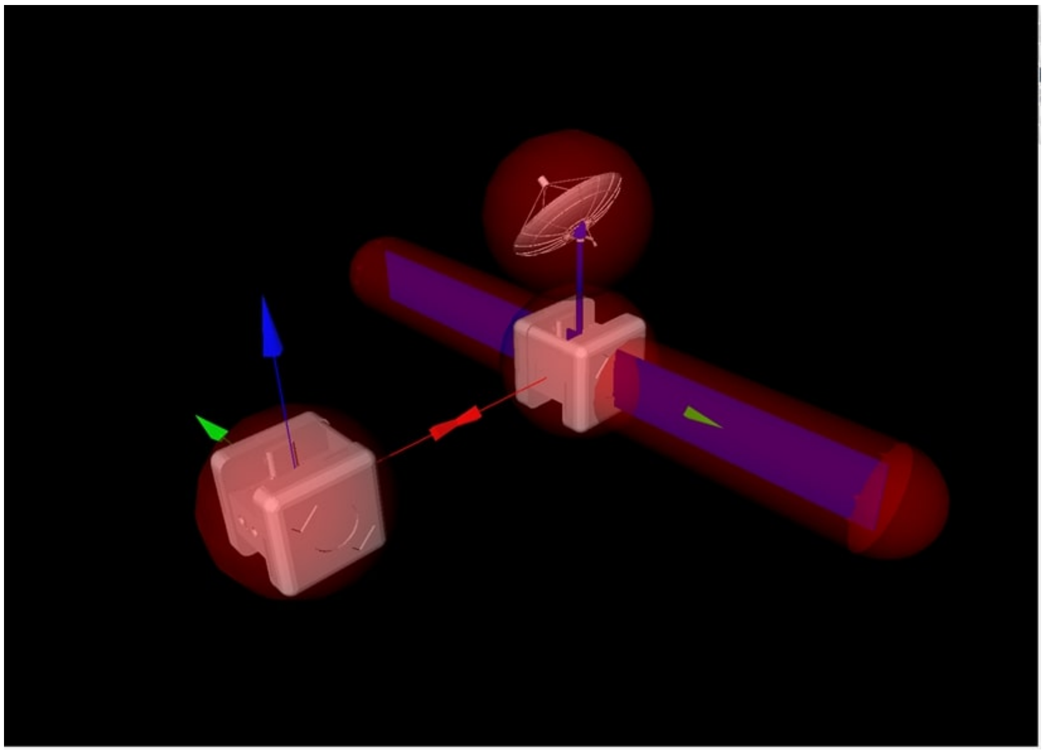
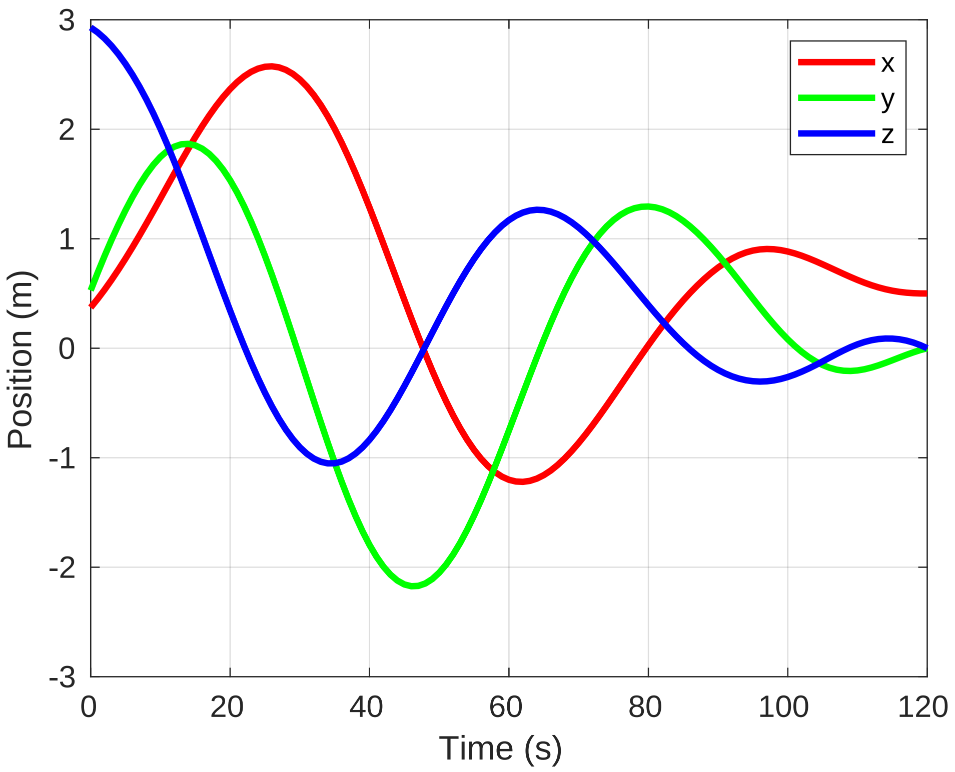


Figure 3-6: A depiction of the simulated chaser and target collision areas. A sphere encapsulates the chaser collision area; ellipsoids and spheres encapsulate the target and its appendages, which include solar panels and an antenna. Image credit: Caroline Specht and Roberto Lampariello, German Aerospace Center (DLR).

3.2.3 Simulating Inertia Estimation Errors

To better analyze the capabilities and performance of the robust tracking controller, different levels of inertia estimation error are chosen for numerous simulated trajectories. The estimated inertia tensor $\widehat{\mathbf{J}}_T$ for each test case always corresponds to Envisat's nominal value (3.13). The real inertia tensor for a particular simulated trajectory $\mathbf{J}_T$ is obtained by adding an error matrix $\widetilde{\mathbf{J}}_T$ to the nominal value:

$$\mathbf{J}_T = \widehat{\mathbf{J}}_T + \widetilde{\mathbf{J}}_T \tag{3.14}$$



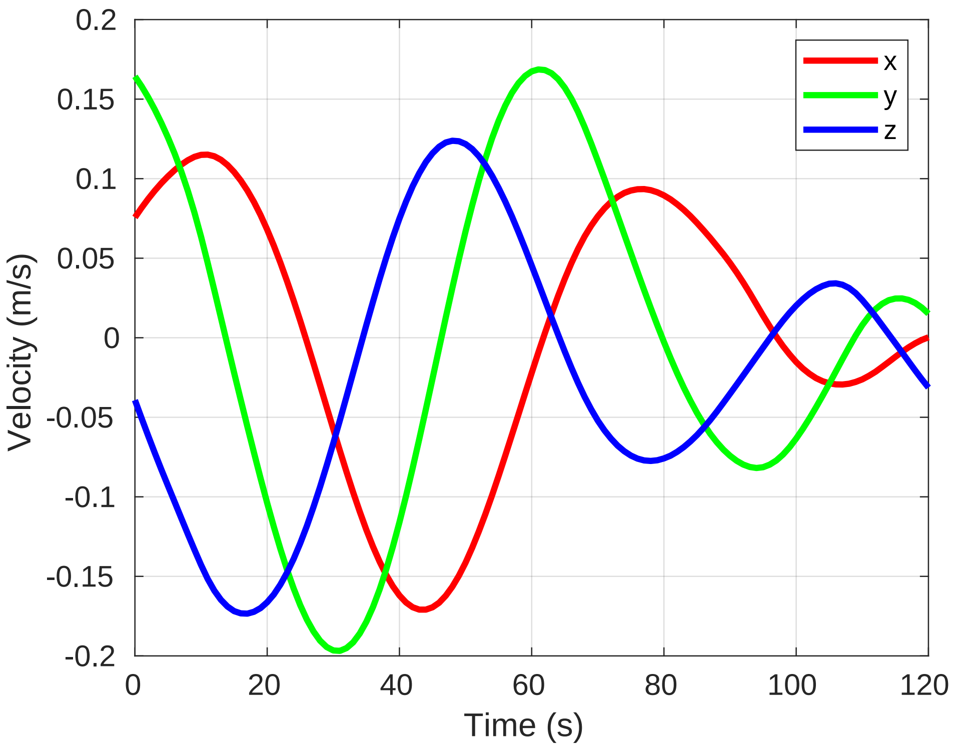
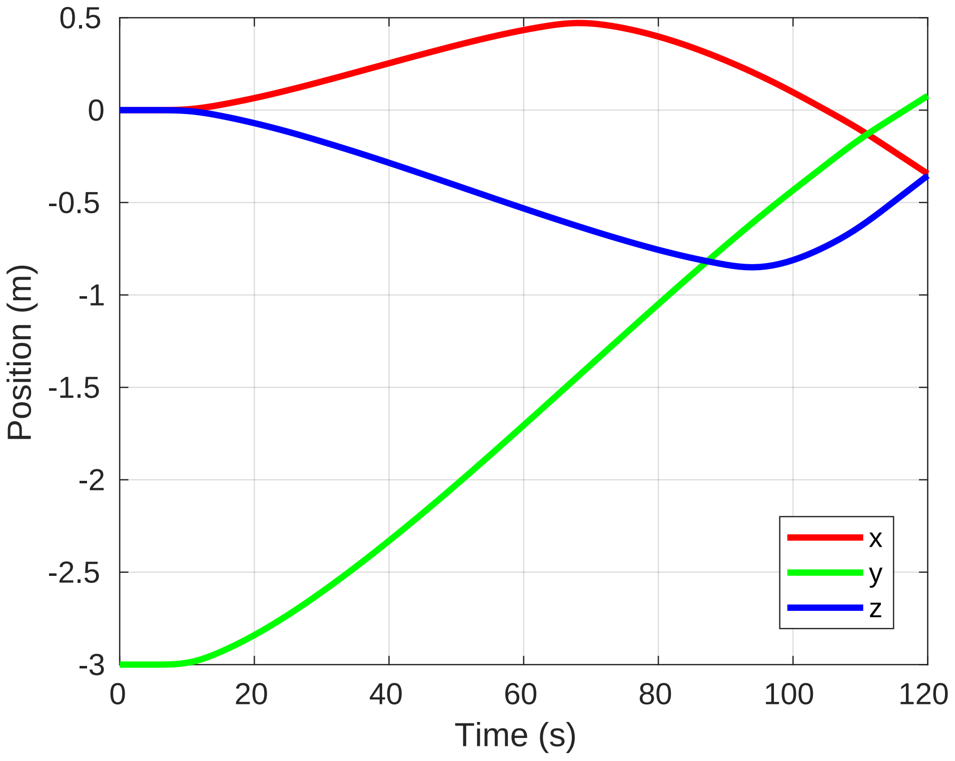


Figure 3-7: The nominal trajectory in the target body frame produced by the motion planner [58] for the initial conditions given in Table 3.1.



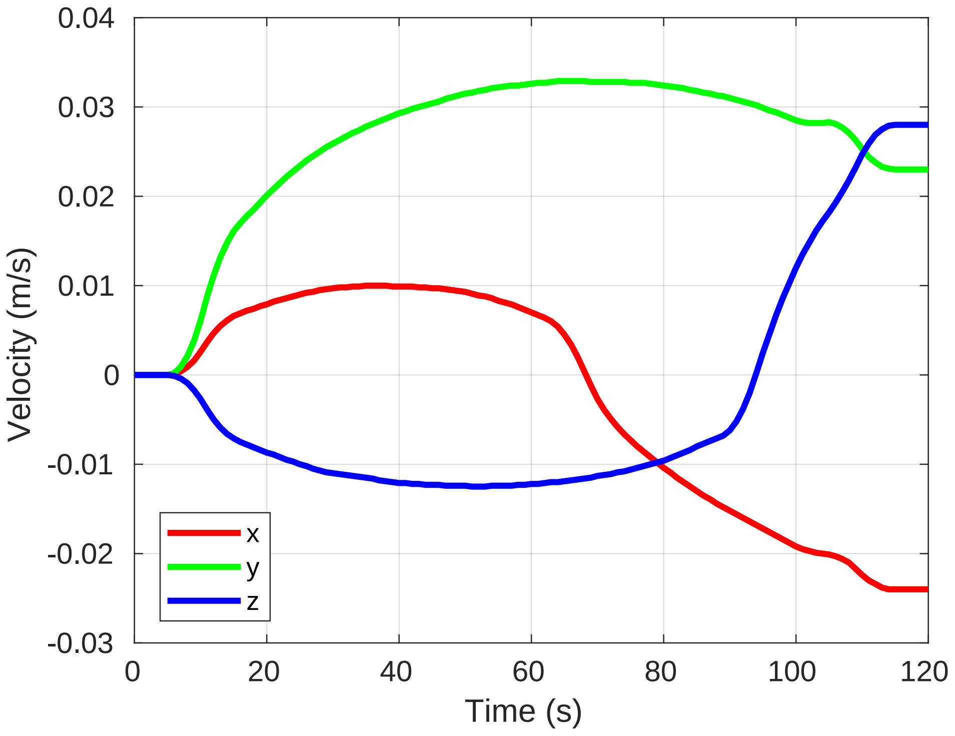


Figure 3-8: The nominal trajectory produced by the motion planner [58], rotated into the inertial frame using the predicted target's motion, for the initial conditions given in Table 3.1.

where $\widetilde{\mathbf{J}}_T$ is explicitly written as

$$\widetilde{\mathbf{J}}_T = \begin{bmatrix} \widetilde{J}_{xx} & \widetilde{J}_{xy} & \widetilde{J}_{xz} \\ \widetilde{J}_{xy} & \widetilde{J}_{yy} & \widetilde{J}_{yz} \\ \widetilde{J}_{xz} & \widetilde{J}_{yz} & \widetilde{J}_{zz} \end{bmatrix} . \tag{3.15}$$

Each error term $\widetilde{J}_{xx}, ...$ can be explicitly chosen for a particular trajectory to analyze, or can be randomly sampled to generate a wide variety of test cases.

However, once $\mathbf{J}_T$ has been initially determined, it must be checked to see if it satisfies the physically realistic inertia tensor constraints:

$$J_{xx} + J_{yy} > J_{zz} \tag{3.16a}$$

$$J_{xx} + J_{zz} > J_{yy} \tag{3.16b}$$

$$J_{yy} + J_{zz} > J_{xx}. \tag{3.16c}$$

If the real inertia tensor does not meet these constraints, the error tensor $\widetilde{\mathbf{J}}_T$ is re-calculated.

3.3 Controller Implementation Details

This section outlines how the online, tube-based MPC algorithm developed in Chapter 2 is specifically applied and tuned for intercepting a tumbling object. This includes specifics on the MPC problem formulation, exogenous input estimation/tube updates, and exogenous input prediction.

3.3.1 MPC Formulation

The online, tube-based MPC controller utilizes the double-integrator dynamics (3.6) for the nominal model of the chaser's dynamics. This system is discretized using a sample time $T_s = 1$ second. The sample time is chosen to realistically ensure that the MPC and tube-update optimization problems can reliably be solved within a full

Table 3.2: Chosen parameters for the nominal MPC problem in intercepting a tumbling target.

Parameter	Value
N	30
$\mathbf{Q}$	$10\mathbf{I}_n$
$\mathbf{R}$	$10^4\mathbf{I}_m$

control loop on hardware. In practice, the online, tube-based MPC would likely be implemented in a cascading framework on hardware with a simpler controller (e.g., proportional-derivative control) running at a faster rate.

The nominal control input is computed via the MPC optimization problem defined in (2.18). The $\mathbf{Q}$ and $\mathbf{R}$ costs are tuned appropriately, after which the $\mathbf{P}$ terminal cost is obtained from the algebraic Riccati equation:

$$\mathbf{P} = \mathbf{A}^\top\mathbf{P}\mathbf{A} - (\mathbf{A}^T\mathbf{P}\mathbf{B})(\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B})^{-1}(\mathbf{B}^\top\mathbf{P}\mathbf{A}) + \mathbf{Q}. \tag{3.17}$$

The prediction horizon N is chosen to balance controller performance and computational simplicity. The relevant MPC parameter values are shared in Table 3.2. The MOSEK optimization software [5] is used to solve this QCQP. After the nominal control input is computed, the final control input applied to the system is computed via (2.19).

3.3.2 Exogenous Input Estimation and Tube Updates

After the control input is applied and the full chaser-target system takes a dynamic step, the algorithm proceeds to estimate the exogenous input and its covariance, which in turn leads to an updated tube for robustness. The exogenous input at time-step k is measured via (3.12). The estimation procedure provided by (2.21) and (2.22) is then applied. The forgetting factor λ is chosen to balance the accuracy/stability of the estimates with the ability to adapt to the true exogenous input dynamics. The initial values of the estimate $\widehat{\mathbf{w}}(0)$ and its covariance $\widehat{\boldsymbol{\Sigma}}(0)$ are set as design parameters.

The covariance estimate is then used along with the probability parameter $p_{\mathbf{w}}$ in

Table 3.3: Chosen parameters for the exogenous input estimation and tube updates in intercepting a tumbling target.

Parameter	Value
λ	0.8
p	0.99
α	0.5
β	0.75
u_p	0.1

(2.23b) to update the set of possible exogenous input values $\widehat{\mathcal{W}}(k+1)$. Care must be taken in setting the probability value $p_{\mathbf{w}}$; values that are too low will significantly reduce the robustness of the controller, while values that are exceedingly high will produce very large $\mathcal{W}$ sets that can render the problem of finding a tube infeasible. In this work, $p_{\mathbf{w}}$ is set to 0.99, meaning that the estimated exogenous input set $\widehat{\mathcal{W}}$ has a 99% chance of containing the true exogenous input.

Using $\widehat{\mathcal{W}}(k+1)$, the updated tube $\mathcal{Z}(k+1)$ and rejection gain $\mathbf{K}_r(k+1)$ are obtained via (2.10). This SDP problem is solved via the MOSEK software suite and the YALMIP modeling toolbox [35]. The values of α and β are tuned to balance acceptable trajectory performance while reliably ensuring there is a valid tube solution. Generally speaking, α values close to 0.5 and β values greater than 0.75 work in most cases. It will be an aspect of future work to more precisely define α and β values that work across different systems. For the tumbling object intercept problem, the maximum available control input in each axis is 0.075 m/s^2; therefore u_p is set to 0.075. Table 3.3 shares the relevant parameters and their values for estimating the exogenous input and updating the tube.

3.3.3 Exogenous Input Prediction

Since the exogenous inputs arise from errors in estimating the target's inertia tensor, they are in a sense *deterministic but unknown*. Therefore, the dynamic patterns of the $\mathbf{w}_C^W$ as the target rotates throughout a trajectory are smooth and generally noise-free. This enables the use of exogenous input prediction as part of the online, tube-

Table 3.4: Chosen parameters for the exogenous input prediction in intercepting a tumbling target.

Parameter	Value
n_a	2
n_c	1
λ_A	0.875
$\boldsymbol{\theta}_A$	[0,0,0]
$\mathbf{P}_A$	$10^6\mathbf{I}_{n_a+n_c}$
warm-up period	60 s

based MPC framework. However, the ARMA-based prediction framework detailed in Section 2.3.3 is still challenging as the dynamic model $\mathbf{A}_{\mathbf{w}}^*$ must be identified purely from collected data during the trajectory.

An ARMA model (2.25) is identified for each of the six components of $\mathbf{w}_C^W$ (three position, three velocity). The ARMA model orders are set as $n_a = 2$ and $n_c = 1$. These values were found to exhibit relatively effective identification capabilities while reducing model complexity (which in turn reduces the number of samples needed for successful convergence of the parameters). The dynamic model $\mathbf{A}_{\mathbf{w}}^*$ is then formulated as in (2.28).

Parameter updates are carried out using the measured exogenous inputs and the recursive least-squares algorithm detailed in (2.29) and (2.30). The parameters λ_A, $\boldsymbol{\theta}_A(0)$, $\mathbf{P}_A(0)$ were tuned to provide acceptable prediction performance for the specific problem at hand. Finally, a warm-up period is implemented, which prevents exogenous input predictions from being used in the nominal MPC problem (2.18) until the ARMA models have converged. Table 3.4 summarizes the parameters and their selected values for the task of predicting each component of the exogenous input vector.

3.3.4 Full Controller Framework

Having defined each component of the online, tube-based MPC algorithm applied to intercepting a tumbling object, a typical control loop can be summarized by the following five steps:

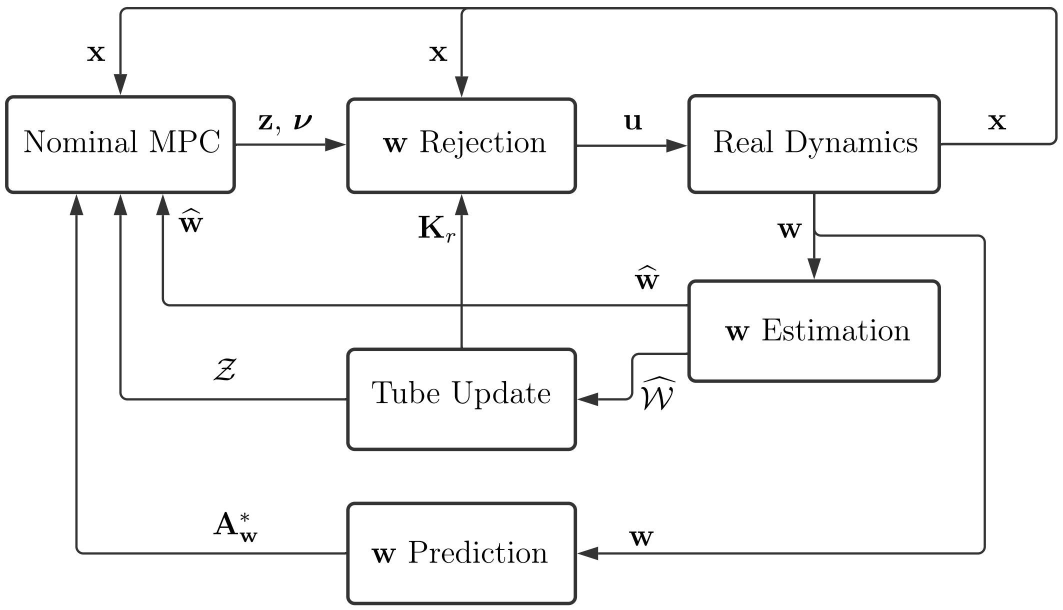


Figure 3-9: A block diagram of the full online, tube-based MPC for providing robust control to intercept a tumbling object.

1. Compute the robust control input.

2. After a step of the dynamic system, characterize the exogenous input via (3.12).

3. Estimate the next exogenous input and update the set of exogenous inputs.

4. Update the robust tube set.

5. Update the exogenous input prediction model.

Figure 3-9 provides a block diagram of the full controller. Using the parameters defined in Tables 3.2-3.4, this control loop runs until the trajectory is completed. A trajectory is successful if the offset point of 0.5 meters in the x-axis is reached with near-zero velocity in the T frame. A trajectory is unsuccessful if constraints are violated or a time-limit is reached (180 seconds).

3.4 Results and Discussion

This section analyzes several test cases where the online, tube-based MPC is applied to the tumbling intercept problem. Each test case is uniquely defined by the selection of the inertia tensor error $\widetilde{\mathbf{J}}$ as well as the initial estimate of the exogenous input $\widehat{\mathbf{w}}(0)$ and its covariance $\widehat{\boldsymbol{\Sigma}}_{\mathbf{w}}(0)$. The covariance directly influences the initial tube set $\mathcal{Z}(0)$. For each test case, a standard, tube-based MPC is also applied to the problem in order to provide performance comparisons. The online and standard controllers both use the same initial values of $\widehat{\mathbf{w}}(0)$ and $\widehat{\boldsymbol{\Sigma}}_{\mathbf{w}}(0)$.

The first two test cases present scenarios where the initially estimated exogenous input set $\widehat{\mathcal{W}}(0)$ is a), over-conservative when compared to the true set, and b), small enough when compared to the true set that the tube-based MPC would not be robust. Then, a Monte Carlo test of 100 different scenarios with randomly selected values of $\widetilde{\mathbf{J}}$ is performed to provide a more comprehensive performance comparison between the online and standard tube-based MPC algorithms.

3.4.1 Case 1: $\widehat{\mathcal{W}}(0)$ is over-conservative

This test case represents the scenario where the initially chosen set of exogenous inputs is over-conservative when compared to the true range of possible values. The test case-specific parameters are given as

$$\widetilde{\mathbf{J}} = \begin{bmatrix} 851.17 & -19.86 & -108.57 \\ -19.86 & -6241.3 & -17.21 \\ 108.57 & -17.21 & 6455.6 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{3.18a}$$

$$\widehat{\mathbf{w}}(0) = \mathbf{0} \tag{3.18b}$$

$$\widehat{\boldsymbol{\Sigma}}_{\mathbf{w}}(0) = (2.5 \times 10^{-5})\ \mathbf{I}_n. \tag{3.18c}$$

Notice that the components of the error inertia tensor represent a 5% estimation error when compared to Envisat's nominal inertia tensor defined in (3.13).

Both the online and standard tube-based controllers successfully completed the

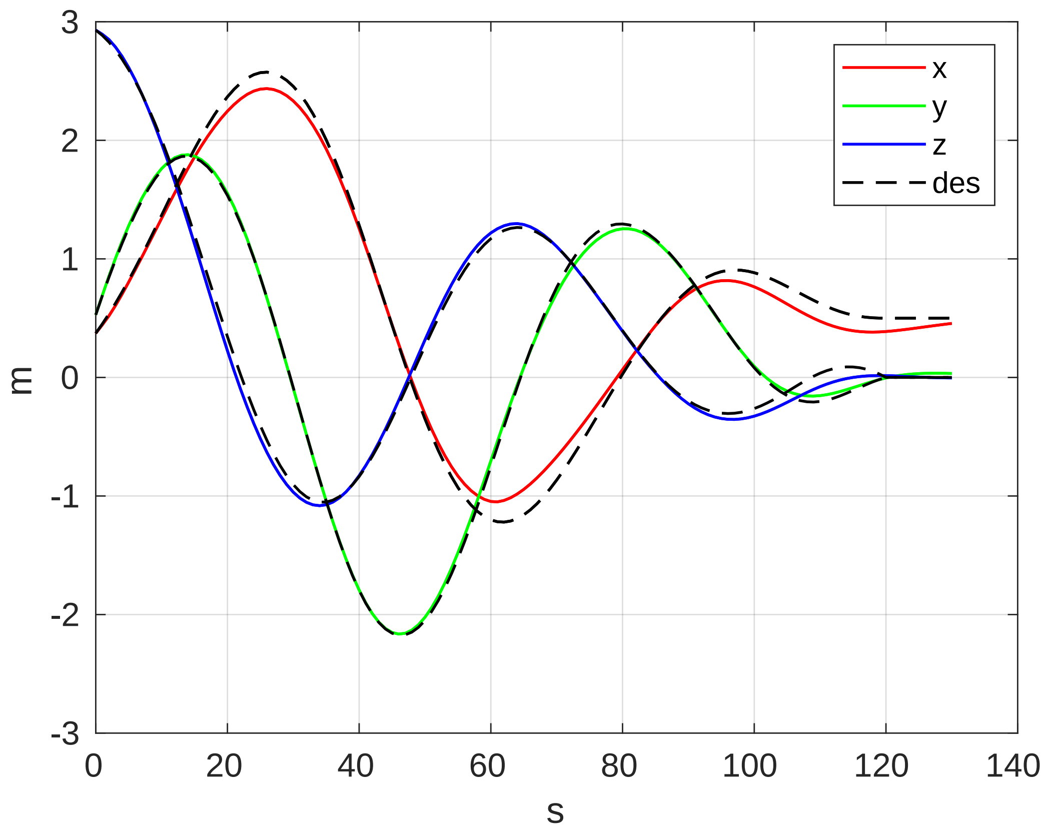


Figure 3-10: The position trajectory in the target body frame produced by the online, tube-based controller for test case 1 (compared to the desired nominal trajectory produced by the motion planner).

intercept trajectories without violating constraints. Figure 3-10 shows the resulting trajectory (position) produced by the online, tube-based MPC in the T frame. The online, tube-based MPC provided better overall performance when compared to the standard, tube-based MPC. Performance was measured by using the following cost function that closely matches the MPC objective function:

$$J = \sum_{k=0}^{t_f} \left|\left|\mathbf{x}(k) - \bar{\mathbf{x}}(k)\right|\right|_{\mathbf{Q}}^2 + \left|\left|\mathbf{u}(k)\right|\right|_{\mathbf{R}}^2. \tag{3.19}$$

Table 3.5 shares the cost function and trajectory time statistics for the standard and online tube-based controllers.

The online, tube-based MPC performs better largely due its shrinking of the tube based on measured exogenous input data and ability to predict exogenous input val-

Table 3.5: Case 1 statistics for online and standard tube-based MPC.

Statistic	Online	Standard
Trajectory time length (s)	130	119
J (cost)	66.98	78.49

ues. This allows the originally shrunk constraints to be somewhat more relaxed. Meanwhile, the standard, tube-based MPC is stuck with the originally shrunk constraints, and thus is more limited in its tracking performance of the desired trajectory. Figure 3-11 shows the state and control input components of the trajectory in the inertial frame. Figure 3-12 depicts the online estimation of the exogenous input (which also implicitly shows the shrinking of the tube), while Figure 3-13 shows the exogenous input predictions using the identified ARMA model.

In Figure 3-11, notice that the standard tube-based MPC is limited by the tightened constraints (visibly evident in the z-axis velocity component). Furthermore, there are more spikes in the control inputs for the standard tube-MPC due to the greater exogenous input rejection gain. Since the online, tube-based MPC adapts to the tube size, these unwanted performance characteristics are largely removed from the trajectory. Figure 3-12 shows that the online, tube-based MPC quickly adapts its estimates of the exogenous input and its associated bounds arising from the estimated covariance. The 3-σ bounds shown in the graph for each component are proportional to the tube size in that particular component. Figure 3-13 shows open-loop predictions produced by the identified ARMA model over the MPC horizon at select time-steps (after the warm-up period has passed). Notice that the initial values of the open-loop predictions are the current estimate $\widehat{\mathbf{w}}(k)$. While there is certainly prediction error, the patterns largely follow the real exogenous input data and still help to improve controller performance. Future work will focus on better tuning and model structures to better predict exogenous inputs. There is also a corresponding spike in control inputs when the exogenous input prediction is activated in the MPC problem after the warm-up period. While this is somewhat expected since the model used in (2.18b) is suddenly altered, future work will focus on providing a smoother

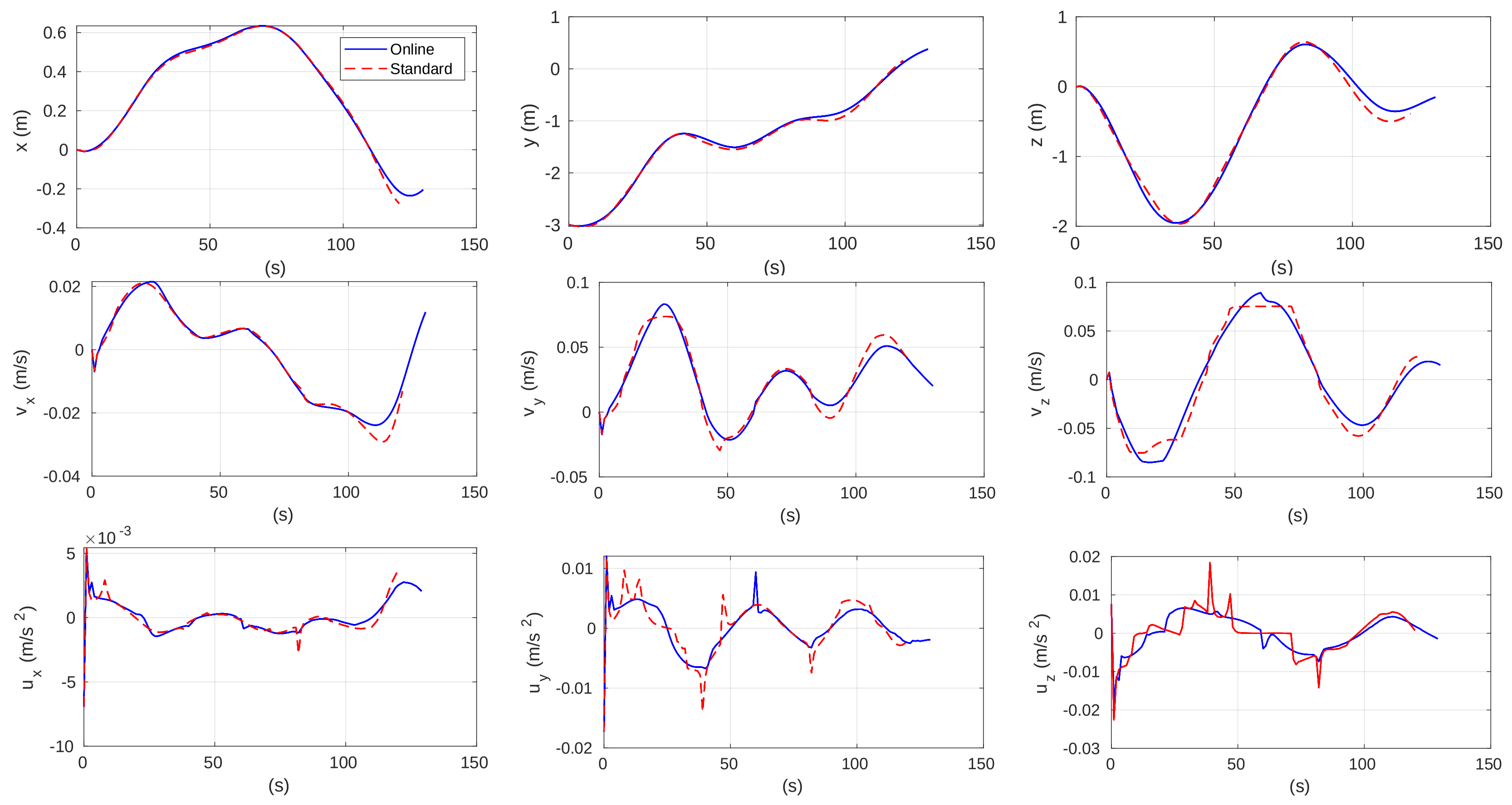


Figure 3-11: State and control input history in the inertial frame for the online and standard, tube-based controllers in test case 1.

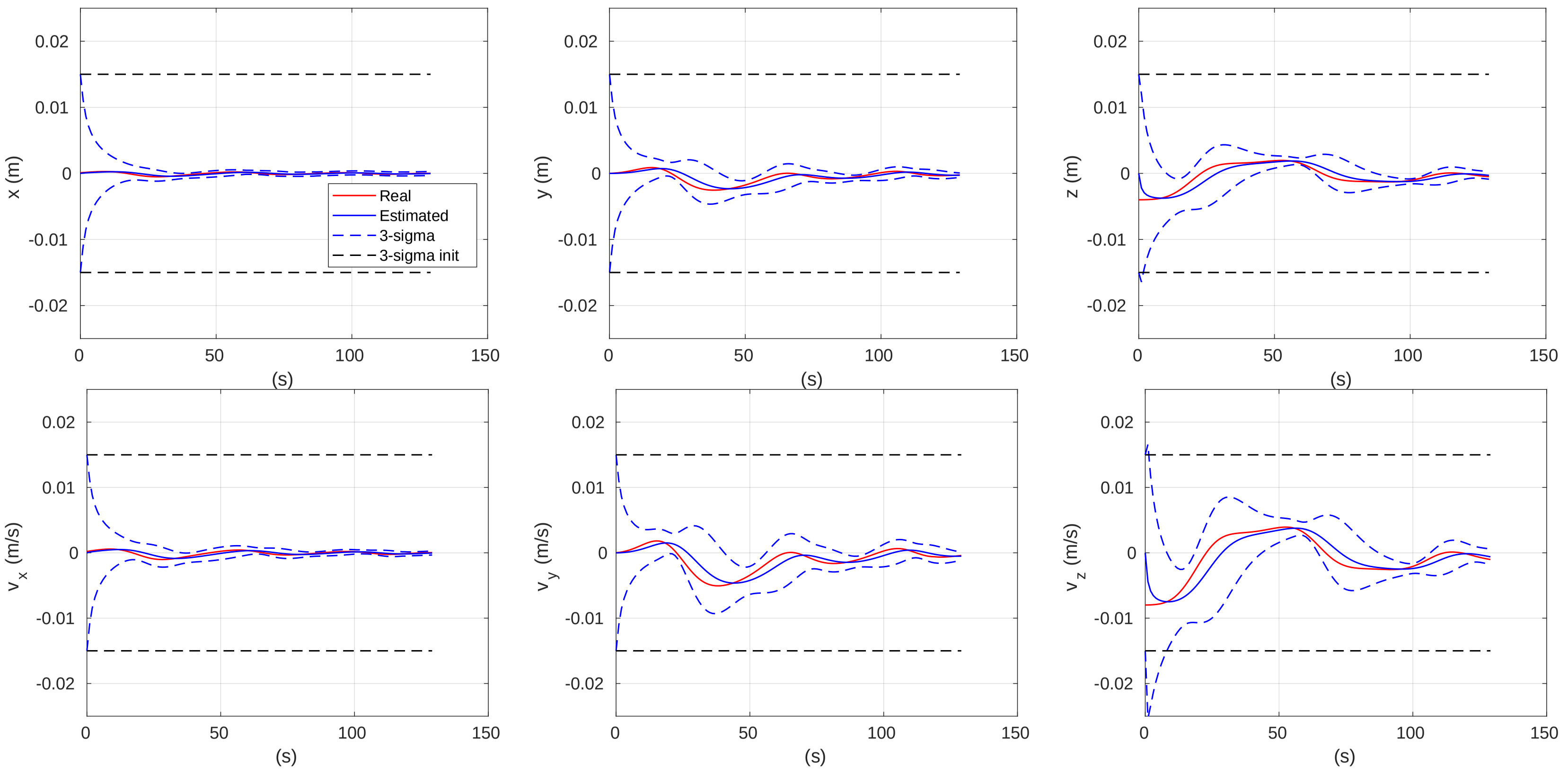


Figure 3-12: Estimated values of the exogenous input. The 3-σ bounds of the estimate (provided by the covariance) help visualize the approximated set of exogenous inputs $\widehat{\mathcal{W}}(k)$.

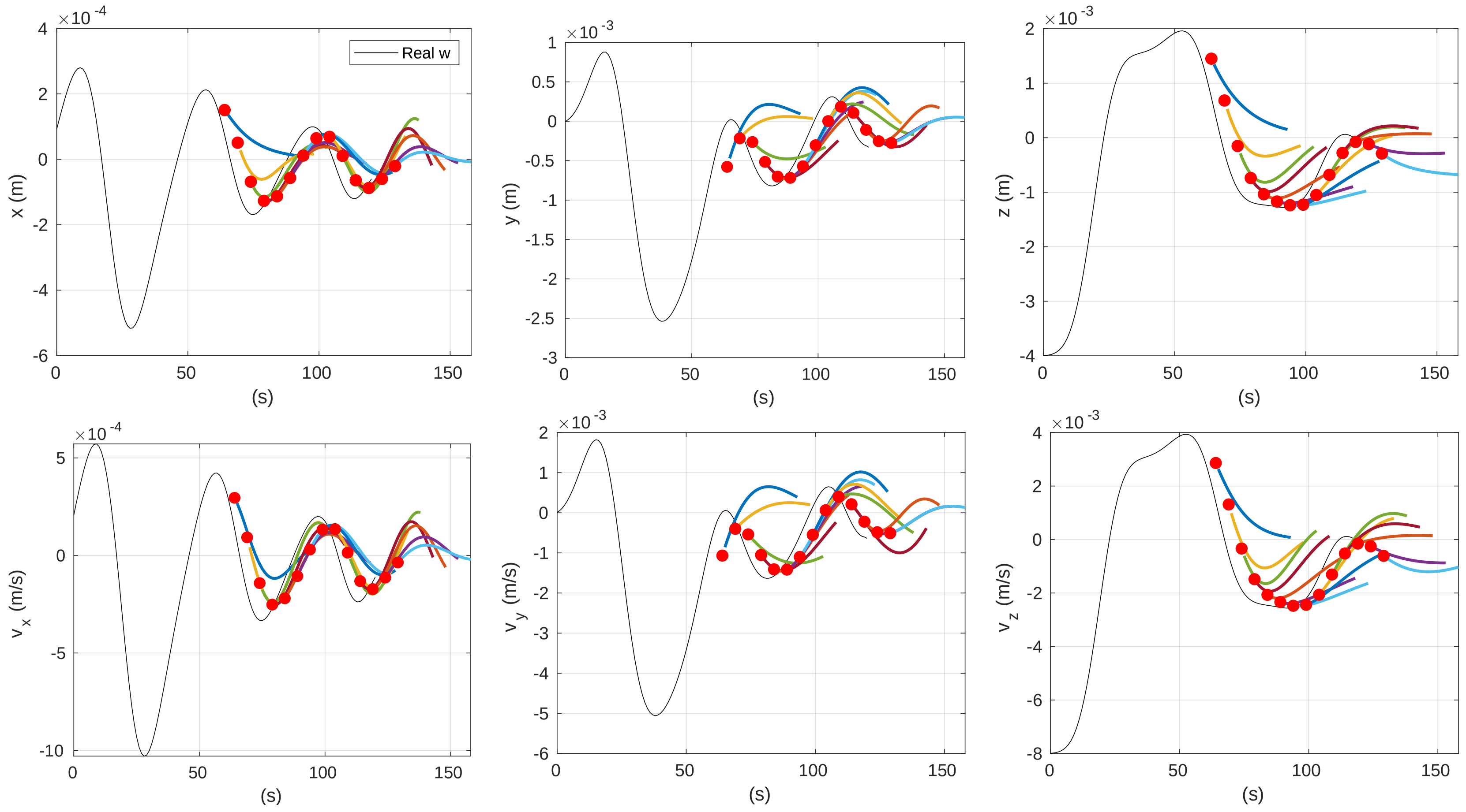


Figure 3-13: Open-loop predictions of the exogenous input throughout the MPC horizon using the identified ARMA model in test case 1. The predictions are denoted in color at select intervals while the real exogenous input is in black.

Table 3.6: Case 2 statistics for online and standard tube-based MPC.

Statistic	Online	Standard
Trajectory time length (s)	139	137
$\sum_{k=0}^{t_f} V(k)$ (cost)	49.22	34.32

transition when activating the exogenous input prediction in the MPC optimization.

3.4.2 Case 2: $\widehat{\mathcal{W}}(0)$ is too small to be robust

Another test case is examined where the initial estimated set of exogenous inputs $\widehat{\mathcal{W}}(0)$ is too small when compared to the true set. The test-specific parameters are given as

$$\widetilde{\mathbf{J}} = \begin{bmatrix} 3404.7 & 39.71 & -434.28 \\ 39.71 & -12482.6 & -172.1 \\ -434.28 & -172.1 & 0 \end{bmatrix} \mathrm{kg} \cdot \mathrm{m}^2 \tag{3.20a}$$

$$\widehat{\mathbf{w}}(0) = \mathbf{0} \tag{3.20b}$$

$$\widehat{\boldsymbol{\Sigma}}_{\mathbf{w}}(0) = \left(5.625 \times 10^{-7}\right) \mathbf{I}_n \ . \tag{3.20c}$$

Once again, both the online and standard tube-based controllers successfully completed the trajectory. In this case, the standard, tube-based MPC outperforms the online, tube-based MPC (Table 3.6). However, the standard, tube-based MPC did not have robust properties for a significant portion of the trajectory, as the initialized tube based on $\widehat{\mathcal{W}}(0)$ was too small. This demonstrates its flaw in requiring accurate initial values of $\widehat{\mathcal{W}}(0)$ in order to retain robust properties.

Conversely, the online, tube-based MPC quickly grows the estimate of $\widehat{\mathcal{W}}$ based on the measured exogenous input data. This allows it to recover the robust properties despite the error in $\widehat{\mathcal{W}}(0)$, demonstrating its flexibility when there is little prior knowledge on the characteristics of the exogenous input set. Figure 3-14 shows the exogenous input estimates in the z-axis velocity component. Notice how the initial

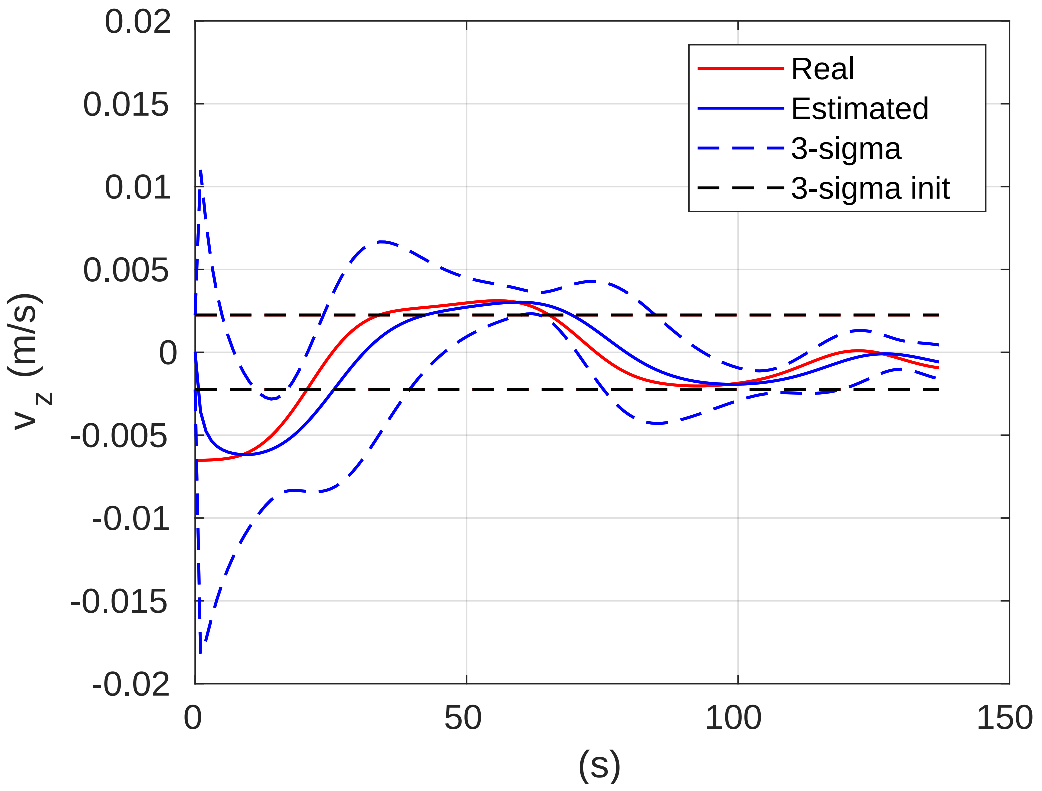


Figure 3-14: The online estimates of the z-axis velocity component of the exogenous input in test case 2. Notice that the online, tube-based MPC grows the bounds on $\mathbf{w}$ accordingly to retain robust properties.

3-σ bounds are not sufficient to capture the exogenous input, but the updated 3-σ bounds provided by online, tube-based MPC work well.

3.4.3 Monte Carlo Performance Analysis

A Monte Carlo test of 100 different trajectories was performed to provide a more comprehensive analysis on the performance differences between online and standard tube-based MPC. For each trajectory, the error inertia tensor was created by randomly perturbing the nominal inertia tensor. The perturbation levels were drawn from the uniform distribution with 10% error bounds. The initial exogenous input estimate/covariance values were the same across each trajectory:

$$\widehat{\mathbf{w}}(0) = \mathbf{0} \tag{3.21a}$$

$$\widehat{\mathbf{\Sigma}}_{\mathbf{w}}(0) = \left(2.5 \times 10^{-5}\right) \mathbf{I}_n. \tag{3.21b}$$

Table 3.7: Monte Carlo test statistics for online and standard tube-based MPC.

Statistic	Online	Standard
Mean trajectory time length (s)	130.57	120.89
Mean J (cost)	54.07	85.23
Mean mRPI comp. time (s)	0.0101	-
Mean MPC comp. time (s)	0.0213	-

Table 3.7 shares key cost and trajectory time statistics from the Monte Carlo trajectories. The online tube-based MPC considerably outperforms the standard tube-based MPC across the 100 trials. Additionally, the online tube-based MPC has the unique advantage of recovering robust properties in the case where the initialized set $\widehat{\mathcal{W}}(0)$ is unsuitable. The solving times for the mRPI approximation and the MPC optimization are also reasonably short; this points to the practical feasibility of online tube-based MPC for future implementations on resource-limited spacecraft processors.

Although online tube-based MPC tends to work better than standard tube-based MPC in the tumbling intercept problem with uncertainty arising from errors in estimating the target's inertia tensor, this does not always mean it should be the algorithm of choice for any application requiring robust control. If the nature of the exogenous input is largely aleatoric with no discernible dynamic patterns, the attempts at dynamic prediction over the MPC horizon will fail. Additionally, while the exogenous input estimate $\widehat{\mathbf{w}}$ can still be successfully estimated, its inclusion into the nominal MPC problem can drive up performance costs due to its high variance between time-steps. Therefore, standard tube-based MPC is recommended if the exogenous input is aleatoric and there is a somewhat reasonable guess for its bounds.

However, in many cases related to autonomous spacecraft proximity operations, the main factors contributing to exogenous inputs are from epistemic uncertainty sources. Examples other than inertia tensor uncertainty include solar radiation pressure disturbances that are dependent on orbit location and incident angles, unmodeled asteroid gravitational terms arising from shape model errors and flexible manipulator dynamics upon grappling an object. Future work on the online, tube-based MPC algorithm will focus on nonlinear formulations and the use of more advanced function approximators to characterize the exogenous input, which in turn will enable it to address these other types of epistemic uncertainty.

Chapter 4

Reinforcement Learning for Six-Degree-of-Freedom Docking

In this section, RL is proposed as a solution framework for 6-DOF docking control with rotating targets. RL involves learning a policy that maps observations to actions in order to maximize a reward signal given by the environment. Since it is a general, model-free framework, RL is potentially advantageous over model-based methods for scenarios where model identification is infeasible or prohibitive; the environment in RL is a black box during the learning process and the resulting policy is solely dependent on the experienced states, actions, and rewards. Moreover, once learned, implementation of the policy requires low amounts of computational effort and memory, making it practically realizable with current spacecraft computing resources.

The source of uncertainty in this scenario is in the initial conditions of the chaser and target spacecraft. If the initial conditions were precisely known, standard trajectory optimization techniques could easily be applied; however, trajectory re-planning (which is computationally expensive in the 6-DOF case) would need to occur if the initial conditions varied significantly from the expected values. As such, RL is implemented here to provide feedback control that accounts for a range of initial conditions while also abiding by collision constraints and successfully docking in the nonlinear, 6-DOF environment. The strength of learning here is thus to generalize the correct docking policy over the range of initial conditions and maintain computational efficiency.

This chapter begins with the required theoretical background on proximal policy optimization (PPO), which is the specific reinforcement learning algorithm used. Then, its adaptation for the 6-DOF problem of docking with rotating targets is presented along with the specific dynamics and constraints of the scenario. Finally, results obtained by applying a developed policy to the simulated Apollo transposition and docking maneuver are shared and discussed. The results also include a comparison with standard optimal control methodology using the GPOPS-II software suite [47]. The main contributions are two-fold: first, to present an RL-based framework for 6-DOF docking policies with rotating and non-rotating target spacecraft, and, second, to provide methods, results, and insight that will benefit future research in learning-based methods for spacecraft proximity operations [42], [43].

4.1 Reinforcement Learning Theory

This section outlines the necessary theoretical background for the specific implementation of PPO to the 6-DOF docking problem.

4.1.1 Conceptual Overview

RL is a subdivision of machine learning where an agent learns a policy that maps observations to actions in order to maximize a numerical reward across experienced trajectories [60]. The agent learns a policy by repeatedly interacting with the environment over numerous trajectories (termed "episodes"), either real or simulated, and receiving rewards based on the action taken at each time step (Figure 4-1). RL is modeled as a Markov decision process that includes a state space $\mathcal{S}$, action space $\mathcal{A}$, state transition distribution $P\Big(\mathbf{x}(k+1)\big|\big(\mathbf{x}(k),\mathbf{u}(k)\big)\Big)$, and reward function $r\big(\mathbf{x}(k),\mathbf{u}(k)\big)$, where state $\mathbf{x}\in\mathcal{S}$, action (control input) $\mathbf{u}\in\mathcal{A}$, and k is the discrete time-step index. During the learning process, the policy $\pi_{\boldsymbol{\theta}}=\big(\mathbf{u}(k)|\mathbf{x}(k)\big)$ is formalized as a conditional probability distribution, dependent upon the parameter vector $\boldsymbol{\theta}$, mapping states to actions.

One episode results in a trajectory of state-action pairs, denoted as

$$\boldsymbol{\tau} = \big[\mathbf{x}(0), \mathbf{u}(0), ..., \ \mathbf{x}(t_f), \mathbf{u}(t_f)\big] \in \mathbb{T} \tag{4.1}$$

with t_f being the number of time steps in the trajectory and $\mathbb{T}$ being the set of all possible state-action pair trajectories. Rewards received at successive time-steps are discounted to accommodate infinite-horizon problems. The sum of discounted rewards over the trajectory is

$$r(\boldsymbol{\tau}) = \sum_{k=0}^{t_f} \gamma^k r\big(\mathbf{x}(k), \mathbf{u}(k)\big), \tag{4.2}$$

where $\gamma \in (0, 1)$ is the discount factor. The goal of RL is to maximize the expectation of discounted rewards across all trajectories experienced by the agent:

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\left[r(\boldsymbol{\tau})\right] = \int_{\mathbb{T}} r(\boldsymbol{\tau}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) d\boldsymbol{\tau} \quad . \tag{4.3}$$

The probability of experiencing a particular trajectory based upon the policy's parameter vector $\boldsymbol{\theta}$ is

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \left[\prod_{k=0}^{t_f-1} p(\mathbf{x}(k+1)\big|\big(\mathbf{x}(k), \mathbf{u}(k)\big)\right] p\big(\mathbf{x}(0)\big), \tag{4.4}$$

where $\mathbf{u}(k)$ is sampled from $\pi_{\boldsymbol{\theta}}\big(\mathbf{u}(k)|\mathbf{x}(k)\big)$. Note that the state transition is stochas-

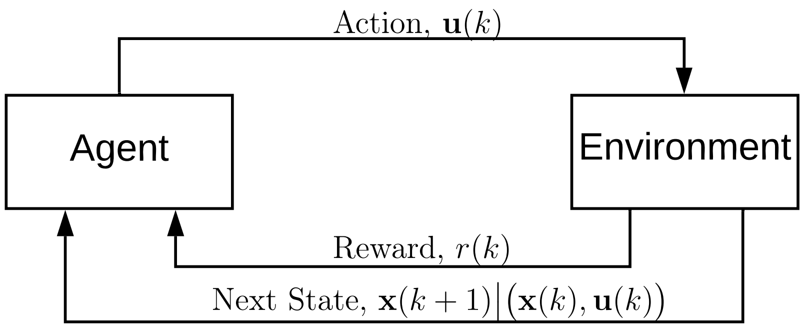

Action, $\mathbf{u}(k)$
Agent
Environment
Reward, $r(k)$
Next State, $\mathbf{x}(k+1)\big|\big(\mathbf{x}(k), \mathbf{u}(k)\big)$

Figure 4-1: The general schematic of reinforcement learning.

tic in the general case: this can be replaced by a deterministic state transition in certain applications. For example, the Apollo docking scenario presented in this work assumes nominal dynamics and thus a deterministic state transition is used. The variance of the policy's conditional distribution results in a stochastic action choice, enabling further exploration of the action space. As learning progresses, the variance is reduced to instead encourage more exploitation of the current policy. After the learning process is completed, the variance of the policy is set to zero, resulting in a deterministic action choice as the mean value of $\pi_{\boldsymbol{\theta}}\big(\mathbf{u}(k)|\mathbf{x}(k)\big)$, i.e., a deterministic feedback control law during implementation.

4.1.2 Proximal Policy Optimization

The specific RL algorithm used in this work is PPO [53]. PPO is a state-of-the-art policy learning algorithm with successful results in control tasks with continuous or discrete state/action spaces. PPO is a model-free, actor-critic algorithm where the policy that selects actions (the actor) and an advantage function that evaluates the selected actions (the critic) are learned concurrently. The state-value function $V^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k)\big)$ (with the parameter vector $\boldsymbol{\eta}$) is used in PPO and estimates the sum of future discounted rewards over the trajectory starting at the current state $\mathbf{x}(k)$ and following the current policy. However, this function is initially unknown and the parameter vector $\boldsymbol{\eta}$ must be learned concurrently with the policy parameter vector $\boldsymbol{\theta}$. The resulting advantage function $A^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k), \mathbf{u}(k)\big)$ is the difference between the empirical rewards received during the learning process and the state-value function's estimate.

$$V^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k)\big) = \mathbb{E}_{\pi}\left[\sum_{i=k}^{t_f} \gamma^{i-k} r\big(\mathbf{x}(i), \mathbf{u}(i)\big) \middle| \mathbf{x}(k)\right] \tag{4.5}$$

$$A^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k), \mathbf{u}(k)\big) = \left[\sum_{i=k}^{t_f} \gamma^{i-k} r\big(\mathbf{x}(i), \mathbf{u}(i)\big)\right] - V^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k)\big) \tag{4.6}$$

PPO is a descendant of the trust region policy optimization algorithm [52], retaining the ability to mitigate large policy updates (thus reducing the risk of learning divergence) while being simpler and more widely implementable. Central to PPO is the policy probability ratio

$$p_{\boldsymbol{\theta}}(k) = \frac{\pi_{\boldsymbol{\theta}}\big(\mathbf{u}(k)|\mathbf{x}(k)\big)}{\widehat{\pi}_{\boldsymbol{\theta}}\big(\mathbf{u}(k)|\mathbf{x}(k)\big)}, \tag{4.7}$$

which compares the probability $\pi_{\boldsymbol{\theta}}\big(\mathbf{u}(k)|\mathbf{x}(k)\big)$ of selecting a particular action *after* a learning update to the probability $\widehat{\pi}_{\boldsymbol{\theta}}\big(\mathbf{u}(k)|\mathbf{x}(k)\big)$ of selecting the same action *prior* to the update. The probability ratio is then directly used in the PPO objective function to be maximized:

$$J_{\boldsymbol{\theta}} = \mathbb{E}_{p(\boldsymbol{\tau})}\Bigg[\ \min\ \Big(p_{\boldsymbol{\theta}}(k) A^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k), \mathbf{u}(k)\big),\ \operatorname{clip}\big[p_{\boldsymbol{\theta}}(k), \xi\big] A^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k), \mathbf{u}(k)\big)\Big)\Bigg] \tag{4.8}$$

where the clip function, defined as

$$\operatorname{clip}\big[p_{\boldsymbol{\theta}}(k), \xi\big] = \begin{cases} 1-\xi & \text{if } p_{\boldsymbol{\theta}}(k) < 1-\xi \\ 1+\xi & \text{if } p_{\boldsymbol{\theta}}(k) > 1+\xi \\ p_{\boldsymbol{\theta}}(k) & \text{otherwise} \quad , \end{cases} \tag{4.9}$$

imposes bounds on the policy probability ratio using the clipping parameter $\xi \in (0,1)$. The clipping parameter controls how close the updated policy is to the old policy, effectively implementing a trust region and eliminating large, unwanted policy updates. Note that this objective function is measured relative to the policy prior to the update. Thus, the numerical value of the objective function over the course of many updates is uninformative. Instead, its immediate gradient is more critical in guiding the policy to maximize rewards over all trajectories.

To learn the state-value function, the commonly used mean squared error cost function is minimized:

$$L_{\boldsymbol{\eta}} = \frac{1}{2}\ \mathbb{E}_{p(\boldsymbol{\tau})}\left[\left(V^{\pi}_{\boldsymbol{\eta}}\big(\mathbf{x}(k)\big) - \sum_{i=k}^{t_f} \gamma^{i-k} r\big(\mathbf{x}(i), \mathbf{u}(i)\big)\right)^2\right]. \tag{4.10}$$

Essentially, the mean squared difference between the state-value function estimate and the actual sum of resulting rewards is minimized. The multiplication by $\frac{1}{2}$ simplifies the loss gradient calculation. There now exists an objective function to improve the policy in (4.8) and a cost function to correct errors in the state-value function in (4.10). Thus, the gradients of these functions can be used to perform gradient ascent on $\boldsymbol{\theta}$ and gradient descent on $\boldsymbol{\eta}$:

$$\boldsymbol{\theta}^{+} = \boldsymbol{\theta}^{-} + \beta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{-}} \tag{4.11a}$$

$$\boldsymbol{\eta}^{+} = \boldsymbol{\eta}^{-} - \beta_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\eta}} L_{\boldsymbol{\eta}}|_{\boldsymbol{\eta}=\mathbf{w}^{-}} \tag{4.11b}$$

where the scalars $\beta_{\boldsymbol{\theta}}$ and $\beta_{\boldsymbol{\eta}}$ are the policy learning rate and state-value function learning rate, respectively, which must be chosen by the designer.

4.2 Implementation for 6-DOF Docking

This section outlines how the theoretical PPO algorithm is formulated and applied to the 6-DOF docking problem with rotating targets.

4.2.1 Dynamics

The world frame W, the chaser body frame C, and the target body frame T are utilized in the 6-DOF docking scenario. The target's rotational motion, if non-zero, is assumed to be around the docking axis with a constant angular velocity $\boldsymbol{\omega}_T^T$. Moreover, the target's body frame origin is assumed to be co-located with the world frame. The state is defined as

$$\mathbf{x} = \left[\left(\mathbf{r}_C^W\right)^\top, \ \left(\mathbf{v}_C^W\right)^\top, \ \left(\mathbf{q}_C^W\right)^\top, \ \left(\boldsymbol{\omega}_C^C\right)^\top, \ \left(\mathbf{q}_T^W\right)^\top, \ \left(\boldsymbol{\omega}_T^T\right)^\top \right]^\top \tag{4.12}$$

with $\mathbf{r}_C^W \in \mathbb{R}^3$ as the position of the chaser in the world frame, $\mathbf{v}_C^W \in \mathbb{R}^3$ as the velocity of the chaser in the world frame, $\mathbf{q}_C^W \in \mathbb{R}^4$ as the attitude quaternion of the chaser with respect to world frame, $\boldsymbol{\omega}_C^C \in \mathbb{R}^3$ as the angular velocity of the chaser in its body frame, $\mathbf{q}_T^W \in \mathbb{R}^4$ as the attitude quaternion of the target with respect to the world frame, and $\boldsymbol{\omega}_T^T \in \mathbb{R}^3$ as the angular velocity of the target with respect to the world frame. The control action $\mathbf{u} = [\mathbf{F}^\top,\ \mathbf{L}^\top]^\top$ consists of a thrust command $\mathbf{F} \in \mathbb{R}^3$ and a torque command $\mathbf{L} \in \mathbb{R}^3$, both in the W frame. The thrust and torque commands are bounded by minimum/maximum actuator constraints. Control actions are commanded by the policy at discrete time intervals. The dynamics are derived below in continuous-time, and are subsequently discretized using a sample period of 1 second.

The translational dynamics are modeled using the double-integrator equations:

$$\dot{\mathbf{r}}_C^W = \mathbf{v}_C^W \tag{4.13a}$$

$$\dot{\mathbf{v}}_C^W = \frac{\mathbf{F}}{m} \tag{4.13b}$$

where m refers to the chaser mass. In practice, to compute individual thruster commands, the net force command must be determined in the chaser body frame. This is calculated using the chaser's current attitude, parameterized as a rotation matrix $\mathbf{R}_C^W \in \mathbb{R}^{3\times 3}$:

$$\mathbf{F}^C = \left(\mathbf{R}_C^W\right)^\top \mathbf{F}. \tag{4.14}$$

The translational state variables are defined with respect to the W frame since the target is spin-stabilized around the docking axis and thus has zero translational motion relative to the chaser.

The chaser's attitude dynamics are modeled using quaternion kinematics and Euler's equations for rigid bodies:

$$\dot{\mathbf{q}}_C^W = \frac{1}{2}\boldsymbol{\Omega}\big(\boldsymbol{\omega}_C^C\big)\mathbf{q}_C^W \tag{4.15a}$$

$$\dot{\boldsymbol{\omega}}_C^C = \mathbf{J}_C^{-1}\Big(\mathbf{L} - \boldsymbol{\omega}_C^C \times \mathbf{J}_C\boldsymbol{\omega}_C^C\Big) \tag{4.15b}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) \triangleq \begin{bmatrix} 0 & -\boldsymbol{\omega}^\top \\ \boldsymbol{\omega} & -\big[\boldsymbol{\omega}\big]_\times \end{bmatrix} \tag{4.16}$$

and $\mathbf{J}_C$ is the chaser's inertia tensor.

The target's attitude dynamics are also propagated via quaternion kinematics. Since the target's spin is assumed to be undisturbed around a constant axis, its attitude dynamics can be defined as

$$\dot{\mathbf{q}}_T^W = \frac{1}{2}\boldsymbol{\Omega}\big(\boldsymbol{\omega}_T^T\big)\mathbf{q}_T^W \tag{4.17a}$$

$$\dot{\boldsymbol{\omega}}_T^T = \mathbf{0}\ . \tag{4.17b}$$

Thus, (4.13), (4.15), and (4.17) govern the dynamics of the nonlinear, 6-DOF system. The relative attitude and angular velocity of the chaser with respect to the target are defined as

$$\mathbf{q}_C^T = \mathbf{q}_C^W \otimes \big(\mathbf{q}_T^W\big)^{-1} \tag{4.18a}$$

$$\boldsymbol{\omega}_C^T = \boldsymbol{\omega}_C^C - \boldsymbol{\omega}_T^T\ , \tag{4.18b}$$

where $\otimes$ refers to quaternion multiplication. As docking requirements are often formulated on the relative state of the chaser and target docking ports, it is useful to define the relative position $\mathbf{r}_p$ and velocity $\mathbf{v}_p$ of the chaser docking port with respect to the target docking port as

$$\mathbf{r}_p = \mathbf{r}_C^W + \mathbf{R}_C^W \mathbf{r}_{dc} - \mathbf{r}_{dt} \tag{4.19a}$$

$$\mathbf{v}_p = \mathbf{v}_C^W + \left[\boldsymbol{\omega}_C^C \times \mathbf{R}_C^W \mathbf{r}_{dc}\right]\ , \tag{4.19b}$$

where $\mathbf{r}_{dc}$ and $\mathbf{r}_{dt}$ refer to the chaser and target docking port positions in the C and T frames, respectively.

4.2.2 Policy Parameterization

The policy and state-value function are modeled through standard, feedforward neural networks. Thus, the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ represent the weights and biases of each network's respective layers. The neural network backpropagation algorithm and Adam optimizer [27] are used to perform gradient ascent/descent on the parameter vectors

Table 4.1: Neural network structural parameters.

	Policy Network		State-Value Function Network	
Layer	Neurons	Activation	Neurons	Activation
1st hidden	200	tanh	200	tanh
2nd hidden	110	tanh	32	tanh
3rd hidden	60	tanh	5	tanh
Output	6	linear	1	linear

$\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ according to (4.11). The policy is specifically a multivariate, Gaussian distribution with a diagonal covariance matrix. The neural network output consists of the resulting mean action based on the given state. The variance for each action is also learned, but is independent of the state. This essentially controls the degree of action space exploration throughout the learning process. In general, the agent learns to set large variance values during the learning process to encourage more exploration, and then diminish them in the later stages of learning to induce more exploitation of the current policy.

The inputs for both the policy and state-value function neural networks are scaled using a running mean and standard deviation of experienced state data while learning. This helps prevent the saturation of activation functions within each network layer. Similarly, neural network outputs are best defined when close to unity. As such, the policy network outputs are scaled accordingly so that an output of ± 1 corresponds to the maximum/minimum thrust or torque command. Table 4.1 shares the structure of network layers and their corresponding activation functions.

This work follows the example of Gaudet et al. [19] when implementing PPO in that we dynamically adjust learning parameters to target a desired Kullback-Leibler (KL) divergence value between successive policy updates [30]. This helps to prevent large policy updates that could derail the learning process, while yielding smoother policy updates. Over the course of learning, both the PPO clipping parameter ξ and the policy learning rate $\beta_{\boldsymbol{\theta}}$ are adjusted to keep the KL-divergence between updates as close as possible to the desired target value (KL_{des}).

4.2.3 Reward Function

Employing a valid reward function is critical to the success of PPO as the policy will learn to explicitly maximize this function. For 6-DOF docking maneuvers with rotating targets, the reward function consists of several terms that together account for minimizing state tracking errors and control effort, preventing collisions, and reinforcing successful docks. All terms are weighted relative to one another through design coefficients.

To account for translational state error, the term

$$\left[\dot{\mathbf{v}}_C^W(k) - \dot{\bar{\mathbf{v}}}_C^W(k)\right]^\top \mathbf{Q}_t \left[\dot{\mathbf{v}}_C^W(k) - \dot{\bar{\mathbf{v}}}_C^W(k)\right] \tag{4.20}$$

defines the quadratic weighted error between the acceleration produced by the RL agent via (4.13b) and a reference acceleration ($\dot{\bar{\mathbf{v}}}_t$) provided by a LQR feedback law:

$$\dot{\bar{\mathbf{v}}}_C^W(k) = -\mathbf{K}\mathbf{x}'(k) \tag{4.21}$$

where $\mathbf{K} \in \mathbb{R}^{3\times 6}$ is the LQR gain matrix and $\mathbf{x}'(k) \in \mathbb{R}^6$ is the translational part of the state vector (chaser position and velocity). The LQR design process is performed offline before the implementation of PPO. By adjusting the standard LQR performance and control cost matrices, the resulting gain matrix can be tuned to target a desired trajectory time length for nominal docking initial conditions. Additionally, the tuning process can account for a desired, non-zero final docking velocity by setting the LQR origin at an offset from the actual docking port location. The benefits of this reward term are two-fold. First, it provides a clear reward signal at all points in the translational state-space that guides the RL agent to achieve a successful docking trajectory. Second, it encourages the RL agent to produce docking trajectories with a specific time length (an important design consideration for many docking maneuvers). This is to alleviate the fact that, unlike most standard guidance and control techniques, the PPO algorithm does not afford the ability to enforce hard time constraints on the docking trajectory. Finally, $\mathbf{Q}_t \in \mathbb{R}^{3\times 3}, \mathbf{Q}_t \succeq \mathbf{0}$ where the weights in

$\mathbf{Q}_t$ are design parameters.

To account for errors between the actual and desired relative attitude and angular velocity, we define the reward function term $\widetilde{\boldsymbol{\alpha}}(k)^\top \mathbf{Q}_a \widetilde{\boldsymbol{\alpha}}(k)$. This term is based on the error quaternion $\widetilde{\mathbf{q}}_C^T(k)$ between the current and desired final relative attitude $\bar{\mathbf{q}}_C^T$:

$$\widetilde{\mathbf{q}}_C^T(k) = \bar{\mathbf{q}}_C^T \otimes \mathbf{q}_C^T(k)^{-1} \quad . \tag{4.22}$$

From Markley [39], the vector component of the error quaternion $\widetilde{\mathbf{q}}_v$ is doubled to measure attitude error. The relative angular velocity error, $\widetilde{\boldsymbol{\omega}}_C^T(k)$, between the current and desired relative angular velocity $\bar{\boldsymbol{\omega}}_C^T$ is also penalized. This results in a vector $\widetilde{\boldsymbol{\alpha}}(k) = \left[2\widetilde{\mathbf{q}}_v(k)^\top,\ \widetilde{\boldsymbol{\omega}}_C^T(k)^\top\right]^\top \in \mathbb{R}^6$ that penalizes both attitude and angular velocity errors. This term also has a weighting design matrix $\mathbf{Q}_a \in \mathbb{R}^{6\times6}, \mathbf{Q}_a \succeq \mathbf{0}$.

A quadratic control cost, collision penalty, and docking bonus are also included in the reward function. The quadratic control cost $\mathbf{u}(k)^\top \mathbf{R}\mathbf{u}(k)$ is applied to the force/torque command with a weighting design matrix $\mathbf{R} \in \mathbb{R}^{6\times6}, \mathbf{R} \succ \mathbf{0}$. The collision penalty $h \sin\left(\frac{\pi}{2}\frac{||\mathbf{r}_P||}{r_{\text{col}}}\right)$ is applied at each time-step the chaser docking port is within the boundary of the target spacecraft (modeled as a rectangular region). The penalty is scaled based upon the relative docking distance at the time of collision and the maximum possible distance for a collision (r_{col}). Together with the sine function and a weighting coefficient $h > 0$, this results in a smooth function that penalizes collisions with significant state error more heavily. The docking bonus $g(\mathbf{x}_t)$ is a discrete term applied if the agent achieves the docking requirements:

$$g\big(\mathbf{x}(k)\big) = \begin{cases} d & \text{if } \mathbf{x}(k) \text{ satisfies docking conditions} \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

where $d > 0$ is a weighting coefficient.

Two distinct reward functions are defined and are based on the aforementioned reward contributions:

$$r_1\big(\mathbf{x}(k),\mathbf{u}(k)\big) = -\left[\dot{\mathbf{v}}_C^W(k) - \dot{\mathbf{v}}_C^W(k)\right]^\top \mathbf{Q}_t \left[\dot{\mathbf{v}}_C^W(k) - \dot{\mathbf{v}}_C^W(k)\right] - \widetilde{\boldsymbol{\alpha}}(k)^\top \mathbf{Q}_a \widetilde{\boldsymbol{\alpha}}(k) - \mathbf{u}(k)^\top \mathbf{R}\mathbf{u}(k) - h\sin\left(\frac{\pi}{2}\frac{||\mathbf{r}_p||}{r_{\text{col}}}\right) \tag{4.24a}$$

$$r_2\big(\mathbf{x}(k)\big) = g\big(\mathbf{x}(k)\big) \tag{4.24b}$$

where r_1 represents the "shaping" penalties (LQR error, attitude/angular velocity error, control cost, and collision penalty) and r_2 represents the terminal docking bonus. Following the example of Gaudet et al. [19], a slightly smaller discount factor is used for the shaping penalties (γ_1) while a larger discount factor is used for the terminal docking bonus (γ_2). This results in the docking bonus being weighted more heavily in the long-term than the shaping penalties. Thus, the advantage function (4.6) and the state-value loss function (4.10) can be re-written as:

$$A_{\boldsymbol{\eta}}^\pi\big(\mathbf{x}(k),\mathbf{u}(k)\big) = \sum_{i=k}^{t_f}\left[\gamma_1^{i-k} r_1\big(\mathbf{x}(i),\mathbf{u}(i)\big) + \gamma_2^{i-k} r_2\big(\mathbf{x}(i)\big)\right] - V_{\boldsymbol{\eta}}^\pi\big(\mathbf{x}(k)\big) \tag{4.25}$$

$$L_{\boldsymbol{\eta}} = \frac{1}{2}\,\mathbb{E}_{p(\boldsymbol{\tau})}\left[\left(V_{\boldsymbol{\eta}}^\pi\big(\mathbf{x}(k)\big) - \sum_{i=k}^{t_f}\left[\gamma_1^{i-k} r_1\big(\mathbf{x}(k),\mathbf{u}(k)\big) + \gamma_2^{i-k} r_2\big(\mathbf{x}(k)\big)\right]\right)^2\right]. \tag{4.26}$$

The training episode either ends if the docking requirements are met or a trajectory time limit has been reached.

4.3 Experimental Setup

The above methodology was applied to the simulated Apollo transposition and docking maneuver. This maneuver involves the CSM re-orienting itself and docking with the lunar module (LM) in order to extract it from the third stage of the Saturn V rocket [2]. Historically, the LM did not have any rotational motion. However, in this particular simulation case, an initial angular velocity around the docking axis is induced in the LM to create a rotating target scenario. Figure 4-2 depicts the rele-

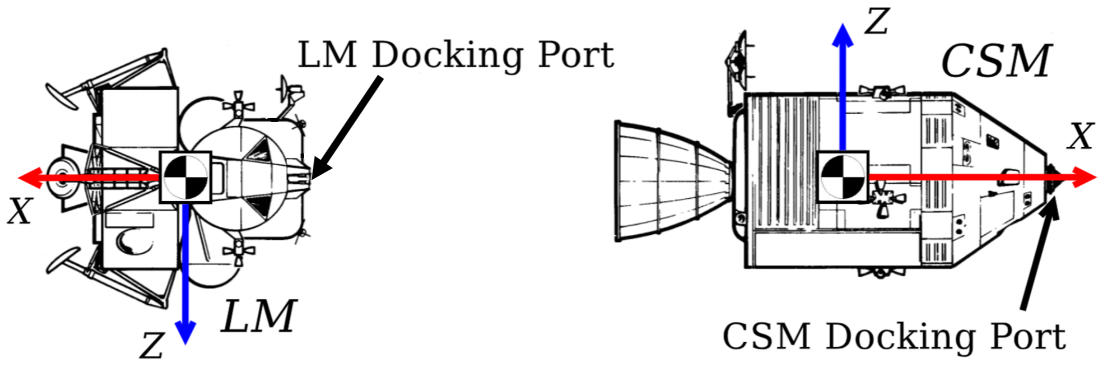


Figure 4-2: Overview of coordinate frames and initial configuration for the Apollo transposition and docking maneuver. The Y-axis is defined according to the right-hand rule. Reproduced with permission from the NASA History Division (Project Apollo Drawings and Technical Diagrams, publicly available).

vant coordinate frames: the LM frame represents the target body frame while the CSM frame represents the chaser's body frame. The docking axis is aligned with the x-axis of the LM frame. Note that Fig. 4-2 shows the LM and CSM in their initial configuration for the maneuver: the CSM frame has a 180° pitch rotation relative to the LM frame.

Actuator constraints are imposed on the minimum/maximum force and torque commands (Table 4.2). These are derived from approximating the maximum thrust and torque outputs achievable through the CSM's sixteen reaction control system thrusters [1]. Note that the policy produces thrust and torque commands within a continuous range of values; in reality, the CSM thrusters produce discrete, on/off thrust. A rectangular region is defined around the LM's center of mass to model collisions (Fig. 4-3). The region's y and z dimensions approximate the largest diameter of the LM and its surface intersects with the LM docking port. The collision geometry parameters, as well as the docking port positions of the CSM/LM in their respective spacecraft frames, are shown in Table 4.3. To perform a successful dock, the CSM docking port must meet the conditions outlined in Table 4.4. The angles $\left(\phi_C^T, \theta_C^T, \psi_C^T\right)$ represent the Euler angle representation of the relative attitude $\mathbf{q}_C^T$ [3]. Notice that the maneuver requires a strictly positive velocity and a $-60°$ relative roll in the x-axis to activate the docking mechanism.

For both training and testing episodes, the policy accepts the given state and

Table 4.2: Actuator minimum/maximum constraints.

Action Command	Value	Units
$\mathbf{F}$	$\pm$ [790.80, 790.80, 790.80]	N
$\mathbf{L}$	$\pm$ [2534.91, 2534.91, 2534.91]	N-m

Table 4.3: Collision geometry and docking port parameters.

Parameter	Value	Units
Collision box y-dim.	7	m
Collision box z-dim.	7	m
$\mathbf{r}_{dc}$	$[4.479, 0, 0]$	m
$\mathbf{r}_{dt}$	$[-3.250, 0, 0]$	m

Table 4.4: Conditions for successful docking.

Docking State Term	Goal	Acceptable Deviation	Units
$\mathbf{r}_p$	$\mathbf{0}$	$\pm$ 0.15	m
v_{px}	0.1	[0.05, 0.15]	m/s
$v_{py},\ v_{pz}$	0, 0	$\pm$ 0.1	m/s
$\phi_C^T,\ \theta_C^T,\ \psi_C^T$	$-60,\ 0,\ 0$	$\pm$ 5	deg
$\boldsymbol{\omega}_C^T$	$\mathbf{0}$	$\pm$ 0.75	deg/s

produces a commanded force/torque at discrete, 1-second intervals. Episodes during training are limited to 150 seconds (a rough approximation of the time needed to dock) to gather suitable data while retaining efficiency in the overall learning process. However, for testing, the time limit is extended to 250 seconds (an arbitrary time limit greater than any time length needed for successful docking) to make sure valid docking trajectories are not prematurely terminated.

An objective of this research is to synthesize a feedback control law that is robust to significant uncertainty in the initial condition of the docking maneuver. To this end, the RL goal is to generate a docking policy that can successfully be employed within a wide range of initial conditions. This range of initial conditions should wholly contain the range of uncertainty with respect to which robustness is required. For the Apollo transposition and docking maneuver, the initial condition range is specifically defined as in Table 4.5. Monte Carlo tests of the policy randomly sample

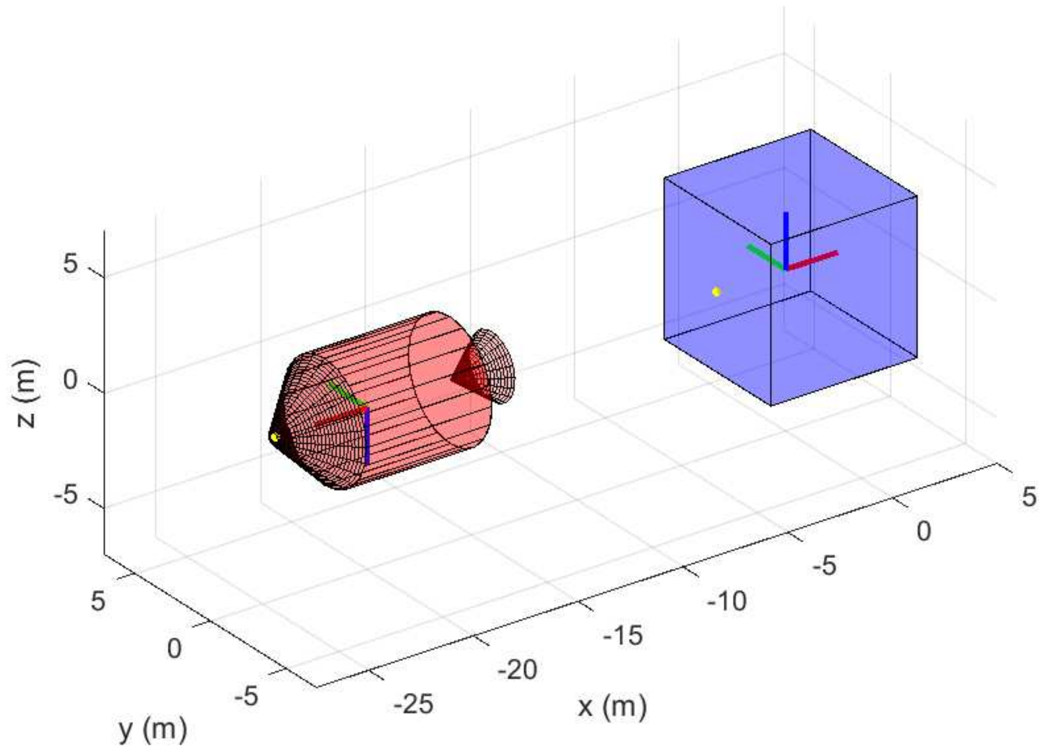


Figure 4-3: Modeling the Apollo transposition and docking maneuver. The CSM is shown in red while the LM rectangular collision area is shown in blue. The LM docking port is the yellow point on the collision area surface.

an initial condition (in each state variable) for the trajectory according the "Testing Range". However, to ensure the policy learns across a wide region of the state space, and thus gains more robust qualities, each training episode's initial conditions are sampled from the wider "Training Range" (also shown in Table 4.5).

A suitable LQR reference gain $\mathbf{K}$ was derived by tuning the LQR cost terms to result in a translational trajectory lasting 105 seconds for the nominal initial condition case $\left(\mathbf{r}_C^W = [-20,\ 0,\ 0] \text{ m},\ \mathbf{v}_C^W = [0,\ 0,\ 0] \text{ m/s}\right)$. The LQR origin state was adjusted by a $+3$ meter offset in the x-axis of the inertial frame to account for the required non-zero final velocity. The resulting position and velocity of CSM center of mass from the LQR reference accelerations is shown in Fig. 4-4.

Table 4.5: Initial condition range. The testing range is used for closed-loop simulation testing while the training range is employed during learning.

State Term	Testing Range	Training Range	Units
$\mathbf{r}_C^W(0)$	$[-20,\ 0,\ 0]\ \pm\ 2$	$[-20,\ 0,\ 0]\ \pm\ 4$	m
$\mathbf{v}_C^W(0)$	$[0,0,0]\ \pm\ 0.1$	$[0,0,0]\ \pm\ 0.2$	m/s
$[\phi_C^W,\ \theta_C^W,\ \psi_C^W](0)$	$[0,180,0]\ \pm\ 20$	$[0,180,0]\ \pm\ 40$	deg
$\boldsymbol{\omega}_C^C(0)$	$[0,\ 0,\ 0]\ \pm\ 5$	$[0,\ 0,\ 0]\ \pm\ 10$	deg/s
$[\phi_T^W,\ \theta_T^W,\ \psi_T^W](0)$	$[0,0,0]\ \pm\ 0$	$[0,0,0]\ \pm\ 0$	deg
$\boldsymbol{\omega}_T^T(0)$	$[0,\ 0,\ 0]\ \pm\ [2.5,\ 0,\ 0]$	$[0,\ 0,\ 0]\ \pm\ [5,\ 0,\ 0]$	deg/s

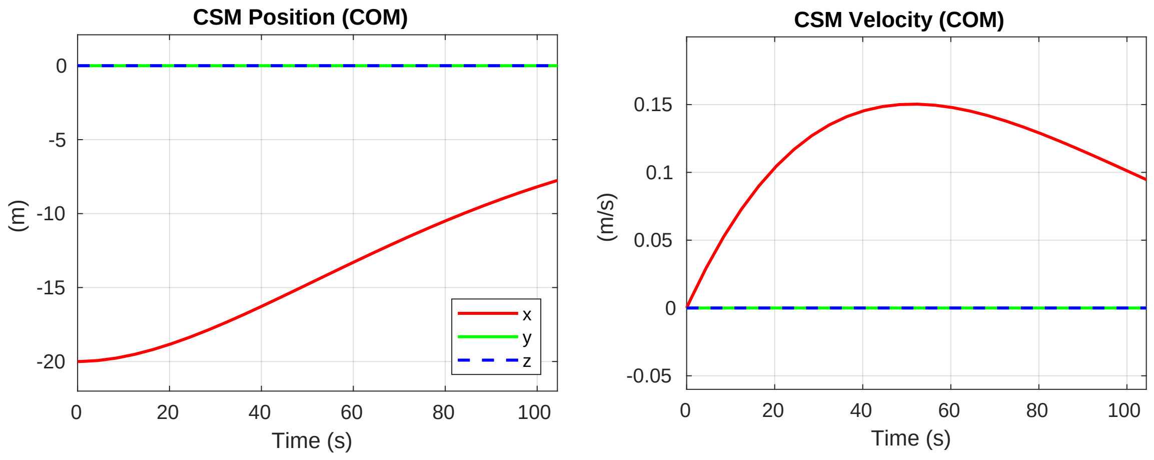


Figure 4-4: The LQR reference for the translational trajectory in the nominal initial condition case. Note that the final, non-zero position of the center of mass (COM) corresponds to a successful dock in the inertial frame.

4.4 Results and Discussion

4.4.1 Learning Results

For training the agent to perform the Apollo transposition and docking maneuver, PPO was implemented using PyTorch[1] and building on Patrick Coady's open-source work[2]. Learning occurred over 1×10^6 episodes. The agent accumulated batches of 128 episodes before performing a policy and state-value function learning update (according to (4.11)) using the collected data. The initial conditions for each episode

[1]https://pytorch.org/
[2]https://github.com/pat-coady/trpo

were randomly sampled from the training range shown in Table 4.5. Table 4.6 shares key learning parameters.

Table 4.6: Training and reward parameters.

Parameter	Value
KL_{des}	0.001
γ_1	0.98
γ_2	0.995
$\mathbf{Q}_t$	$2 \times 10^5 \mathbf{I}_3$
$\mathbf{Q}_a$	$20\mathbf{I}_6$
$\mathbf{R}$	$\left([10, 10, 10, 1.11, 1.11, 1.11]^\top \times 10^{-6}\right)\mathbf{I}_6$
c	10
d	1000

The algorithm kept track of the "best" policy over the course of learning by performing a 128-episode test using the current policy after each update. The 128 episodes for this deterministic test (i.e., no policy variance) used initial conditions at the limits of the testing ranges shown in Table 4.5. Specifically, the combinations of the minimum/maximum initial values for the seven variables

$$\left[\left(v_C^W\right)_x, \left(v_C^W\right)_y, \left(v_C^W\right)_z, \left(\omega_C^C\right)_x, \left(\omega_C^C\right)_y, \left(\omega_C^C\right)_z, \left(\omega_T^T\right)_x\right] \tag{4.27}$$

were used as edge cases to provide an accurate evaluation of the policy ($2^7 = 128$). If the current policy was able to achieve greater than or equal to the number of successful docks from the previous, "best" policy, it was saved as the new "best" policy. This process ensured that a high-performing policy was extracted over the duration of the learning process.

Figure 4-5 depicts the mean score (sum of rewards) per batch over the course of the entire learning process. Also included are the score portions attributed to the main shaping reward terms in (4.24) (LQR reference, attitude, and control). Generally, the terms were maximized roughly in unison. Each reward term had a significant increase early on in the learning process, with the remainder of the learning process dedicated to making fine-tuned improvements to achieve more docks, follow the LQR reference more closely, and decrease control efforts. There is a section of learning (episodes 400,000-600,000) where the scores increased significantly. This is where the policy started to experience successful docks and generally reached the approximate optimal solution.

Figure 4-6 depicts the KL-divergence between policy updates, the PPO clipping parameter (ξ), and the maximum action variance over the course of learning. The variance shows the expected trend: initially large to permit adequate exploration of the action space and then tapering off as the policy converges. The significant decrease in variance around episodes 500,000-700,000 reflect the strong score improvements shown in Fig. 4-5. This exemplifies the desired behavior of the policy learning to decrease variance and further exploit the current policy upon an increase in score performance. The KL-divergence is largely controlled (via the adjustable PPO clipping parameter ξ and policy learning rate $\beta_{\boldsymbol{\theta}}$) around the desired value despite a few spikes in the early and late stages of training. The desired KL-divergence value of 0.001 was chosen to match previous RL research works [19, 18].

4.4.2 Monte Carlo Test Results

After termination of the learning process, the learned policy was tested in a series of 1000 Monte Carlo trials across randomly sampled initial conditions from the test range in Table 4.5. All variance was removed from the policy, resulting in a deterministic state-feedback control law, based on the mean policy action as a function of the state. The policy produced successful docks over all 1000 test trajectories. It took on average about 1 millisecond (on a medium performance desktop PC) for the policy neural network to compute a control input, exhibiting the fast implementation speed of an RL controller. Figure 4-7 shows five superimposed trajectories from the Monte Carlo trials. Table 4.7 shares key statistics on the Monte Carlo test. The total thrust and torque expenditures are calculated by integrating the thrust/torque commands over the trajectory. Figure 4-7 shows that the policy exhibits varied behavior over the different trials, which is especially influenced by the LM's angular velocity. This points to the motivation of developing a docking policy: the resulting feedback control

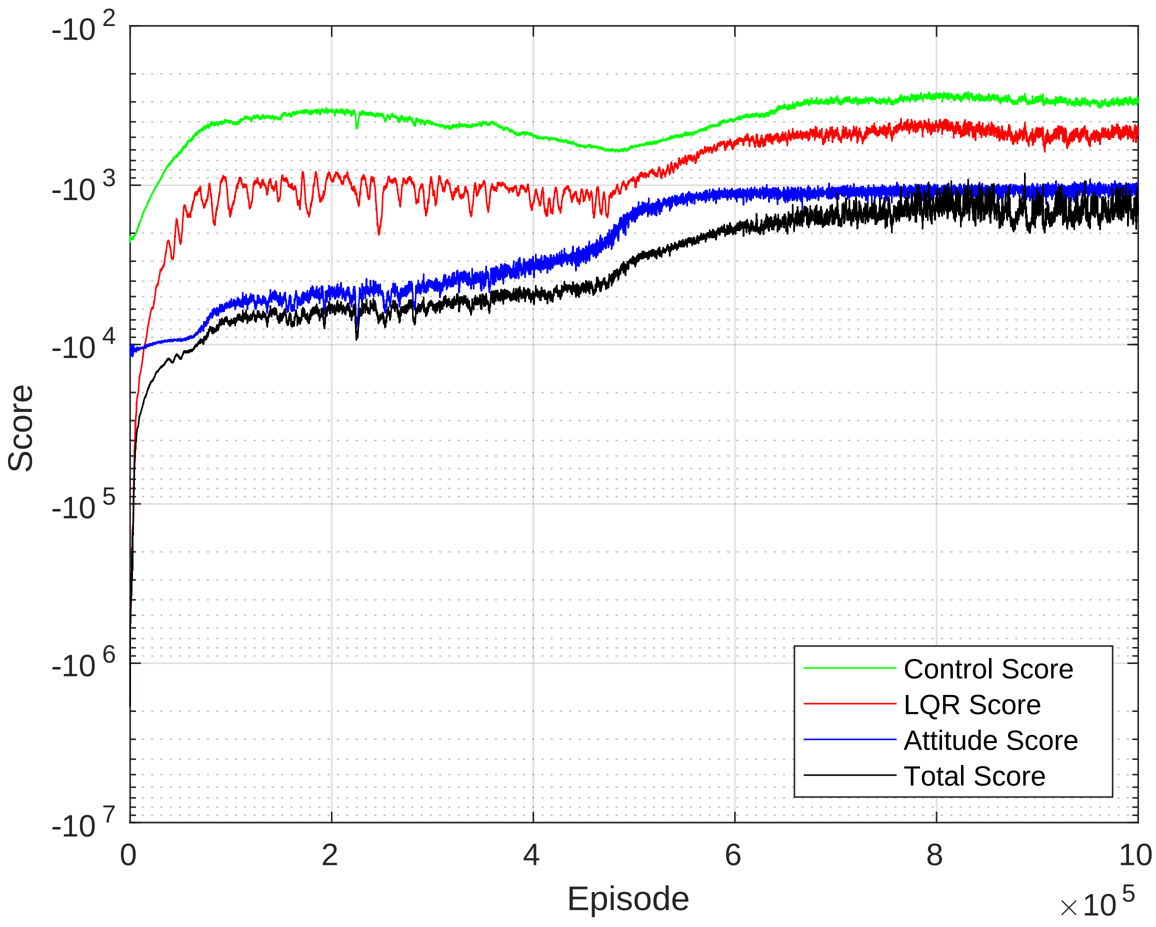


Figure 4-5: Mean scores (sum of rewards) experienced per training batch over the learning process, including score portions from some individual reward terms.

law produces trajectories in a fast and robust manner across a range of scenarios. In a general sense, the policy tended to quickly correct the attitude to match the rotational state of the LM, all while correcting translational errors and approximating the LQR trajectory to achieve the dock.

4.4.3 Comparisons with GPOPS-II Solutions

The General Purpose Optimal Control Software (GPOPS-II) [47] was utilized to provide comparisons between the converged policy and fully optimized solutions (as the former is an approximation to maximize the reward signal). Additionally, nonlinear optimization techniques (such as GPOPS-II) are typically the standard approach to characterizing (offline) optimal solutions to problems subject to nonlinear dynamics,

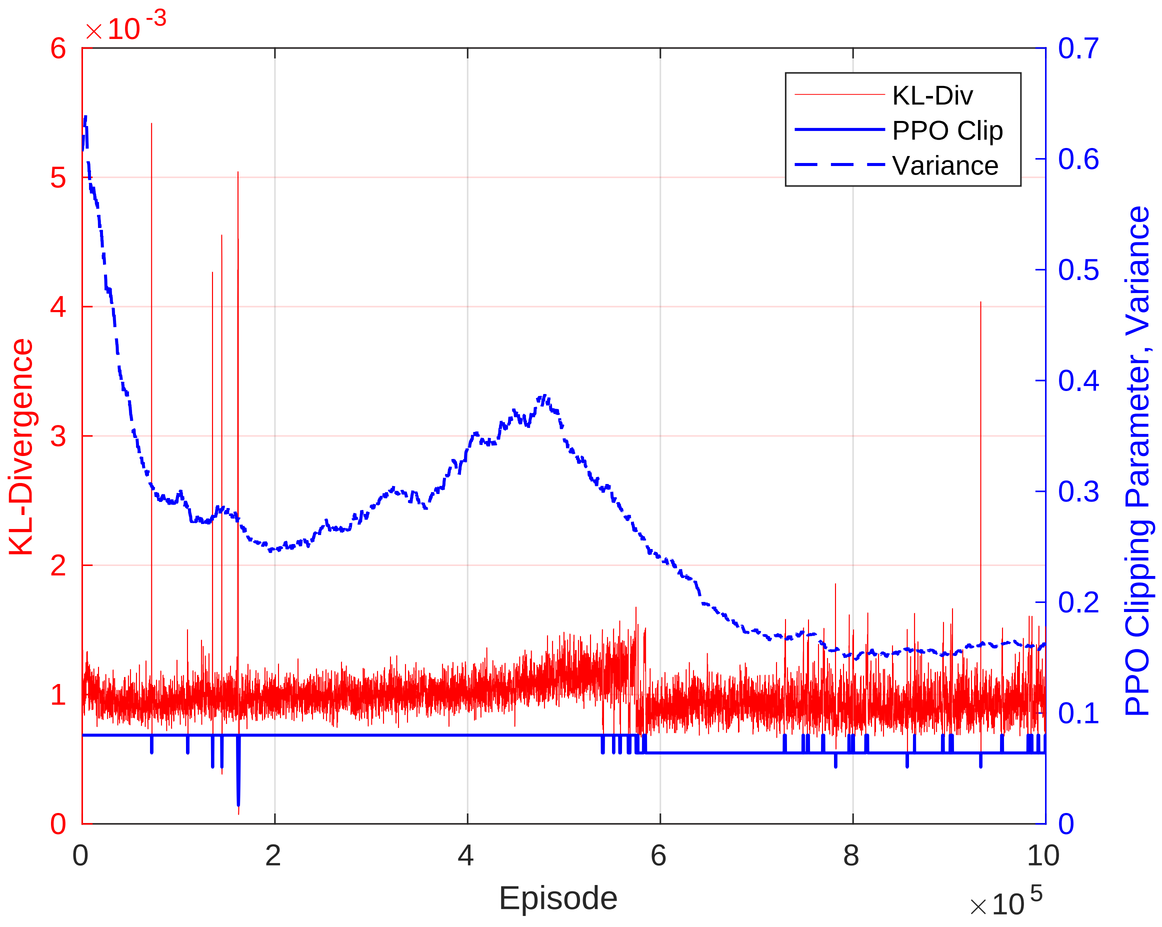


Figure 4-6: KL-divergence, PPO clipping parameter ξ, and maximum action variance over the course of learning.

Table 4.7: Monte Carlo test statistics.

Statistic	Mean	Max	Units
Trajectory time length	111.32	130	s
Cross-track position error $[y,\ z]$	$[0.031, 0.057]$	$[0.102, 0.129]$	m
Cross-track velocity error $[y,\ z]$	$[4.3, 3.0] \times 10^{-3}$	$[0.0092, 0.0095]$	m/s
Final v_x	0.069	0.078	m/s
Final attitude error (axis-angle)	1.57	3.49	deg
Final angular velocity error	$[0.03, 0.04, 0.03]$	$[0.09, 0.08, 0.06]$	deg/s
Total thrust expenditure	$8,951$	$13,471$	N
Total torque expenditure	$58,795$	$82,549$	N-m

a high-dimensional state space, and non-convex constraints (i.e., the collision constraint). Thus, the GPOPS-II solutions represent the comparison of the RL policy with standard trajectory optimization techniques more typically seen in spacecraft

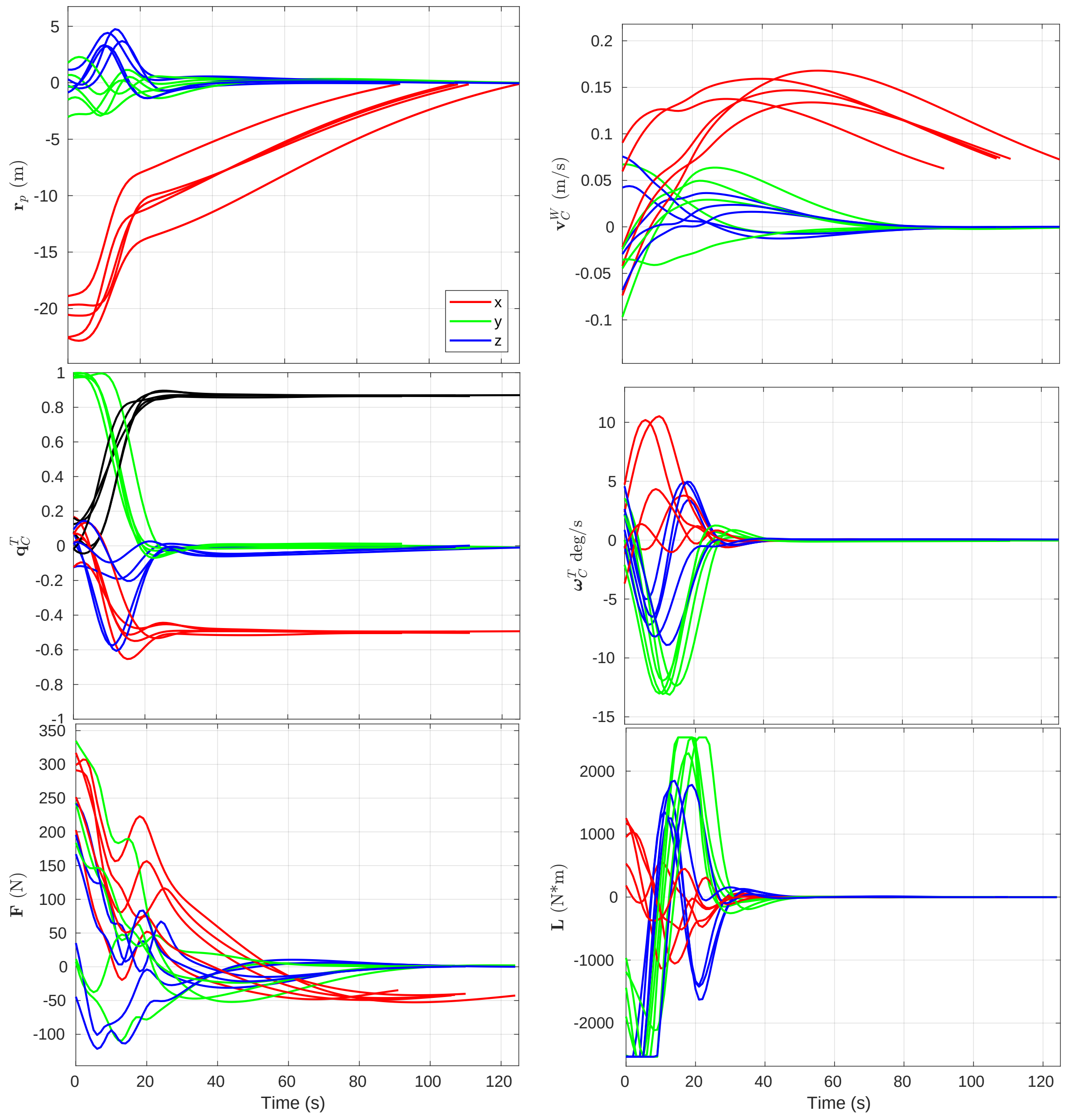


Figure 4-7: A selection of five successful trajectories from the Monte Carlo test.

guidance and control design. The first application of GPOPS-II was to calculate the optimal trajectory using the LQR reference, attitude, and control penalties from Eq. (4.24) as the objective function, thus providing a truly optimal solution to the RL reward function. The collision penalty and docking bonus terms are neglected, and instead final state constraints are included to enforce the successful docking conditions from Table 4.4. The second application of GPOPS-II was to calculate the optimal trajectory that simply minimizes overall control effort, which provides a comparison between the converged policy and the best possible solution for the problem as a whole.

The comparisons were made on the nominal docking trajectory:

$$\begin{aligned}
\mathbf{r}_C^W(0) &= [-20,\ 0,\ 0]\ \text{m} \\
\mathbf{v}_C^W(0) &= [0,\ 0,\ 0]\ \text{m/s} \\
\mathbf{q}_C^W(0) &= [0,1,0,0] \\
\boldsymbol{\omega}_C^C(0) &= [0,\ 0,\ 0]\ \text{deg/s} \\
\mathbf{q}_T^W(0) &= [1,0,0,0] \\
\boldsymbol{\omega}_T^T(0) &= [2.5,\ 0,\ 0]\ \text{deg/s}\ .
\end{aligned}$$

The GPOPS-II solutions were obtained using a fixed final time of 105 seconds to match the designed LQR reference trajectory's length. A detailed formulation of the GPOPS-II optimal control problems for the two applications is given in Appendix C. The trajectory resulting from employing the policy as a closed-loop control law $\mathbf{u}(k) = \pi_{\boldsymbol{\theta}}\big(\mathbf{x}(k)\big)$ and the optimal trajectories produced by GPOPS-II are shown shown in Figures 4-8 and 4-9. Quantitative statistics on the comparison of the two solutions are shown in Tables 4.8 and 4.9.

From Fig. 4-8, it is clear that the policy's trajectory is a sub-optimal approximation of the optimal trajectory for the reward function. The benefit of this approximation is that the control law $\mathbf{u}(k) = \pi_{\boldsymbol{\theta}}\big(\mathbf{x}(k)\big)$ resulting from the policy is quickly implementable in a closed-loop fashion (albeit within a restricted subset of the state-space). One notable difference between the two solutions is that the policy lags behind the optimal solution and produces a longer trajectory. This is likely an indicator that the policy was not fully optimized with regards to the LQR reference error penalty. Tighter adherence to the LQR reference would result in a shorter trajectory (closer to the designed 105 seconds). Also present in the policy's solution are slight, unnecessary thrust inputs in the y and z axes.

From Fig. 4-9, the policy's trajectory is noticeably different from the minimum control effort trajectory. This is largely due to the direct influence of the designed

Table 4.8: Policy vs. GPOPS-II Optimal Reward Function, Comparison Statistics

Statistic	Policy	GPOPS-II	Units
Time Length	110	105	s
Cross-track position error	$[-0.043 - 0.058]$	$[5.1, -2.0] \times 10^{-3}$	m
Cross-track velocity error	$[7.1, -4.3] \times 10^{-3}$	$[-1.4,\ 0.6] \times 10^{-3}$	m/s
Final v_x	0.071	0.0956	m/s
Final attitude error	1.12	0.056	deg
Final angular velocity error	$[3.2, 2.6, 1.2] \times 10^{-2}$	$[-0.01, -0.5, 2] \times 10^{-2}$	deg/s
Thrust expenditure	$6,952$	$6,163$	N
Torque expenditure	$63,568$	$56,379$	N-m

Table 4.9: Policy vs. GPOPS-II Optimal Control Effort, Comparison Statistics

Statistic	Policy	GPOPS-II	Units
Time Length	110	105	s
Cross-track position error	$[-0.043 - 0.058]$	$[0.150, -0.133]$	m
Cross-track velocity error	$[7.1, -4.3] \times 10^{-3}$	$[0.061,\ 0.067]$	m/s
Final v_x	0.071	0.148	m/s
Final attitude error	1.12	5.69	deg
Final angular velocity error	$[3.3, 2.6, 1.2] \times 10^{-2}$	$[-0.75, -0.75, 0.75]$	deg/s
Thrust expenditure	$6,952$	$5,170$	N
Torque expenditure	$63,568$	$8,738$	N-m

reward function on the policy's behavior. The optimal trajectory produces far less acceleration (in both translational and rotational motion) than the converged policy. Notably, to minimize control effort, the optimal trajectory results in a final state that is at the upper limit of the successful docking conditions. Based on this comparison, there needs to be clear improvements to the learning process to produce a policy for minimum-fuel docking trajectories. However, simply removing the other shaping reward terms and only penalizing control effort would not be sufficient for a successful learning process. The docking problem is far too sparse to solely rely on the discrete docking bonus term for learning. As such, the other shaping penalties (LQR reference and attitude error) are needed to provide rich reward signals and improve agent performance. This comes with the disadvantage of losing control effort optimality.

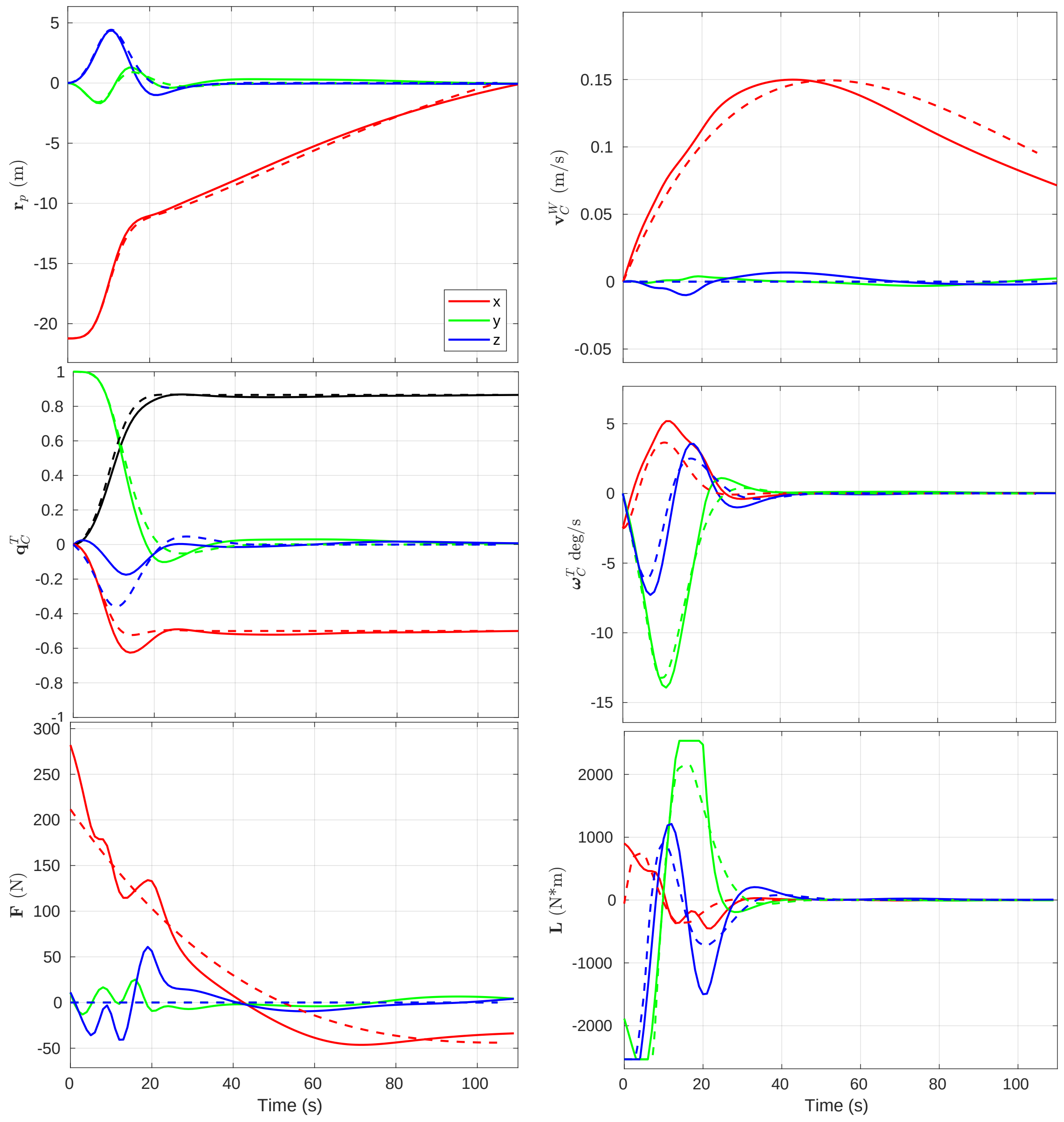


Figure 4-8: Comparison of policy and optimal reward function trajectory (GPOPS-II) on the nominal case (dashed lines represent the GPOPS-II solution).

4.4.4 Discussion

The motivation of using RL for 6-DOF docking is to generate a policy that is implementable as a feedback control law. This control law should be capable of producing successful docking maneuvers and be robust to initial conditions within a subset of the state-space. This contrasts with the more common, standard trajectory generation techniques (such as GPOPS-II) where the trajectory is optimized for a single scenario

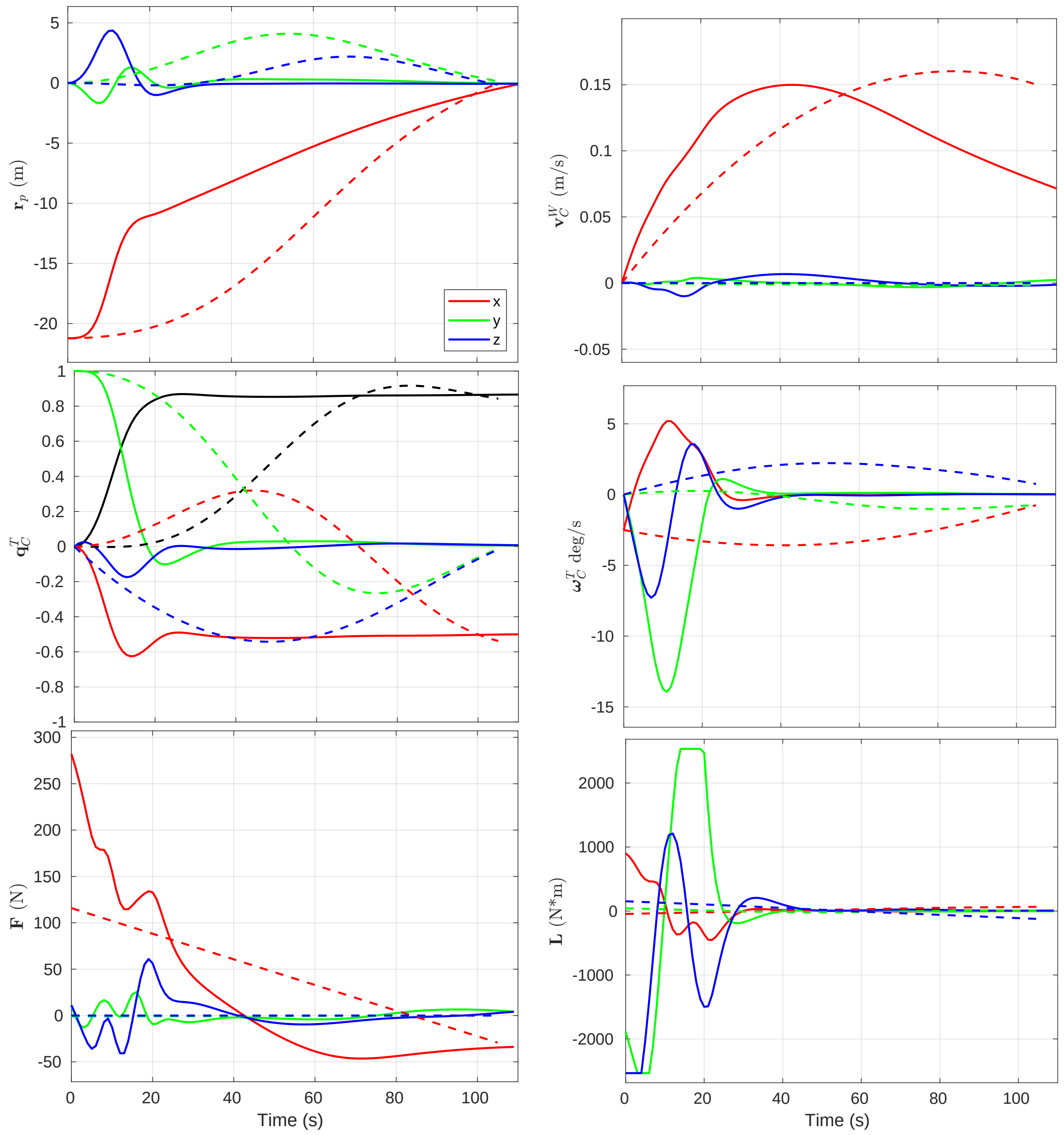


Figure 4-9: Comparison of policy and minimum control effort trajectory (GPOPS-II) on the nominal case (dashed lines represent the GPOPS-II solution).

and must be re-calculated in the case of deviations from this particular scenario. An RL-based control law is also potentially robust to disturbances, noise, uncertain dynamics, and faults if the learning process is implemented appropriately, as exemplified in Gaudet et al. [18].

However, there are several shortcomings with the current RL methodology, particularly with respect to docking maneuvers. First, it is difficult to target a desired time length for the docking trajectory, which may be a critical parameter in spacecraft docking operations. Secondly, it is difficult to strategically enforce constraint satisfaction. Docking scenarios often include complex path constraints preventing collisions, maintaining the target docking port within the sensor field-of-view, and avoiding plume impingement upon the target [36]. The policy generated in this work is able to yield trajectories that suffer no collisions, but only a simple model is used and there is no *explicit* guarantee that collisions are prevented. Extending the current methodology to include more complex constraints (such as plume impingement) would be challenging. Finally, the learning process as a whole is currently difficult to interpret from a design perspective, as it is sensitive to the tuning of learning and reward parameters. These problems are often unique to learning-based methods; for example, the online tube-based MPC algorithm outlined in Chapters 2 and 3 (not learning-based) explicitly enforces constraints and is intuitive to tune. Thus, it is expected that future research efforts in the use of RL for autonomous spacecraft maneuvers will address these concerns.

To facilitate future research in this area, several specific challenges experienced in this work, and their work-arounds, are discussed below.

1. The dilemma of exploration vs. exploitation is a challenge in most applications of RL. In this work, the variance enables the agent to explore different control actions and better improve its maximization of rewards. However, the stochastic nature of control inputs during training due to this variance directly affects the received control effort reward signal: more variance leads to more control "chatter", and the agent inevitably accrues a higher overall control cost. Therefore, high control cost coefficients can possibly lead to a sharp, premature decrease in the policy variance that prohibits successful learning convergence. By carefully tuning the control cost coefficient, as well as the degree to which policy variance adjustments are made, this problem can be mitigated.

2. The author initially experimented with a simple state error term in the reward function to encourage the agent to achieve the desired position and velocity.

However, issues were experienced with the scaling of the respective state variables and there was also no control over the trajectory time length. Thus, it is advocated to use a rich reward term [19, 18] that is equally effective across the entire state-space and also enables the agent to target a desired trajectory time length. In this work, the LQR reference reward term fulfills both of these objectives. However, there is certainly room for improvement as the time target was not precisely met, and the LQR design process is not valid for nonlinear translational dynamics. Additionally, the LQR reward basis limits the general applicability of the policy, which is typically one of the main factors when deciding to use RL for guidance and control. In this work, the use of the LQR reward term is considered a trade-off between general applicability and reliable learning convergence. While the policy still performed well in off-nominal initial condition cases, it is expected that future work will seek to use a less specific reward term to encourage general behavior. Specifically, the advantages of meta-RL could be utilized to address this issue.

3. Rare, but large, spikes in the KL-divergence between successive policy updates (due to the stochastic nature of action exploration) have the potential to derail the learning process. Thus, tight, frequent updates to the PPO clipping parameter and policy learning rate were made to ensure the KL-divergence stays close to the desired value of 0.001 throughout the entire learning process.

4. In addressing the collision avoidance constraint, a smooth, continuous penalty was found to be most effective. Additionally, it is advantageous to *not* terminate the episode upon collision. Terminating the episode may result in the agent determining that the most optimal policy is to violate the collision constraint as quickly as possible, end the episode, and thus avoid the accrual of other penalties in the reward function.

The use of meta-RL via recurrent neural networks is a potential way to address some of the above concerns. Promising results obtained in other areas of spacecraft guidance and control [18] indicate its ability to develop a policy that adapts to the scenario at hand. While this is typically geared towards fault-tolerance and adapting to unknown dynamics (which are both areas of future work in the spacecraft docking scenario), meta-RL could potentially overcome the limitations imposed by the LQR reward term in the current work. Simply put, the adaptive nature of meta-RL allows for more flexible and informative policy learning over the differing episodes. This more effective way of learning could remove the need for specific reward terms like the LQR basis in the current work, and instead could be replaced by simpler terms that encourage more general behavior.

Chapter 5

Conclusions and Future Work

5.1 Conclusions and Contributions

This work focused on extending robust control and reinforcement learning methods to address some of the challenges associated with autonomous proximity operations and their associated sources of uncertainty. Autonomously addressing uncertainty in a reliable manner is currently one of the most significant challenges in large-scale on-orbit servicing or active debris removal missions.

Chapter 2 developed the online tube-based MPC algorithm, which allows for the exogenous input (i.e., the uncertainty) to be measured and estimated online. This in turn enables the robust tube to be shrunk and grown appropriately as the trajectory progresses. This results in two main benefits. The first is improved performance in cases where the initial tube is over-conservative. The second is greater flexibility in preserving robust properties without requiring precise initial information about the exogenous input characteristics. This allows the robust controller to adapt to a wider range of scenarios and improve its practicality.

Chapter 3 outlined the autonomous proximity operations scenario of intercepting an unknown, tumbling space object. The online tube-based MPC algorithm was successfully applied in simulation to track the nominal reference trajectories despite significant errors in the estimates of the target's inertial properties. The two benefits of online, tube-based MPC mentioned above were confirmed in this realistic scenario.

Moreover, it was shown that the computational run-time of the online, tube-based MPC was reasonably fast.

Finally, chapter 4 utilized reinforcement learning to develop a policy for docking with rotating target spacecraft in a full 6-DOF dynamic environment. For the simulated Apollo transposition and docking maneuver, the policy was able to handle a range of uncertain initial conditions while preventing collisions and optimizing performance. Moreover, once learned, using the policy allowed computing control inputs to be performed very quickly, thus enabling it to run real-time in a feedback loop. Comparisons with standard, optimized trajectories using GPOPS-II yielded useful results and discussions. To the best of the author's knowledge, this is the first use of RL to develop a spacecraft docking policy in a full 6-DOF dynamic environment.

5.2 Future Work

With regards to the robust control developments covered in this thesis, there are a number of avenues for future work. Chief among them is extending the online, tube-based MPC algorithm for nonlinear dynamics. This would enable the robust controller to address uncertainty in terms of the chaser's attitude and angular velocity as well as its translational motion. The algorithm should also be tested with more types of uncertainty sources, such as noisy navigational state updates, thruster noise, and actuator faults. Different autonomous proximity scenarios should be considered as well, such as asteroid hovering and landing maneuvers or post-capture movements with a target that has flexible dynamics. Improvements also need to be made to the exogenous input prediction methodology, as it seems the ARMA model formulation is somewhat weak in capturing dynamic patterns with a relatively low amount of data samples.

With regards to the use of reinforcement learning for autonomous proximity operations, the most pressing item is to rigorously test and define key characteristics of the feedback control policy, such as its region of stability and rate of constraint satisfaction. Without stronger verification methods, reinforcement learning-based methods cannot be reliably used in autonomous proximity operations as the risk is too high. Additionally, to enable the policy to deal with a larger, more varied class of uncertainty sources, meta-learning should be implemented in the training process.

In general, more approaches that connect the domains of robust control and learning should be investigated. For instance, in the online, tube-based MPC one could potentially use a learning-based model identification method to yield more effective predictions of the exogenous inputs. Similarly, learning-based techniques could also enhance the estimation of the set of exogenous inputs. Robust control helps to ensure safety and reliability despite uncertainty; learning seeks to generalize the autonomous agent across a range of scenarios. Together, these domains have the potential to greatly improve the capabilities of autonomous proximity operations in future mission concepts.

Appendix A

2-D Double-Integrator System

This appendix provides an overview of the two-dimensional, double-integrator system utilized for demonstrating the controller methods developed in Chapter 2. The state vector $\mathbf{x} \in \mathbb{R}^2$ is defined as $\mathbf{x} = [r, v]^\top$, where r is the position and v is the velocity (only in one direction). The control input $u \in \mathbb{R}$ is the acceleration applied to the system. The sample time is set to 0.25 seconds; thus, the discrete-time $\mathbf{A}$ and $\mathbf{B}$ matrices are written as

$$\mathbf{A} = \begin{bmatrix} 1 & 0.25 \\ 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0.03125 \\ 0.25 \end{bmatrix} . \tag{A.1}$$

Limits on the position, velocity, and acceleration are implemented as polytope constraints. For the state constraints, the position is limited to 2 m in either direction from the origin, and the velocity is limited to 0.75 m/s in either direction. For the control input constraints, the acceleration is limited to 0.4 m/s^2 in either direction. The initial state is $\mathbf{x}(0) = [-1, -0.5]^\top$ and the desired state for all time-steps is $\bar{\mathbf{x}} = [1.5, 0]^\top$.

Exogenous inputs $\mathbf{w} \in \mathbb{R}^2$ are simulated by randomly sampling from a multivariate normal distribution with a diagonal covariance matrix:

$$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{w}), \ \boldsymbol{\Sigma}_\mathbf{w} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix} . \tag{A.2}$$

For each simulation trial, the characteristics of the exogenous input can be adjusted by adjusting the standard deviation values σ_r and σ_v for position and velocity as well as the mean exogenous input $\boldsymbol{\mu}_{\mathbf{w}}$.

Appendix B

Polytopic and Ellipsoidal Sets

This appendix provides a brief overview of polytope and ellipsoidal sets along with their notation. Polytope and ellipsoidal sets are an integral part in the design of online, tube-based MPC and robust control algorithms in general. A convex polyhedron set $\mathcal{P}$ takes the form of

$$\mathcal{P} = \left\{ \mathbf{d} \in \mathbb{R}^n : \mathbf{A}_p \mathbf{d} \leq \mathbf{b}_p \right\} \tag{B.1}$$

where $\mathbf{A}_p \in \mathbb{R}^{c \times n}, \mathbf{b}_p \in \mathbb{R}^c$ where c denotes the number of constraints that define the set. As such, the constraints that formulate the polyhedron set are linear. The set is deemed a polytope if it is bounded and not empty.

General ellipsoidal sets $\mathcal{E}$ take the form of

$$\mathcal{E} = \left\{ \mathbf{d} \in \mathbb{R}^n : (\mathbf{d} - \mathbf{p}_e)^\top \mathbf{P}_e^{-1} (\mathbf{d} - \mathbf{p}_e) \leq 1 \right\} \tag{B.2}$$

where $\mathbf{P}_e \in \mathbb{R}^{n \times n}, \mathbf{P}_e \succ \mathbf{0}$ is the ellipsoid shape matrix and $\mathbf{p}_e \in \mathbb{R}^n$ is the ellipsoid center. An alternative definition of ellipsoid sets is given by

$$\mathcal{E} = \left\{ \mathbf{d} \in \mathbb{R}^n : ||\mathbf{A}_e \mathbf{d} + \mathbf{b}_e|| \leq 1 \right\} \tag{B.3}$$

where $\mathbf{A}_e = \mathbf{P}_e^{-1/2}$, $\mathbf{b}_e = -\mathbf{A}_e \mathbf{p}_e$ and $||(\cdot)||$ represents the L2 norm. The constraints represented by either (B.2) or (B.3) are quadratic in nature. Examples of generic polytope and ellipsoid sets are given in Figure B-1.

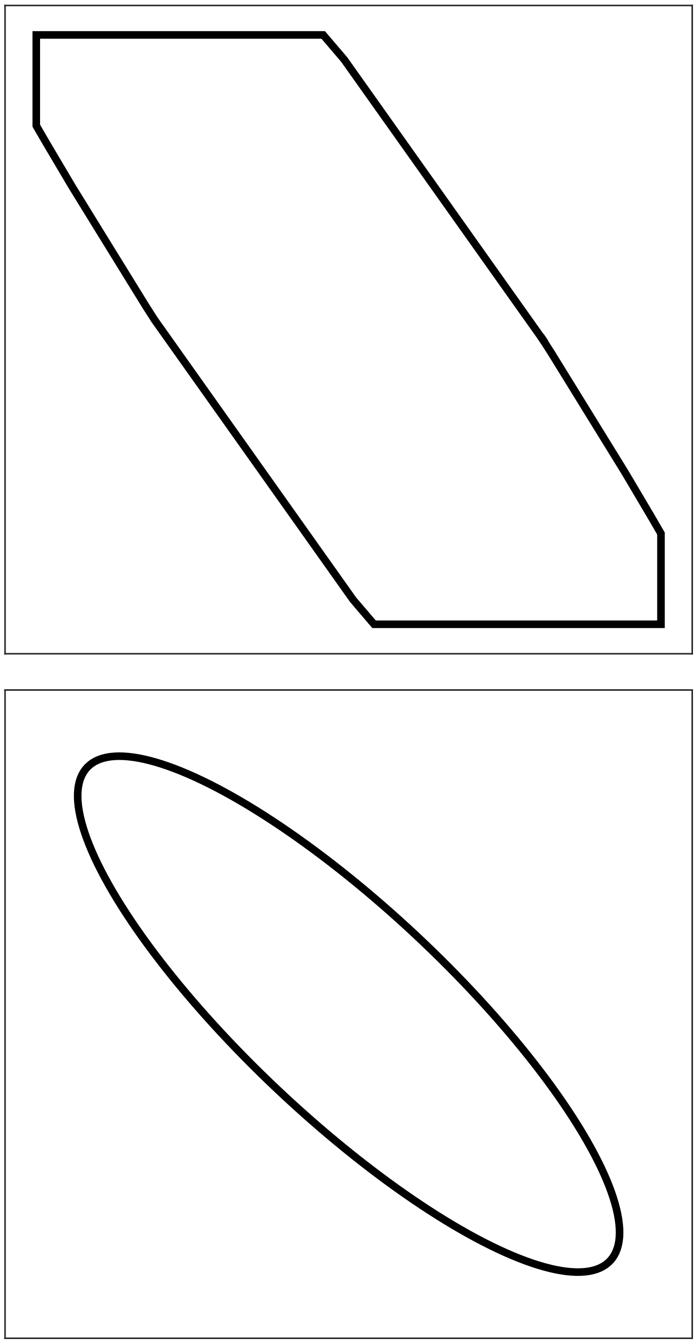


Figure B-1: Examples of polytope (top) and ellipsoidal (bottom) sets in $\mathbb{R}^2$

Appendix C

GPOPS-II Optimal Control Problems

This appendix formally outlines the optimal control problems that are solved by GPOPS-II in order to provide a comparison to the 6-DOF docking policy developed in Chapter 4. Two problems are solved; one using the reinforcement learning reward function as the objective function, and another using the minimum control effort as the objective function.

The following is the optimal control problem solved by GPOPS-II to maximize the RL reward function:

$$\min_{\mathbf{x},\mathbf{u}} \quad \int_{t_0}^{t_f} \left[\dot{\mathbf{v}}_C^W - \dot{\hat{\mathbf{v}}}_C^W\right]^\top \mathbf{Q}_t \left[\dot{\mathbf{v}}_C^W - \dot{\hat{\mathbf{v}}}_C^W\right] + \widetilde{\boldsymbol{\alpha}}^\top \mathbf{Q}_a \widetilde{\boldsymbol{\alpha}} + \mathbf{u}^\top \mathbf{R} \mathbf{u} \, dt \tag{C.1a}$$

$$\text{s.t.} \quad \dot{\mathbf{r}}_C^W = \mathbf{v}_C^W \tag{C.1b}$$

$$\dot{\mathbf{v}}_C^W = \frac{\mathbf{F}}{m} \tag{C.1c}$$

$$\dot{\mathbf{q}}_C^W = \frac{1}{2}\boldsymbol{\Omega}\boldsymbol{\omega}_C^C \tag{C.1d}$$

$$\dot{\boldsymbol{\omega}}_C^C = \mathbf{J}_C^{-1}\left(\mathbf{L} - \boldsymbol{\omega}_C^C \times \mathbf{J}_C \boldsymbol{\omega}_C^C\right) \tag{C.1e}$$

$$\dot{\mathbf{q}}_T^W = \frac{1}{2}\boldsymbol{\Omega}\boldsymbol{\omega}_T^T \tag{C.1f}$$

$$\dot{\boldsymbol{\omega}}_T = \mathbf{0} \tag{C.1g}$$

$$\mathbf{r}_p = \mathbf{r}_C^W + \mathbf{R}_C^W \mathbf{r}_{dc} - \mathbf{r}_{dt} \tag{C.1h}$$

$$\mathbf{v}_p = \mathbf{v}_C^W + \left[\boldsymbol{\omega}_C^C \times \mathbf{R}_C^W \mathbf{r}_{dc}\right] \tag{C.1i}$$

$$\mathbf{q}_C^T = \mathbf{q}_C^W \otimes \left(\mathbf{q}_T^W\right)^{-1} \tag{C.1j}$$

$$\boldsymbol{\omega}_C^T = \boldsymbol{\omega}_C^C - \boldsymbol{\omega}_T^T \tag{C.1k}$$

$$t_0 = 0 \ \text{ sec} \tag{C.1l}$$

$$t_f = 105 \ \text{ sec} \tag{C.1m}$$

$$\mathbf{r}_C^W(t_0) = [-20, 0, 0] \ \text{ m} \tag{C.1n}$$

$$\mathbf{v}_C^W(t_0) = [0, 0, 0] \ \text{ m/s} \tag{C.1o}$$

$$\mathbf{q}_C^W(t_0) = [0, 0, 1, 0] \tag{C.1p}$$

$$\boldsymbol{\omega}_c(t_0) = [0, 0, 0] \ \text{ deg/s} \tag{C.1q}$$

$$\mathbf{q}_T^W(t_0) = [1, 0, 0, 0] \tag{C.1r}$$

$$\boldsymbol{\omega}_T^T(t_0) = [2.5, 0, 0] \ \text{ deg/s} \tag{C.1s}$$

$$-0.15 \leq r_{p_x}(t_f) \leq 0, \ \left| r_{p_y}(t_f), r_{p_z}(t_f) \right| \leq 0.15 \ \text{ m} \tag{C.1t}$$

$$0.05 \leq v_{p_x}(t_f) \leq 0.15, \ \left| v_{p_y}(t_f), v_{p_z}(t_f) \right| \leq 0.1 \ \text{ m/s} \tag{C.1u}$$

$$-65 \leq \phi_C^T(t_f) \leq 55, \ \left| \theta_C^T(t_f), \psi_C^T(t_f) \right| \leq 5 \ \text{ deg} \tag{C.1v}$$

$$\left| \boldsymbol{\omega}_C^T(t_f) \right| \leq 0.75 \ \text{ deg/s} \tag{C.1w}$$

The following is the optimal control problem solved by GPOPS-II to minimize overall control effort:

$$\min_{\mathbf{x},\mathbf{u}} \quad \int_{t_0}^{t_f} \mathbf{F}^2 + \mathbf{L}^2 \ dt \tag{C.2a}$$

$$\text{s.t.} \quad \dot{\mathbf{r}}_C^W = \mathbf{v}_C^W \tag{C.2b}$$

$$\dot{\mathbf{v}}_C^W = \frac{\mathbf{F}}{m} \tag{C.2c}$$

$$\dot{\mathbf{q}}_C^W = \frac{1}{2}\boldsymbol{\Omega}\boldsymbol{\omega}_C^C \tag{C.2d}$$

$$\dot{\boldsymbol{\omega}}_C^C = \mathbf{J}_C^{-1}\left(\mathbf{L} - \boldsymbol{\omega}_C^C \times \mathbf{J}_C\boldsymbol{\omega}_C^C\right) \tag{C.2e}$$

$$\dot{\mathbf{q}}_T^W = \frac{1}{2}\boldsymbol{\Omega}\boldsymbol{\omega}_T^T \tag{C.2f}$$

$$\dot{\boldsymbol{\omega}}_T = \mathbf{0} \tag{C.2g}$$

$$\mathbf{r}_p = \mathbf{r}_C^W + \mathbf{R}_C^W\mathbf{r}_{dc} - \mathbf{r}_{dt} \tag{C.2h}$$

$$\mathbf{v}_p = \mathbf{v}_C^W + \left[\boldsymbol{\omega}_C^C \times \mathbf{R}_C^W\mathbf{r}_{dc}\right] \tag{C.2i}$$

$$\mathbf{q}_C^T = \mathbf{q}_C^W \otimes \left(\mathbf{q}_T^W\right)^{-1} \tag{C.2j}$$

$$\boldsymbol{\omega}_C^T = \boldsymbol{\omega}_C^C - \boldsymbol{\omega}_T^T \tag{C.2k}$$

$$t_0 = 0 \ \ \text{sec} \tag{C.2l}$$

$$t_f = 105 \ \ \text{sec} \tag{C.2m}$$

$$\mathbf{r}_C^W(t_0) = [-20, 0, 0] \ \ \text{m} \tag{C.2n}$$

$$\mathbf{v}_C^W(t_0) = [0, 0, 0] \ \ \text{m/s} \tag{C.2o}$$

$$\mathbf{q}_C^W(t_0) = [0, 0, 1, 0] \tag{C.2p}$$

$$\boldsymbol{\omega}_c(t_0) = [0, 0, 0] \ \ \text{deg/s} \tag{C.2q}$$

$$\mathbf{q}_T^W(t_0) = [1, 0, 0, 0] \tag{C.2r}$$

$$\boldsymbol{\omega}_T^T(t_0) = [2.5, 0, 0] \ \ \text{deg/s} \tag{C.2s}$$

$$-0.15 \leq r_{p_x}(t_f) \leq 0, \ \ \left| r_{p_y}(t_f), r_{p_z}(t_f) \right| \leq 0.15 \ \ \text{m} \tag{C.2t}$$

$$0.05 \leq v_{p_x}(t_f) \leq 0.15, \ \ \left| v_{p_y}(t_f), v_{p_z}(t_f) \right| \leq 0.1 \ \ \text{m/s} \tag{C.2u}$$

$$-65 \leq \phi_C^T(t_f) \leq 55, \ \ \left| \theta_C^T(t_f), \psi_C^T(t_f) \right| \leq 5 \ \ \text{deg} \tag{C.2v}$$

$$\left| \boldsymbol{\omega}_C^T(t_f) \right| \leq 0.75 \ \ \text{deg/s} \tag{C.2w}$$

Bibliography

[1] *Apollo Operations Handbook, Block II Spacecraft, Volume 1: Spacecraft Description*. National Aeronautics and Space Administration, 1969. SN2A-03-Block II-(1) ed.

[2] *CSM/LM Spacecraft Operation Data Book, Volume 3: Mass Properties*. National Aeronautics and Space Administration, 1969. SNA-8-D-027(III) REV 2 ed.

[3] *CSM/LM Spacecraft Operation Data Book, Volume 1: CSM Data Book, Part 1: Constraints and Performance*. National Aeronautics and Space Administration, 1970. SNA-8-D-027(I) REV 3 ed.

[4] F. Aghili and C. Y. Su. Robust Relative Navigation by Integration of ICP and Adaptive Kalman Filter Using Laser Scanner and IMU. *IEEE/ASME Transactions on Mechatronics*, 21(4):2015–2026, 2016.

[5] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.2.*, 2021.

[6] K. Berry, B. Sutter, A. May, K. Williams, B. W. Barbee, M. Beckman, and B. Williams. OSIRIS-REx Touch-and-Go (TAG) Mission Design and Analysis. *Advances in the Astronautical Sciences*, 149:667–678, 2013.

[7] G. Boyarko, O. Yakimenko, and M. Romano. Optimal Rendezvous Trajectories of a Controlled Spacecraft and a Tumbling Object. *Journal of Guidance, Control, and Dynamics*, 34(4):1239–1252, 2011.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[9] J. C. Brannan, C. R. Carignan, and B. J. Roberts. Hybrid Strategy for Evaluating On-orbit Servicing, Assembly, and Manufacturing Technologies. In *Accelerating Space Commerce, Exploration, and New Discovery Conference (ASCEND)*, November 2020.

[10] L. Breger and J. P. How. Safe Trajectories for Autonomous Rendezvous of Spacecraft. *Journal of Guidance, Control, and Dynamics*, 31(5):1478–1489, 2008.

[11] J. Broida and R. Linares. Spacecraft Rendezvous Guidance in Cluttered Environments via Reinforcement Learning. *Advances in the Astronautical Sciences*, 168, 2019.

[12] C. Buckner and R. Lampariello. Tube-Based Model Predictive Control for the Approach Maneuver of a Spacecraft to a Free-Tumbling Target Satellite. *Proceedings of the American Control Conference*, pages 5690–5697, June 2018.

[13] D. M. Chan and A. Agha-mohammadi. Autonomous Imaging and Mapping of Small Bodies Using Deep Reinforcement Learning. *IEEE Aerospace Conference Proceedings*, March 2019.

[14] A. Espinoza. *Versatile Inference Algorithms using the Bayes Tree for Robot Navigation*. PhD thesis, Massachusetts Institute of Technology, 2020.

[15] W. Fehse. *Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press, Cambridge, UK, 2003.

[16] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26, 2014.

[17] J. L. Forshaw, G. S. Aglietti, N. Navarathinam, H. Kadhem, T. Salmon, A. Pisseloup, E. Joffre, T. Chabot, I. Retat, R. Axthelm, S. Barraclough, A. Ratcliffe, C. Bernal, F. Chaumette, A. Pollini, and W. H. Steyn. RemoveDEBRIS: An in-orbit active debris removal demonstration mission. *Acta Astronautica*, 127:448–463, 2016.

[18] B. Gaudet, R. Linares, and R. Furfaro. Adaptive guidance and integrated navigation with reinforcement meta-learning. *Acta Astronautica*, 169:180–190, 2020.

[19] B. Gaudet, R. Linares, and R. Furfaro. Deep reinforcement learning for six degree-of-freedom planetary landing. *Advances in Space Research*, 65(7):1723–1741, 2020.

[20] F. Gavilan, R. Vazquez, and E. F. Camacho. Chance-constrained model predictive control for spacecraft rendezvous with disturbance estimation. *Control Engineering Practice*, 20(2):111–122, 2012.

[21] K. Hovell and S. Ulrich. Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance. *Journal of Spacecraft and Rockets*, 58(2):254–264, 2021.

[22] C. Jewison, R. S. Erwin, and A. Saenz-Otero. Model Predictive Control with Ellipsoid Obstacle Constraints for Spacecraft Rendezvous. *IFAC Proceedings Volumes*, 28(9):257–262, 2015.

[23] C. Jewison and D. W. Miller. Probabilistic Trajectory Optimization Under Uncertain Path Constraints for Close Proximity Operations. *Journal of Guidance, Control, and Dynamics*, 41(9):1843–1858, 2018.

[24] C. M. Jewison. *Guidance and Control for Multi-stage Rendezvous and Docking Operations in the Presence of Uncertainty*. PhD thesis, Massachusetts Institute of Technology, 2017.

[25] B. Jiang, Q. Hu, and M. I. Friswell. Fixed-Time Rendezvous Control of Spacecraft With a Tumbling Target Under Loss of Actuator Effectiveness. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1576–1586, 2016.

[26] C. Joppin and D. E. Hastings. On-Orbit Upgrade and Repair: The Hubble Space Telescope Example. *Journal of Spacecraft and Rockets*, 43(3):614–625, 2006.

[27] D. P. Kingma and J. L. Ba. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations*, May 2015.

[28] B. Kouvaritakis and M. Cannon. *Model Predictive Control*. Springer, Switzerland, 2016.

[29] D. Kucharski, G. Kirchner, Franz K., C. Fan, R. Carman, C. Moore, A. Dmytrotsa, M. Ploner, G. Bianco, M. Medvedskij, A. Makeyev, G. Appleby, M. Suzuki, J. M. Torre, Z. Zhongping, L. Grunwaldt, and Q. Feng. Attitude and Spin Period of Space Debris Envisat Measured by Satellite Laser Ranging. *IEEE Transactions on Geoscience and Remote Sensing*, 52(12):7651–7657, 2014.

[30] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[31] R. Lampariello, H. Mishra, N. Oumer, P. Schmidt, M. De Stefano, and A. Albu-Schaffer. Tracking Control for the Grasping of a Tumbling Satellite with a Free-Floating Robot. *IEEE Robotics and Automation Letters*, 3(4):3638–3645, 2018.

[32] D. R. Lee and H. Pernicka. Optimal Control for Proximity Operations and Docking. *International Journal of Aeronautical and Space Sciences*, 11(3):206–220, 2010.

[33] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho. On the design of Robust tube-based MPC for tracking. *IFAC Proceedings Volumes*, 41(2):15333–15338, 2008.

[34] L. Ljung. *System Identification: Theory for the User (2nd edition)*. Pearson, Upper Saddle River, NJ, 1999.

[35] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. *Proceedings of the CACSD Conference*, pages 284–289, September 2004.

[36] D. Malyuta, T. Reynolds, M. Szmuk, B. Açıkmeşe, and M. Mesbahi. Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints. *Proceedings of the AIAA SciTech Forum*, January 2020.

[37] M. Mammarella, E. Capello, H. Park, G. Guglieri, and M. Romano. Tube-based robust model predictive control for spacecraft proximity operations in the presence of persistent disturbance. *Aerospace Science and Technology*, 77:585–594, 2018.

[38] C. P. Mark and S. Kamath. Review of Active Space Debris Removal Methods. *Space Policy*, 47:194–206, 2019.

[39] F. L. Markley. Attitude Error Representations for Kalman Filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317, 2003.

[40] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[41] M. Mirshams and M. Khosrojerdi. Attitude control of an underactuated spacecraft using tube-based MPC approach. *Aerospace Science and Technology*, 48:140–145, 2016.

[42] C. E. Oestreich, R. Linares, and R. Gondhalekar. Autonomous Six-Degree-of-Freedom Spacecraft Docking Maneuvers via Reinforcement Learning. *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, August 2020.

[43] C. E. Oestreich, R. Linares, and R. Gondhalekar. Autonomous Six-Degree-of-Freedom Spacecraft Docking with Rotating Targets via Reinforcement Learning. *To appear in the Journal of Aerospace Information Systems*, 2021.

[44] H. Park, S. Di Cairano, and I. Kolmanovsky. Model Predictive Control of Spacecraft Docking with a Non-rotating Platform. *IFAC Proceedings Volumes*, 44(1):8485–8490, 2011.

[45] H. Park, R. Zappulla, C. Zagaris, J. Virgili-Llop, and M. Romano. Nonlinear Model Predictive Control for Spacecraft Rendezvous and Docking with a Rotating Target. *Advances in the Astronautical Sciences*, 160:1135–1148, 2017.

[46] G. Patterson, M. Sorge, and W. Ailor. Space Traffic Management in the Age of New Space. *Center for Space Policy and Strategy, the Aerospace Corporation*, 2018.

[47] M. A. Patterson and A. V. Rao. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Transactions on Mathematical Software*, 41(1):1–37, 2014.

[48] B. T. Polyak, A. V. Nazin, M. V. Topunov, and S. A. Nazin. Rejection of Bounded Disturbances via Invariant Ellipsoids Technique. *Proceedings of the IEEE Conference on Decision and Control*, pages 1429–1434, December 2006.

[49] C. M. Pong, A. Saenz-Otero, and D. W. Miller. Autonomous Thruster Failure Recovery on Underactuated Spacecraft Using Model Predictive Control. *Advances in the Astronautical Sciences*, 141:107–126, 2011.

[50] S. V. Raković, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne. Invariant Approximations of the Minimal Robust Positively Invariant Set. *IEEE Transactions on Automatic Control*, 50(3):406–410, 2005.

[51] N. T. Redd. Bringing satellites back from the dead: Mission Extension Vehicles give defunct spacecraft a new lease on life. *IEEE Spectrum*, 57(8):6–7, 2020.

[52] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel. Trust Region Policy Optimization. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1889–1897, July 2015.

[53] J. Schulman, F. Wolski, and P. Dhariwal. Proximal Policy Optimization Algorithms. *Computing Research Repository*, 2017.

[54] A. Scorsoglio, R. Furfaro, R. Linares, and M. Massari. Actor-Critic Reinforcement Learning Approach to Relative Motion Guidance in Near-Rectilinear Orbit. *Advances in the Astronautical Sciences*, 168:1737–1756, 2019.

[55] T. P. Setterfield, D. W. Miller, A. Saenz-Otero, E. Frazzoli, and J. J. Leonard. Inertial Properties Estimation of a Passive On-orbit Object Using Polhode Analysis. *Journal of Guidance, Control, and Dynamics*, 41(10):2214–2231, 2018.

[56] M. D. Shuster. A Survey of Attitude Representations. *Journal of the Astronautical Sciences*, 41(4):439–517, 1993.

[57] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone. Spacecraft Autonomy Challenges for Next-Generation Space Missions. *Advances in Control System Technology for Aerospace Applications*, 460:1–48, 2016.

[58] S. Stoneman and R. Lampariello. A Nonlinear Optimization Method to Provide Real-Time Feasible Reference Trajectories to Approach a Tumbling Target Satellite. *Proceedings of the 13th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, May 2016.

[59] B. Sullivan, B. Kelm, G. Roesler, and C. G. Henshaw. Robotic Satellite Service Concept: On-Demand Capabilities in GEO. *Proceedings of the AIAA SPACE Conference and Exposition*, August 2015.

[60] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (2nd edition).* MIT Press, Cambridge, MA, 2018.

[61] J. Telaar, W. Rackl, M. De Stefano, R. Lampariello, N. Santos, P. Serra, M. Canetri, F. Ankersen, and J. Gil-fernandez. GNC architecture for the e.Deorbit mission. *Proceedings of the 7th European Conference for Aeronautics and Space Sciences (EUCASS)*, 2017.

[62] B. E. Tweddle, A. Saenz-Otero, J. J. Leonard, and D. W. Miller. Factor Graph Modeling of Rigid-body Dynamics for Localization, Mapping, and Parameter Estimation of a Spinning Object in Space. *Journal of Field Robotics*, 32(6):897–933, 2015.

[63] W. S. Widnall. Apollo Guidance Navigation and Control: Guidance System Operations Plan for Manned CM Earth Orbital and Lunar Missions Using Program COLOSSUS I and Program COLOSSUS IA. Technical Report R-577 Section 3, MIT Instrumentation Laboratory, Cambridge, MA, December 1968.

[64] W. Xu, B. Liang, B. Li, and Y. Xu. A universal on-orbit servicing system used in the geostationary orbit. *Advances in Space Research*, 48(1):95–119, 2011.