Master's Thesis

DLR-IB-RM-OP-2017-17

Tube-based Model Predictive Control for the Approach Maneuver of a Spacecraft to a free-tumbling Target Satellite

Caroline Buckner

DLR German Aerospace Center
Robotics and Mechatronics Center

Weßling

Deutsches Zentrum für Luft- und Raumfahrt

Institut für Robotik und Mechatronik	BJ.: 2017
	IB.Nr.: DLR-IB-RM-OP-2017-17

MASTERARBEIT

TUBE-BASED MODEL PREDICTIVE CONTROL FOR THE APPROACH MANEUVER OF A SPACECRAFT TO A FREE-TUMBLING TARGET SATELLITE

Freigabe:	Der Bearbeiter:	Unterschriften
	Caroline Elizabeth Buckner	[illegible]
	Betreuer:	
	Roberto Lampariello	[illegible]
	Der Institutsdirektor	
	Dr. Alin Albu-Schäffer	[illegible]

Dieser Bericht enthält 206 Seiten, 55 Abbildungen und 5 Tabellen

Ort: Oberpfaffenhofen	Datum: 26.01.2017	Bearbeiter: Caroline Elizabeth Buckner	Zeichen:

Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

Munich, ……………………… ……………………………….
(Date) (Author's signature)

Acknowledgements

I would first like to thank my thesis supervisor, Roberto Lampariello, for his patience, support, guidance, and help in channelling my sometimes very scattered thoughts. His encouragement and feedback has been invaluable. I am genuinely looking forward to continuing to work with him.

I would also like to thank my family. Nik, Rufus, Sophie, and Anni, have provided me with a constant environment of love and support, stayed up the long nights with me, and, kept me in coffee and chocolate. Thank you for doing this with me. This would not have been possible without you. To my parents, my brothers, and Hartmut and Heidi, thank you for your constant support, in its various forms, and interest in my work. I love you all.

Abstract

Rendezvous and proximity maneuvers are historically critical operations. Such maneuvers include the approach and docking of a transport vessel to the International Space Station or the recovery of a tumbling satellite. Numerous control approaches have been proposed in recent decades for handling such maneuvers. In the past decade, many of these approaches have been based in Model Predictive Control. In this work, a tube-based robust model predictive controller is proposed for the robust control of a rendezvous maneuver.

In the first part of this thesis, the theory for rendezvous and proximity maneuvers and for nominal and tube-based robust model predictive control is presented. Tube-based robust model predictive control is then applied to the well-studied double pendulum problem. The design methodology and controller are implemented in software, and the controller is simulated to reproduce the results presented for the application of this control method to the double pendulum problem in literature.

In the second part of this thesis, the tube-based robust model predictive control framework is applied to a rendezvous maneuver with the unresponsive, tumbling Envisat. Uncertainty is introduced to this application of robust model predictive control through the uncertainty in the motion of the Envisat satellite. The chaser spacecraft will be required to successfully rendezvous with Envisat while tracking a pre-determined optimal trajectory and robustly satisfying the system constraints.

Keywords: tube-based robust model predictive control, satellite, tumbling, rendezvous, reference tracking

Contents

List of figures

List of tables

List of figures

List of symbols

A. General

$\mathbf{0}$	zero matrix of the appropriate dimensions
$\mathbf{1}$	ones matrix of the appropriate dimensions
$\mathbf{I}$	identity matrix of the appropriate dimension

B. Rendezvous maneuvers

$\mathbf{F} = [F_x, F_y, F_z]$	applied forces
m_c	mass of the chaser
μ	gravitational constant, Earth
n	rate of orbit
ω_p	angular velocity of the target
$\boldsymbol{\delta}\mathbf{r} = [\delta x, \delta y, \delta z]$	relative position of the chaser to the target
r_t	radius of the target
$\mathbf{R_c}$	position vector of the chaser from the center of the Earth, the origin of the inertial frame
$\mathbf{R_t}$	position vector of the target from the center of the Earth, the origin of the inertial frame
R_t	modulus of R_t
$\mathbf{u} = [u_x, u_y, u_z]$	acceleration of the chaser

C. MPC theory

A	system matrix
B	input matrix
C	output matrix

D	feedthrough matrix
$\boldsymbol{\kappa}_{\mathbf{N}}(\cdot)$	MPC control law
$l(\cdot,\cdot)$	stage cost function
m	input dimensions
n	state dimension
N	prediction horizon length
P	terminal weighting matrix, solution of the Riccati equation
p	output dimension
Q	state weighting matrix
R	control weighting matrix
T_s	sample time
$\mathbb{U}$	control constraint set
$\mathbf{u}$	sequence of control inputs
$u, u(k)$	control inputs, discrete time
$u(t)$	control inputs, continuous time
$V_f(\cdot)$	terminal cost function
$V_N(\cdot)$	cost function
$\mathbb{X}$	state constraint set
$\mathbb{X}_f$	terminal constraint set
$x, x(k)$	current state, discrete time
$x(k+1)$	next state, discrete time
$x(k+i\|k)$	estimated state at sample step $\mathrm{k+i}$ as determined at sample k
$x(t)$	current state, continuous time
$y, y(k)$	output, discrete time
$y(t)$	output, continuous time

D. Tube-based robust MPC theory

A_e	state matrix for the extended state
ϵ	percentage error bound
K_{dr}	disturbance rejection gain
K_{lqr}	LQR gain
K_θ	closed loop gain
Ω	RPI set for the perturbed system
Ω^e	Maximal robust positively invariant set
$\mathbb{U}$	robust input constraint
$\bar{\mathbb{U}}$	nominal input constraint
u	perturbed system actuation
v	nominal actuation
$\mathcal{W}$	set of disturbances
w	unknown, bounded input disturbance
$\mathbb{X}$	robust state constraint
$\bar{\mathbb{X}}$	nominal state constraint
x	perturbed system state
x^+	successive perturbed system state
x_e	the extended state incorporating the state and θ
y	system output

$\mathcal{Z}$	mRPI set
z	nominal system state
z^{+}	successive nominal state

E. Problem formulation and control strategy

A	state matrix, discretized
A_c	state matrix, continuous time
B	control matrix, discretized
B_c	control matrix, continuous time
F	force applied by thruster
m_c	mass of chaser
μ	gravitational constant of Earth
n	orbital rate
$\mathbf{r_d}$	position of docking port
r_t	radius of target sphere
T_s	sampling time
$\mathbb{U}$	robust input constraint
$\bar{\mathbb{U}}$	nominal input constraint
$\mathbb{X}$	robust state constraint
$\bar{\mathbb{X}}$	nominal state constraint
$\mathcal{Z}$	mRPI set

Nomenclature

CWH	Clohessy-Wiltshire-Hill equations
DLTI	discrete linear time invariant system
DOF	Degree(s) of freedom
HDRM	Hold Down Release Mechanism
LAR	Launch Adapter Ring
LEO	low Earth orbit
LMI	linear matrix inequality
LQP	linear-quadratic problem
LQR	linear-quadratic regulator
LTI	linear time invariant
MPC	Model Predictive Control
MPI	maximal positively invariant set
MPT, MPT3	Multi-Parametric Toolbox, v 3
mRPI	minimal robust positively invariant set
RPI	robust positively invariant set
RPO	rendezvous and proximity operations

1 Introduction

1.1 Background and motivation

Rendezvous and proximity maneuvers are historically critical operations. Such maneuvers include the approach and docking of a transport vessel to the International Space Station or the recovery of a tumbling satellite. Numerous control approaches have been proposed in recent decades for handling such maneuvers. In the past decade, many of these approaches have been based in Model Predictive Control.

It is important to consider sources of uncertainty and their management when controlling a system. In a rendezvous maneuver, there are at least two bodies for which uncertainties need be considered. As the maneuver is governed by relative dynamics, the uncertainty in the motion of one participant will have a direct impact on the uncertainty in the state of the second participant. Robust control methods are needed to account for this uncertainty.

The goal of this work is therefore to robustly conduct a rendezvous maneuver while tracking a provided optimal trajectory. A tube-based robust model predictive controller is proposed for the control of the maneuver.

1.2 Study objectives and contributions of this thesis

The first objective of this work is to demonstrate the use of the tube-based robust model predictive control methodology. A review of model predictive control and tube-based robust model predictive control is first presented. This theory is then applied to the well-studied double-pendulum control problem, which is a frequently used sample problem in literature – for example in [1-4]. In literature, the results are presented with some discussion, but with little or no exposition to the design process other than the strictly theoretical definition of the control methodology. In this work, the results for the double pendulum sample problem from literature are reproduced through an implementation in software.

The second objective of this work is to apply the proposed control methodology to a satellite rendezvous maneuver. In this rendezvous problem, the goal is to bring a chaser spacecraft from its initial position in space to a docking point on a target spacecraft along a provided reference trajectory while robustly satisfying actuation constraints. The target spacecraft is taken to be travelling on a circular orbit about the Earth; the relative motion between the spacecraft is then appropriately described using the Clohessy-Wiltshire-Hill equations. The Hill frame is centered on the center of mass of the target spacecraft revolving at a rate equal to the orbital rate, effectively equivalent to the target orbital reference frame. The target spacecraft is tumbling, and the target body frame follows this same motion. As is typical to the rendezvous phase of rendezvous and proximity operations maneuvers, the chaser and target spacecraft are assumed to be located within the same orbit. These concepts will be revisited in *chapter 2.*

The reference trajectory is provided by a motion planner, which makes use of a selected inertia and rotational velocity state for the target to determine the trajectory optimized to a set of costs. The motion planner devises a relative trajectory between the two spacecraft with respect to the target orbital frame. However, there is an amount of uncertainty in the motion of the target spacecraft resultant from uncertainty in the inertia or spin state of the target craft. In the orbital frame, the uncertainty in the motion of the docking point on the target can be described by a spherical shell, or the upper most bound of a scaled unitary ball $\mathcal{B}^n = \{b \in \mathbb{R}^n : \|b\|_p \leq 1\}$, centered on the center of mass of the target satellite. This uncertainty must be accounted for in the robust control of the chaser to the docking point. This uncertainty shall be more thoroughly considered in *chapter 6.*

Fortunately, the reference trajectory is defined relatively with respect to the two spacecraft, and, under robust reference tracking control methods, it can be validly tracked irrespective of the actual position of the docking point in the target orbital frame, permitting that the actuation constraints are not violated. As the boundary of the uncertainty is characterizable, there exist robust control methods which will allow the reference trajectory to be tracked within this uncertainty bound and actuation constraints. The development and simulation of such a robust controller is conducted and the results of the simulation presented and analysed in pursuit of this second objective.

1.3 Software packages

The controller for the sample double pendulum problem and the rendezvous task are to be designed, implemented, and evaluated using a combination of MATLAB and Simulink.

In the design phase, the CVX and Multi-parametric toolboxes are utilised in the MATLAB environment. The CVX toolbox is an open-source toolbox specifically built for the solution of disciplined convex programs [5, 6]. The toolbox is specifically targeted for linear, quadratic, and semi-definite programs. While MATLAB has a native functionality for handling matrix inequalities in optimization problems, the CVX toolbox applies a more intuitive matrix interpretation to these inequalities, providing for a more natural implementation. The Multi-parametric toolbox (MPT) is an open-source toolbox for MATLAB designed for parametric optimization, computational geometry, and model predictive control [7]. The most recent version of the toolbox, often referred to as MPT3, relies on an additional open-source MATLAB toolbox called YALMIP [8]. The inclusion of YALMIP into MPT3 allows the latter to handle custom nominal model predictive control scenarios.

In the simulation phase, the Model Predictive Control Toolbox is used [9-11]. The toolbox provides MATLAB functions and Simulink blocks for the construction, tuning, simulation, and evaluation of model predictive controllers. This is a paid-toolbox available through Mathworks.

For more information on these toolboxes, the reader is referred to the documentation cited with the toolbox.

1.4 Plan of development

The rendezvous maneuver to be conducted brings a chaser craft to the launch adapter ring on the ESA Envisat satellite. The control is conducted in the target orbital frame, and the chaser is required to track a reference trajectory provided by a motion planner. The relative orbital dynamics are described by the Clohessy-Wiltshire-Hill equations. The considered uncertainty is introduced to the system through an uncertainty in the inertia and angular velocity characteristics of the target spacecraft. A tube-based robust model predictive controller is designed taking into account this uncertainty as a state disturbance. The controller is designed in so far as possible so as to not limit the generality of the control algorithm so that it might be applied to other rendezvous maneuvers and is evaluated by simulation in Simulink. In the subsequent chapters of this work, these concepts will be presented and discussed.

In the next section, a brief review will be presented on the literature pertaining to the control of satellite rendezvous maneuvers with a focus on model predictive control methods. Then, in Part 1, the necessary theoretical background in rendezvous maneuvers and model predictive control methods are presented. A practical example of the tube-based robust model predictive control is offered in preparation for the rendezvous control task. In Part 2, the system to be controlled is characterized, a tube-based robust model predictive controller is designed for the task, and finally evaluated.

1.5 Literature Review

Spacecraft rendezvous maneuvers came onto the research scene in the late 1950's. The study of manual and automatic rendezvous maneuvers conducted by the NASA Langley Research Center lead to the development of low Earth orbit and Lunar rendezvous maneuvers. Goodman presents a detailed history of rendezvous and proximity maneuvers in [12], from the Mercury program to Space Shuttle rendezvous with the International Space Station. The reader is directed to [12], and the references within, for a detailed review of the history and literature pertaining to the satellite rendezvous and proximity maneuvers. This remainder of this literature review is dedicated to the control of rendezvous maneuvers using model predictive control and the development of tube-based robust model predictive control.

Model predictive control (MPC) [13, 14] was developed in the 1970's and is now considered a mature technique for linear and slow systems, like those found in process control [15]. There are of course many other applications than process control, such as solar technology and flight control, for which MPC methodologies are suitable [15-18]. In 2003, Richards and How analysed the performance of MPC in rendezvous maneuvers as compared to other methods in [18].

Various nominal and robust model predictive control methods have been proposed for the control of rendezvous maneuvers. In [18], a variable horizon method is proposed in which a mixed-integer linear program is required to be solved in each control cycle. This work was extended in [16], to develop what was termed failure-safe trajectories, and in [19] with rubber-band MPC, which makes use of a decreasing horizon. In [20, 21], an approach for the control of relative motion maneuvers using linear quadratic MPC with dynamically reconfigurable constraints was developed in two-dimensions. In this method, the model and constraints are re-evaluated online at each control step to determine an optimal control sequence to a stationary, rotating, or tumbling target platform while avoiding obstacles over a finite horizon. In [22], this method was extended to include the third spatial dimension, taking into account the cross-track dynamics, and in [23] the rendezvous maneuver is integrated with a docking maneuver with the same control method, but with distinct requirements, constraints, and sampling rates. In [15], it was shown that conventional MPC is not capable of handling additive disturbances (refer to *chapter 4.1*). Gavilan et al. used a *min-max worst-case disturbance* method and an estimate of the disturbance bounds to robustly satisfy constraints; producing a method robust to additive and multiplicative disturbances as well as unmodelled dynamics. In [24], a robust dual control MPC method is proposed which guarantees constraint satisfaction for simultaneous identification and control of uncertain systems.

The concepts of Tube-based robust MPC first appeared in [25], but [26] is widely said to be the formal birth of the framework. The basis of the methodology is to robustly control an uncertain system through its nominal dynamics and an additional feedback term which rejects a bounded additive disturbance. Further research was conducted for the linear regulator and presented in [27-30]. The method was adapted for tracking in [1, 3, 4, 31-33] and for non-linear systems in [34-36]. To the best of the author's knowledge, there exist only two publications, [37] and [38], on the use of tube-based robust model predictive control to control rendezvous and proximity operations maneuvers. In these two works, the satellites are taken to be travelling on elliptical orbits and the uncertainty was derived from navigation and thruster timing errors, respectively.

In [37], ideas taken from tube-based robust model predictive control are used to control a rendezvous with the participants travelling on eccentric orbits. The goal of the work is to steer the chaser spacecraft from an initial relative state to a neighbourhood about a desired set point in the presence of navigation uncertainty. The uncertainty comes from error injected into the measurements of relative position by the sensor. The costs considered are the minimization of fuel and minimizing the size of the arrival set. The study claims to improve on efficiency and infeasibility problems in the application of classical MPC, as repeated re-computations are not necessary to achieve robustness. The method does, however require some online estimation of disturbance terms, which is counter-indicative of tube-based robust model predictive control.

In [38], tube-based robust model predictive control is used as a part of a guidance algorithm robust to thrusting errors to best guarantee precision performance under propulsion uncertainties. The authors show that the robust algorithm is able to preclude the spread of error while following a strict fuel budget. They also found that the polytopic uncertainty set definition may result in a more conservative controller than necessary for this specific situation.

In this work, the rendezvous maneuver will be controlled through an implementation of tube-based robust model predictive control for tracking. The two implementations cited above are regulation implementations, foregoing the tracking steps. This difference will become clear in *chapters 3* and *4.* Another difference of this work to the two works discussed above, the uncertainty is introduced through the target, rather than the chaser.

Part 1: Theoretical Background

2 Spacecraft Rendezvous and Proximity Operations

A historically critical stage in many spacecraft missions is that of the rendezvous and proximity operations (RPO), which have received substantial consideration in literature (see [15, 18, 20-24, 37-43], and the references therein). Such maneuvers include the approach and docking of a transport vessel to the International Space Station or the recovery of a tumbling satellite. Numerous control approaches have been proposed in recent decades for handling RPOs; however none has yet emerged as universally successful [15].

Subsequently, this chapter will outline the overall rendezvous and docking process, followed by a description of the close-range rendezvous relative motion of the spacecraft and the related dynamics.

2.1 Spacecraft rendezvous and docking

Spacecraft rendezvous and docking is an interactive absolute or relative motion problem, commonly between two space vehicles. While it is normally possible for both spacecraft to maneuver, there are circumstances where only one of the vehicles is maneuverable. It is common, therefore, to consider one vehicle as active and the other passive [42]. The passive spacecraft is often referred to as the target, platform, or client, while the active vehicle is frequently termed the servicer or chaser. These terms are often used interchangeably in literature, as demonstrated in the literature cited in this chapter. This work will refer to the active and passive vehicles as chaser and target, respectively.

The rendezvous process is considered in a series of stages, as indicated in the following figure.

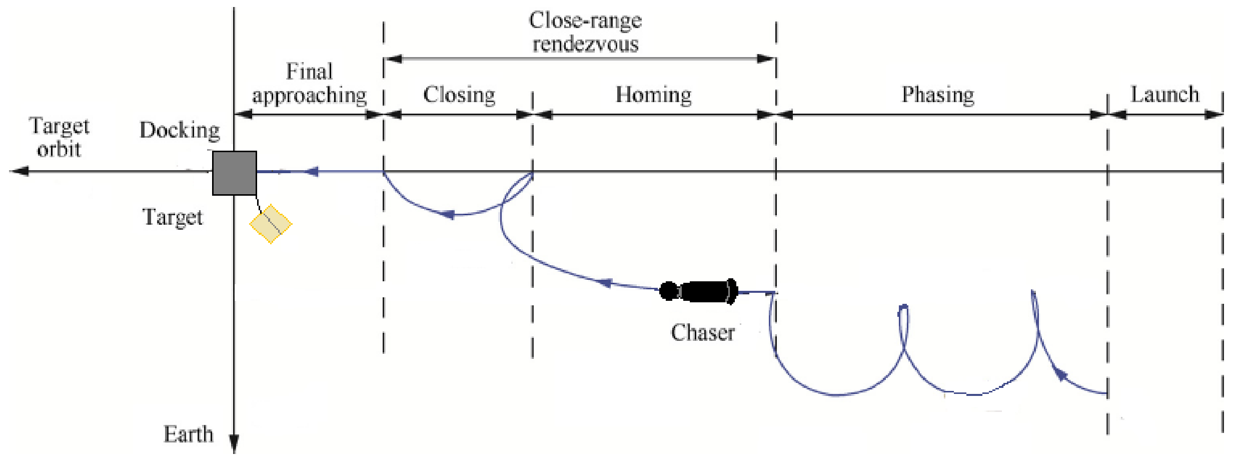


Figure 2-1 Phases of spacecraft rendezvous and docking procedure, adapted from [43]

The procedure begins with the vehicles at a remote distance, out of sight of each other [42]. The spacecraft will rendezvous from two initial, separate orbits. In this *Phasing* stage, the chaser is brought from its initial orbit into the orbit of the target [43]. With careful planning, the process can begin at the launch of the chaser spacecraft, possibly drastically reducing the maneuver time [44-46]. However, this is not required. It is required, however, that the chaser knows the approximate orbit of itself and the target [42]. The orbits can be assumed, found using a navigation package, or through ground tracking methods [42, 47, 48]. Using the orbit information, orbit transfer maneuvers are calculated to maneuver the chaser into the same orbit as the target.

The second phase therefore begins with the spacecraft on the same orbit. However, the chaser is often inserted into the target orbit a fair way away from and still out of sight of the target and allowed to drift toward it. This is called a *drift orbit* [42]. The relative insertion positon of the chaser is chosen to be suitable for docking, typically in a fairly straight line to the docking port. That is, for example, if the chaser is to dock to the aft of the target, the chaser would preferably be inserted into the platform's orbit behind the platform; or if the chaser is to approach the target from below, the chaser may be inserted into a slightly lower orbit and behind the target [42].

There are some clear advantages to transferring to a drift orbit. Such an orbit avoids collision of the vehicles. Additionally, aiming to place the chaser short of the target guarantees that correction burns will be in the same direction as the preceding burn, this avoids wasting energy and propellant by having to maneuver in the opposite direction [42].

As the chaser travels along the drift orbit, it will approach the horizon at which it will begin to be able to have direct spacecraft-to-spacecraft viewing and communication. This horizon distance is typically 3,000 to 10,000 km for spacecraft in low Earth orbits (LEO) [42]. The chaser is said at this point to have entered *Drift Orbit B* or the *homing* obit. The chaser is then in the first stage of so called *close-range rendezvous* maneuvers [43].

While in the *homing* orbit, it is possible for the chaser to use position information communicated by a cooperative target spacecraft to navigate, or the chaser can use the target as a point of light relative to the background of stars for relative navigation [42]. The chaser closes in on the target using this navigation information by simply drifting or through a series of decreasing thruster fires. At a distance of about 100 to 1 km, the chaser is often placed into a parking obit, so that the correct timing, geometry, and lighting conditions can be obtained for rendezvous and proximity operations [42, 46].

The proximity operations are divided into two phases based on the distance of the chaser from the target at which they are performed. The first phase, c*losing*, is typically conducted using a relative navigation and/or docking sensor(s) with the chaser located between 1 km and approximately 100 m from the chaser. Depending on implemented control methods, the series of very small thruster fires used to approach the target can be computed on- or offline, and executed at a rate which allows navigation to be corrected for between firings. These maneuvers are typically designed to bring the chaser to the target along either the chaser's velocity vector (V-bar) or the radial vector (R-bar) [42, 46, 49].

The second part of the rendezvous phase, often called the *terminal rendezvous phase* [39], commonly begins between 10 and 100 m from the target. The remainder of the process is dependent on if the chaser is intended to dock with the target or simply undertake an inspection mission of it. In the docking case, the chaser is maneuvered to the docking port of the target using a series of very small thruster fires or allowed to drift toward the target, exploiting the small differences in the orbital elements of the two vehicles [42]. This procedure will usually make use of a docking sensor to provide relative position and attitude information [42]. This present work is concerned with docking missions and the remainder will discount inspection missions.

The process in which the chaser physically attaches to the target is called *Docking*. The key to a successful docking is a compliant capture mechanism which can: compensate for small differences in attitude, position, velocity or even acceleration; capture the target using a gripping or grasping mechanism; couple the target and chaser, effectively into a rigid, single spacecraft, in what is called a *hard dock* [42].

The now joint spacecraft may maneuver together for an arbitrary amount of time, where relative motion between the constituent vehicles is no longer the subject of navigation. The dynamic properties of the joint spacecraft are determined and the orbit adjusted for continued safe and efficient maneuvering [42]. On completion of these *joint maneuvers*, if the mission parameters dictate, the chaser *undocks* from the target and the vehicles *separate*. After *escape*, the spacecraft continue normal operations as dictated by their mission parameters. The functions after escape are similar to those during rendezvous, except it is possible now for the target to do the maneuvering [42]. It is, of course, possible for one or both of the vehicles to be parked into a *graveyard orbit* or the orbit(s) allowed to decay and the satellite(s) to re-enter the atmosphere [42].

While these maneuvers have been thoroughly studied and docking maneuvers conducted by crewed missions with cooperative targets since the early days of space travel, the current challenge is to perform such procedures completely autonomously, possibly out of contact with any ground station, and with ever more complex mission goals – including interaction with non-cooperative targets.

2.2 RPO maneuvers and dynamics

Recall the *terminal rendezvous phase* which brings the chaser to within a small distance to the order of a few meters of the docking port. Traditionally, this relative motion problem is considered separately from the final docking process which physically tethers the two spacecraft [15, 20, 21, 37]. To this end, the chaser is brought within a mission dependent margin of the docking port from an initial position some tens of meters distant through V-bar or R-bar approach RPO maneuvers.

An experience base of rendezvous mission planning and execution during the Gemini-era turned rendezvous theory into a reality. Goodman lists the achievements of the mission which came to robustly define the practical theory in [12]. Rendezvous maneuvers became a well-practiced art under the Apollo missions [12]. In the nine years of the program, rendezvous systems and piloting techniques were successfully exercised, permitting such feats as the lunar missions, and the maneuvers made more efficient under study. The techniques developed still form the basis of RPO maneuvers today.

As indicated previously, the critical problem inherent to these maneuvers is the relative navigation. The characterization of the dynamics of this problem depend largely on the nature of the orbits of the involved spacecraft, in particular if they are taken to be circular or elliptical, and if the reference frame is inertial or rotating [39]. Furthermore, the problem is typically defined such that: the relative dynamics can be investigated within the confines of the orbital plane, defining the problem in the 2 degrees of freedom (DOF) corresponding the radial and along-track directions in translation; the cross-track dynamics are incorporated to define the problem in 3 DOF, while considering only the positional dynamics; or attitude can be additionally accounted for to expand the problem definition into 6 DOF. However, it is common for the position and attitude to be modelled separately, so the first two options are more common in literature. This work considers the case of circular orbits and 3 DOF dynamics. These dynamics will now be explored.

When the distance between a pair of spacecraft is large, as in the *Phasing* or *Homing* stages (see above in *section 2.1*), their relative dynamics are typically described in an Earth-centered inertial frame; when the distance is small, as in the *Closing* stage (again, refer above), the relative motion is typically described in a target-centered reference frame [43]. In the terminal rendezvous phase, the distance between the target and chaser is relatively small. Consider the situation illustrated in the following figure.

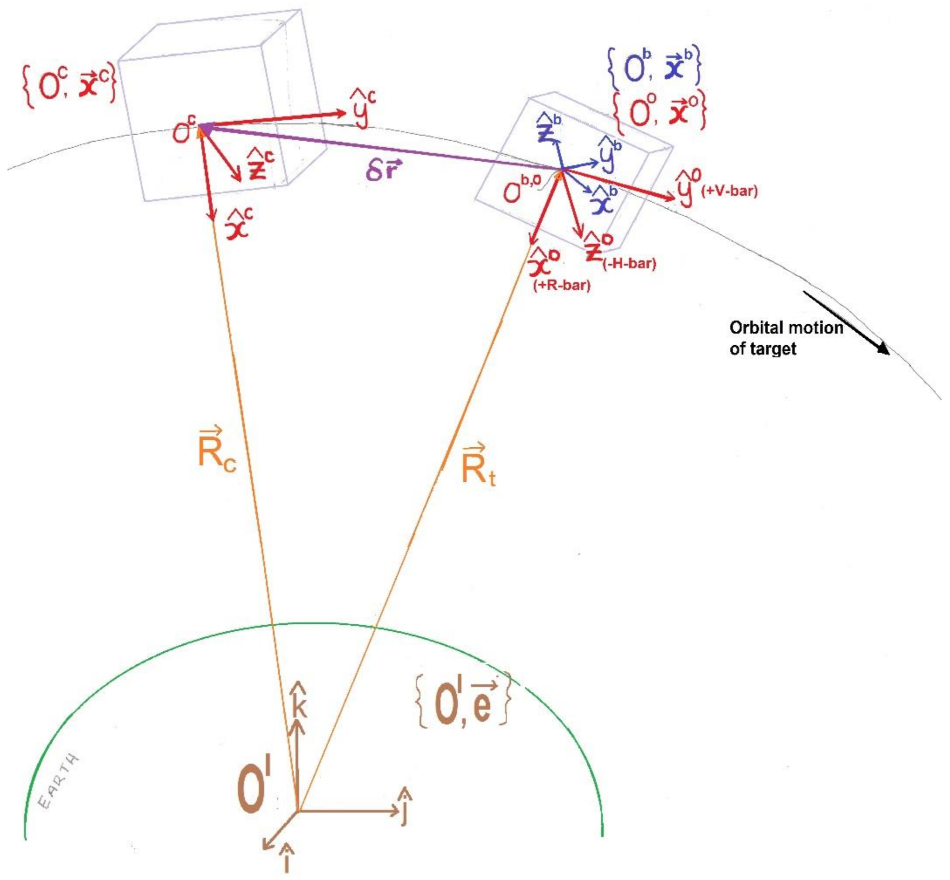


Figure 2-2 Definition of reference frames and vectors

The center of mass of the target is located on a circular orbit with a radius R_t [m] measured from the center of the Earth in the inertial frame $\{O^I, \vec{\boldsymbol{e}}\}$, with its origin fixed to the center of the Earth, giving a location of the target with respect to the center of the Earth by $\mathbf{R_t}$. The orbital frame $\{O^O, \vec{\boldsymbol{x}}^{\boldsymbol{O}}\}$ and the target body frame $\{O^b, \vec{\boldsymbol{x}}^{\boldsymbol{b}}\}$ are each centered on the center of mass of the target. Similarly, the center of mass of the chaser is located by $\mathbf{R_c}$**,** where the chaser body frame is $\{O^c, \vec{\boldsymbol{x}}^c\}$ is centered**.**

Let the target tumble at an angular velocity $\omega_t \geq 0$ [rad/s]. The chaser is represented as a point mass and is in an orbit close to that of the target. The relative position between the chaser and target is defined by $\boldsymbol{\delta}\mathbf{r} = [\delta x, \delta y, \delta z]$**.** For the ease of describing the dynamics, consider the target as a sphere of sufficient radius r_t [m] to over-bound its structure. The docking point is located in the target body frame by $\mathbf{r_d^b} = [r_{dx}^b, r_{dy}^b, r_{dz}^b]$ [m].

The relative dynamics can now be expressed. In the situation where satellites travelling relative to each other on circular orbits, the relative dynamics can be expressed using the Clohessy-Wiltshire-Hill (CWH) equations [15, 21, 39, 43, 49], as demonstrated in the following. First, consider the circular orbit of the target at radius R_0 [m] from the Earth. The target has an orbital rate of $n = \sqrt{\mu/R_0^3}$ $[\mathrm{rad/s}]$ where μ [m^3/s^2] is the gravitational constant of Earth. The reference Hill frame centers on the target's center of mass and revolves with orbital rate n with respect to the inertial frame.

Now, let the position of the target's center of mass can be re-written as

$$\mathbf{R_t} = R_0\hat{x}^O \tag{2-1}$$

and the relative positon of the chaser as

$$\boldsymbol{\delta}\mathbf{r} = \delta x\hat{x}^O + \delta y\hat{y}^O + \delta z\hat{z}^O \tag{2-2}$$

where $\delta x, \delta y, \delta z$ are components of the relative position of the chaser with respect to the center of the target. Therefore, the position of the chaser in the inertial frame is characterized by

$$\begin{aligned}\mathbf{R_c} &= \mathbf{R_t} + \boldsymbol{\delta}\mathbf{r^r}\\ &= (R_0 + \delta x)\hat{x}^O + \delta y\,\hat{y}^O + \delta z\,\hat{z}^O\end{aligned} \tag{2-3}$$

The equations of motion for spacecraft in a circular orbit are given by

$$\ddot{\mathbf{R}} = -\mu\frac{\mathbf{R_c}}{R_c^3} + \frac{1}{m_c}\mathbf{F} \tag{2-4}$$

where m_c [kg] is the mass of the spacecraft and $\mathbf{F}$ the applied forces. Substituting $R_c = \sqrt{(R_t + \delta x)^2 + \delta y^2 + \delta z^2}$ [m] into the equations of motions, making a Taylor series expansion, and taking only the first order terms, the three dimensional CWH equations are obtained [49]:

$$\begin{aligned}\delta\ddot{x} - 3n^2\delta x - 2n\delta\dot{y} &= \frac{F_x}{m_c} = u_x\\ \delta\ddot{y} + 2n\delta\dot{x} &= \frac{F_y}{m_c} = u_y\\ \delta\ddot{z} - 3n^2\delta z &= \frac{F_z}{m_c} = u_z\end{aligned} \tag{2-5}$$

where $u_x, u_y,$ and u_z [m/s^2] are the components of the acceleration of the spacecraft in the $x, y,$ and z directions induced by the thrust forces $\mathbf{F} = [F_x, F_y, F_z]$. Some research groups are considering higher order terms of the Taylor series expansion in the hopes that more accurate modelling and less control effort are possible [40].

Despite the apparent simplicity of a system which these linearized equations of motion might suggest, the docking port may exhibit complex motion if the target is spinning, tumbling, or exhibiting unpredictable behaviour. A non-cooperative or incompletely characterized cooperative target may introduce additional difficulties to the rendezvous and/or docking processes as the precise position of the docking point is more complex to determine.

It should thus also be noted that RPO maneuvers are customarily subject to state and control constraints in order to aid and allow for successful docking procedures [20]. These constraints often include, but are not limited to:

- LOS constraint: in the Closing maneuvers, the chaser spacecraft and the docking or grasping point on the target must remain within the line-of-sight (LOS) of each other. This is often so that the chaser remains within the view of a docking port antenna used to identify the relative location of the docking point in real-time, however this is only useful in the case of a cooperative target with predictable dynamics.
- Soft-docking constraint: the velocity of the SC at rendezvous should match the velocity of the docking port.

It is also generally desirable that collisions with debris and other objects in the path of the chaser be avoided [21]. These requirements necessitate the consideration of the spacecraft state and control pointwise-in-time [20].

2.3 Control of RPO maneuvers

Relative motion maneuvers, like the SC rendezvous problem, have traditionally been performed using open loop maneuver planning with ad hoc error correction. More recently, research has been more focused on the application of model predictive control (MPC) to the control problem presented [22]. Sun et al. provide a brief survey of the steps taken from the early considerations of constraint in autonomous rendezvous and terminal rendezvous autonomous guidance laws, through disturbance handling, to parametric control methods and finally constrained MPC methods in [41].

More pointedly, though, it has been demonstrated in [15, 20, 21] that a spacecraft can be suitably controlled with MPC to approach rotating, non-rotating, and tumbling targets while avoiding debris. Robustness of conventional MPC to disturbances has also been evaluated. It has additionally been shown that if a spacecraft approaches a non-rotating target along a known LOS cone, an explicit MPC approach does not require on-board optimization. [22]

Further works and their advances relevant to RPO control by MPC include the following: In [22], the orbital plane was departed from, and the problem considered in 3 dimensions. The MPC implementations in [15, 20-22] consider situations where a set of initial and final conditions in relative space are set, and the controller provides the acceleration (or forces) required from the thrusters to optimally reach the docking port in a classic MPC regulation problem. Other papers propose a “rubber band” MPC approach, which makes use of a varying horizon.

Recently, Robust MPC methods have also been applied to the rendezvous problem. In [15], it was shown that conventional MPC is not capable of handling additive disturbances (see *section 4.1*) and introduces robust satisfaction of system constraints. Gavilan et al. used a *min-max worst-case disturbance* method and an estimate of the disturbance bounds to robustly satisfy constraints; producing a method robust to additive and multiplicative disturbances as well as unmodelled dynamics. In [24], Weiss and Di Cairano present an adaptive implementation of Robust Dual Control MPC, which guaranteed constraint satisfaction. This method re-identifies the system and updates the prediction model at each time step to define an up-to-date nominal problem and employed a robust dual-control MPC method.

In [37, 38], Tube-based robust MPC were applied to regulation of the rendezvous problem. In each of these paper, different uncertainty sources are investigated. In these two works, the satellites are taken to be travelling on elliptical orbits and the uncertainty was derived from navigation and thruster timing errors, respectively. In [37], some ideas taken were adopted from tube-based robust model predictive control to steer the chaser spacecraft from an initial relative state to a neighbourhood about a desired set point in the presence of navigation uncertainty. The uncertainty comes from error injected into the measurements of relative position by the sensor. In [38], tube-based robust model predictive control is used as a part of a guidance algorithm robust to thrusting errors to best guarantee precision performance under propulsion uncertainties.

The remainder of this review will first outline Model Predictive Control, then robust control elements will be introduced, finally leading up to the proposed control method of Tube-base Robust MPC.

3 Model Predictive Control

MPC can be thought of as a form of optimal control specifically for dynamic systems with the goal of robust stability. An MPC controller typically makes use of a linear or linearized model and its constraints to compute the optimal control sequence from the current conditions up to a finite horizon (at a future point in time). The first element of this sequence is applied to the system. The process repeats itself at each sampling instant. The model can, of course, be nonlinear, but this results in a more complex optimization process. [13, 50-53]

All model predictive control algorithms come in three parts: a prediction model, an objective function, and the formation of the control law [13]. The following sections will endeavour to express the concepts at the core and the mathematical basis of the MPC formulation.

3.1 The receding horizon

Before delving into the formal formulation of MPC, it is important to understand the concept of the receding horizon. Consider the following scenario [53]:

There are two joggers attempting to maintain a path between two lines painted on the ground. One of the joggers is jogging in reverse – that is, the jogger's back is facing the direction in which they are running – and the other is jogging facing forward.

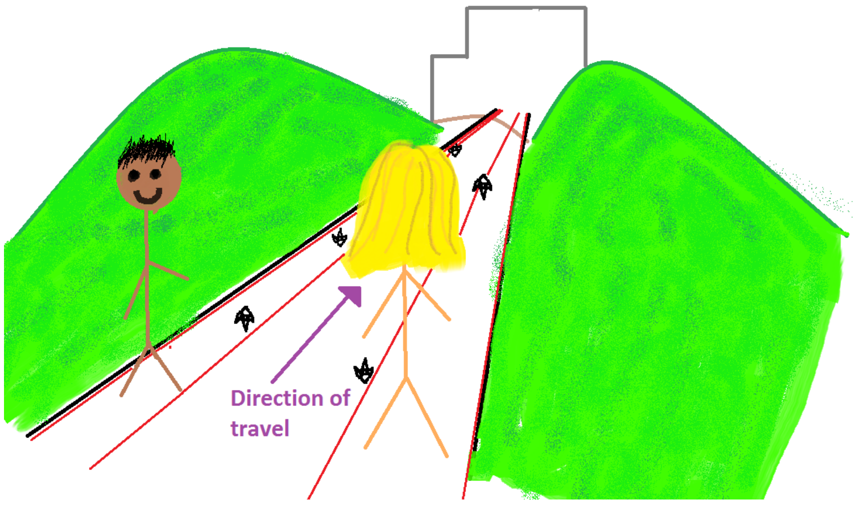


Figure 3-1 Jogger scenario

The forward-facing jogger has all the benefits of sensing the environment ahead of them – they can see the lines demarcating the path ahead, within which they must remain, and make decisions based on terrain and obstacles – while the reverse-facing jogger can only see where they currently are and what they have already past. The reverse-facing jogger essentially has two choices: the jogger can *(a)* always be extra cautious and therefore slow, or *(b)* go fast and handle any accidents as they occur. The forward-facing jogger, on the other hand, can approach the coming obstacles with the requisite speed and appropriate level of caution.

The forward-facing jogger seems the logically superior option. There is one caveat to note: the forward-facing jogger can only see so far into future circumstances; they will only know what is within their *horizon*. As the jogger moves forward, they will be able to see further ahead with respect to the starting point, but the same fixed-distance away.

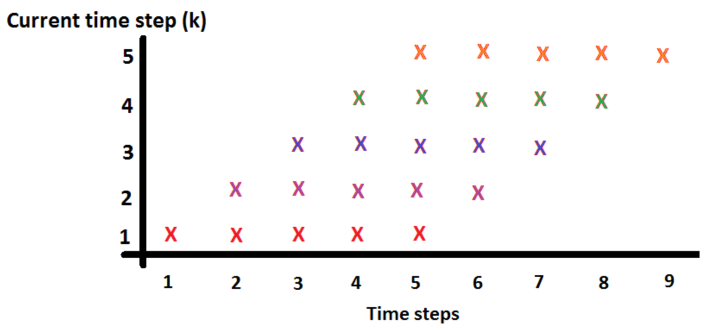


Figure 3-2 Receding horizon: horizon length 5 time steps

The horizon can be thought of as continuously moving forward as the jogger advances. This is called the *receding horizon*. The above figure illustrates this concept: at the current step, for example $k = 2$, the jogger can see only 5 steps in the future, or only as far as step 6.

Like the forward-facing jogger, MPC methods use future information up to a given horizon which recedes with time. This feature is an important aspect which sets MPC apart from other optimal and parametric control methods [53]. In some systems, for example a robotic manipulator, the future is well planned out, and it is thus intuitive to take advantage of this information when determining control actions. In other systems, future information may not be available, so some form of estimation of the future of the system must be used to assist in determining the control action.

3.2 Regulation vs Tracking

There are two umbrella forms MPC problem: *regulation* problems and *tracking* problems. In the *regulation* problem, the objective is to optimally steer the state(s) of the system with respect to some cost, or performance measure, to the origin while satisfying input, output, and state constraints at all times. The *tracking* problem takes on the engineering task of optimally tracking a given output reference trajectory while ensuring the constraints are satisfied. The problems operate under the basic assumptions that the system is controllable and the constraints satisfiable. [2, 14]

For ease of explanation, the control process will first be described in terms of the regulation problem. After this, the few necessary differences will be explained for reference tracking.

3.3 Prediction

The estimation mentioned above is the prediction part of MPC. In this step, a model of the system – including past and present error, disturbance, input and output parameters – are used to estimate the possible states of the system in the near future [53]. The accuracy of the model has a heavy influence on the closed loop performance of the control law. Or quite simply: misinformation leads to erroneous decision making.

3.3.1 Dynamic modelling

The model is the bedrock of MPC: the future response is predicted using a dynamic model of the system. The design of the model should be complete enough to capture the system dynamics and allow predictions of the future system state to be calculated, while also being intuitive and permitting analysis [13]. It is common to form the dynamic model using state space, as this is said to be the most intuitive way to model system dynamics. In some fields, though, it is common to define the model in the form of a transfer function [13, 14, 53]. For the intuitiveness in explanation and the prevalence in the field of RPO research, the state space representation will be presented here.

The major advantages of the state space formulation in MPC are related to the control stability, its usefulness in both mono- and multivariate systems, and that such a formulation can be easily extended to nonlinear processes [13].

In state space modelling, a system is represented by differential equations decomposed into a set of first order differential equations. Each system has a set of internal variables, or *states.* The states and the output of the system evolve with time as a function of the present values of the states and any present inputs. The representation in continuous time is of the form

$$\frac{d}{dt}x(t) = \mathbf{f}_x\big(x(t), u(t)\big)$$

$$y(t) = \mathbf{f}_{\mathbf{y}}(x(t), u(t)) \tag{3-1}$$

where $\mathbf{f}_x$ is the set of functions of the states $x(t) \in \mathbb{R}^n$ and inputs $u(t) \in \mathbb{R}^m$ and $y \in \mathbb{R}^{\mathbf{p}}, \mathbf{f}_y$ set of functions determining the outputs of the system, with t indicating the instant in time. The first equation is called the *state equation* and the second the *output equation*. These functions, in the undisturbed continuous time case, take the form

$$\frac{d}{dt}x(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) \tag{3-2}$$

where $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{\mathbf{n}\times\mathbf{m}}$, and $C \in \mathbb{R}^{\boldsymbol{p}\times\boldsymbol{n}}$ are the system, input, and output matrices, respectively.

While it is possible to model the system in continuous time, MPC is typically implemented digitally, and it is therefore more conducive to model the system in, or convert the model to, discrete time [14, 52]. In this case, the system takes the form

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) \tag{3-3}$$

Now, k indicates the time step under consideration and the matrices $A, B,$ and C are the discretized versions of their continuous time counterparts.

The prediction for this model has the form [13]

$$\tilde{y}(k+i|k) = C\tilde{x}(k+i|k) = C\left[A^i x(k) + \sum_{j=1}^{i} A^{j-1}Bu(k+i-j|k)\right] \tag{3-4}$$

Here the notation $\tilde{x}(k+i|k)$ indicates a predicted value $\tilde{x}$ for prediction step $(k+i)$ calculated at time step k. This prediction is part of the iterative prediction process which derives the estimated states and outputs on the prediction horizon, and will be revisited in *section 3.3.3.*

State space incorporates disturbances relatively simply. In a state space system, disturbances can occur at the input, output, along the measurement path, or in some internal variable of the system as illustrated in the following figure [53].

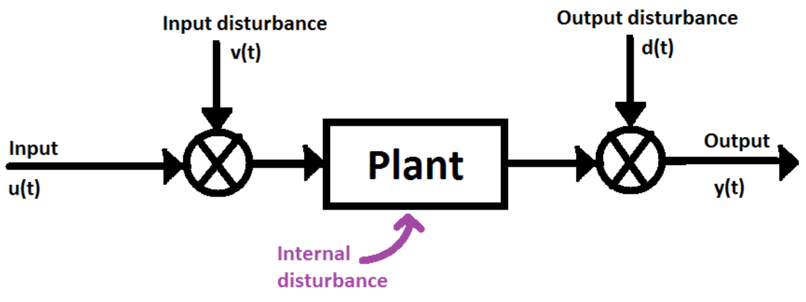


Figure 3-3 System disturbances

These disturbances can be included in the system model and estimated using an observer. A future disturbance takes the form

$$d(k+1) = d(k) + \gamma(k) \tag{3-5}$$

where $\gamma(k)$ is the variation of the disturbance from the current time step k to the future step $k+1$. The incorporation of the disturbance into the system model depends on the position in the system of the present disturbance. In this work, only additive or state disturbances are of interest and will be outlined in the following [53].

Let the state disturbance present be of the form $\mathbf{n}(k+1) = \mathbf{n}(k) + \gamma(k)$ within a system

$$x(k+1) = Ax(k) + Bu(k) + Ln(k)$$

$$y(k) = Cx(k) + Du(k) \tag{3-6}$$

When posed as a state, the disturbance can also be estimated, and the state space representation of the system becomes

$$\begin{bmatrix} x(k+1) \\ n(k+1) \end{bmatrix} = \begin{bmatrix} A & L \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ n(k) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \begin{bmatrix} u(k) \\ \gamma(k) \end{bmatrix}$$

$$y(k) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ n(k) \end{bmatrix} + Du(k) \tag{3-7}$$

3.3.2 Linearity of the model

There are, of course, a wide variety of forms that the system models may take. Systems can be linear or nonlinear, discrete or continuous in time. The prediction modelling may be deterministic, stochastic, or fuzzy. The rest of this section will distinguish the handling of linear from nonlinear system prediction.

Linear plant models

In a linear system, the predictions of the states $x(k)$ has a linear dependence on the control input $u(k)$ [52]. A quadratic prediction cost will then be a quadratic function of the control inputs. The input and state constraints linearly imply a constraint on the control inputs.

Nonlinear plant models

In a nonlinear system, there is a nonlinear dependence of the prediction of states on the control inputs. This results in a considerably more difficult optimization problem. In such a *nonlinear programming* problem, there is no general guarantee that the optimization solver will converge to a global extremum [13].

3.3.3 Prediction modelling

The prediction process is an iterative application of the plant dynamic model on the predicted or, if possible, measured control parameters in an effort to predict the possible future control parameter. Using the dynamic model characterized in the preceding, an estimation can be formed to predict the system's future states, disturbances, and control inputs. There are two main methods of prediction in the state space formulation. Camacho and Bordons explain the incremental model in [13], and with the aid of reference, explain a method based on (3-4). To explain this process, consider the following system:

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) \tag{3-8}$$

Of course, if disturbances are present, they must be taken into account by augmenting the system as described. For a constrained system, the constraints $x(k) \in \mathbb{X}$ and $u(k) \in \mathbb{U}$ are applicable to the states and inputs, respectively. The control constraint set $\mathbb{U} \subset \mathbb{R}^m$ is convex and compact, the state constraint set $\mathbb{X} \subset \mathbb{R}^n$ is convex and closed, and both contain the origin.

Now, prediction of the future state of the system is achieved by iteratively considering the state equation, starting with the initial state $x(k|k) = x(k)$ and with a terminal constraint $x(k+N|k) \in \mathbb{X}_f$, where $\mathbb{X}_f$ is the terminal set of the prediction. Given a sequence of predicted control inputs $\tilde{\mathbf{u}}(k)$, a sequence of predicted states $\tilde{\mathbf{x}}(k)$ can be generated by simulating the model forward over the *prediction horizon* of N sampling intervals. This series of predictions takes the following form [53]:

$$\tilde{x}(k+1|k) = Ax(k) + B\tilde{u}(k|k)$$

$$\tilde{x}(k+2|k) = A\tilde{x}(k+1|k) + B\tilde{u}(k+1|k) = A^2x(k) + AB\tilde{u}(k|k) + B\tilde{u}(k+1|k)$$

$$\tilde{x}(k+3|k) = A\tilde{x}(k+2|k) + B\tilde{u}(k+2|k) = A^3x(k) + A^2B\tilde{u}(k|k) + AB\tilde{u}(k+1|k) + B\tilde{u}(k+2|k)$$

$$\dots$$

$$\tilde{x}(k+N|k) = A\tilde{x}(k+N-1|k) + B\tilde{u}(k+N-1|k)$$

$$= A^Nx(k) + A^{N-1}B\tilde{u}(k|k) + A^{N-2}B\tilde{u}(k+1|k) + A^{N-3}B\tilde{u}(k+2|k) + \dots + A^{N-n_u-1}B\tilde{u}(k+n_u|k) \tag{3-9}$$

where n_u indicates the control horizon. In this notation, $(k+i|k)$ indicates a prediction of the value at time step $k+i$ made at time step k. In matrix form, the prediction progression becomes [14, 52, 53]

$$\begin{bmatrix} \tilde{x}(k+1|k) \\ \tilde{x}(k+2|k) \\ \tilde{x}(k+3|k) \\ \dots \\ \tilde{x}(k+N|k) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \dots \\ A^N \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & 0 & \dots & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \dots & A^{N-n_u-1}B \end{bmatrix} \begin{bmatrix} \tilde{u}(k|k) \\ \tilde{u}(k+1|k) \\ \tilde{u}(k+2|k) \\ \dots \\ \tilde{u}(k+n_u|k) \end{bmatrix} \tag{3-10}$$

The associated output predictions are similarly represented by

$$\begin{bmatrix} \tilde{y}(k+1|k) \\ \tilde{y}(k+2|k) \\ \tilde{y}(k+3|k) \\ \dots \\ \tilde{y}(k+N|k) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \dots \\ CA^N \end{bmatrix} x(k) + \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ CA^{N-1}B & CA^{N-2}B & CA^{N-3}B & \dots & CA^{N-n_u-1}B \end{bmatrix} \begin{bmatrix} \tilde{u}(k|k) \\ \tilde{u}(k+1|k) \\ \tilde{u}(k+2|k) \\ \dots \\ \tilde{u}(k+n_u|k) \end{bmatrix} \tag{3-11}$$

It is often desirable to cast the prediction in terms of the estimated state and the current and preceding control input. The output predictions are then of the form [13]

$$\begin{bmatrix} \tilde{y}(k+1|k) \\ \tilde{y}(k+2|k) \\ \dots \\ \tilde{y}(k+N|k) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \dots \\ CA^N \end{bmatrix} \tilde{x}(k) + \begin{bmatrix} CB \\ CA^2B \\ \dots \\ \sum_{i=0}^{N-1} CA^iB \end{bmatrix} u(k-1) + \begin{bmatrix} B & \dots & 0 \\ C(B+AB) & \dots & 0 \\ \dots & \ddots & \dots \\ \sum_{i=0}^{N-1} CA^iB & \dots & \sum_{i=0}^{N-n_u} CA^iB \end{bmatrix} u(k) \tag{3-12}$$

The second term, however has no effect on the optimization, as it does not contain the decision variable.

3.4 Objective function

As a precursor to the optimization process, a critical control aspect needs to be introduced: the objective or cost function. While each of the various MPC algorithms and many authors suggest different cost functions for obtaining the control law, the general idea is that the future output of the system within the horizon should follow a reference signal while the control effort to do so should be penalized.

Harkening back to the basis of MPC in optimal control, the general form of the cost function is as follows [2, 14]

$$V_N(\mathbf{x}(k), \mathbf{u}(k)) = \sum_{i=k}^{k+N-1} l(\tilde{x}(k+i|k), \tilde{u}(k+i|k)) + V_f(x(k+N|k)) \tag{3-13}$$

where the stage cost function $l(\cdot,\cdot)$ is a positive definite function

$$l(\tilde{x}(k+i|k), \tilde{u}(k+i|k)) = \|x_i\|_Q^2 + \|u_i\|_R^2 \tag{3-14}$$

satisfying $l(\mathbf{0}, \mathbf{0}) = 0^2$. The notation in the stage cost function $\left||x|\right|_Q^2 = x^TQx$ and $\left||u|\right|_R^2 = u^TRu$ indicates the squared weighted Euclidean norms, where x and u are the current state and control inputs, respectively. The second term in (3-13), $V_f(\cdot)$, is the terminal cost function and also positive definite. The terminal cost must satisfy $V_f(\mathbf{0}) = 0$ and is frequently of the form $V_f(x(k+n_y)) = \left||x(k+N|k)|\right|_P^2$, where P is the solution to the Riccati equation. When coupled with the typical linear model described in the preceding sections, this forms the basis of the so-called Linear-Quadratic optimal control problem (LQP). A regulator controller which makes use of the LQP is referred to a Linear Quadratic Regulator (LQR).

It can be noted that the implied reference in (3-13) and (3-14) is the origin. The deviation is then between the estimated (or measured) value and the desired reference of 0: that is $\|x_i\|_Q^2$ implies $\|x_i - 0\|_Q^2$, and so on. The regulation problem can therefore be thought of as a special case of reference tracking where the point to be tracked is the same for all points on the prediction horizon and the desired terminal state is the same for each prediction horizon. As the input or state values can come from estimation or measurement, the tilde in the notation can be dropped in future, and the notation for a non-zero reference will be made clear.

3.5 Optimization

Now that the cost function has been characterized, the optimization process can be undertaken. First, a few more properties of the system should be defined.

Let

$$\mathbf{\Phi}(i; x, \mathbf{u}) \tag{3-15}$$

be the solution of (3-8) at time i, controlled by a series of inputs $\mathbf{u}$ when the initial state at time 0 is x – i.e. $\mathbf{\Phi}(0; x, \mathbf{u}) = x$. Additionally, for a given state x, let the set of admissible control sequences $\mathbf{u}$ be denoted by

$$\mathcal{U}_N = \{\mathbf{u} | u_i \in \mathbb{U}, \mathbf{\Phi}(i; x, \mathbf{u}) \in \mathbb{X} \text{ for } i = 0,1, \dots, N-1, \mathbf{\Phi}(N; x, \mathbf{u}) \in \mathbb{X}_f\} \tag{3-16}$$

Furthermore, let the set of initial states for which the set of admissible control sequences is non-empty, known as the region of attraction of the controller, be defined by

$$\mathcal{X}_N = \{x | \mathcal{U}_N(x) \neq \emptyset\} \tag{3-17}$$

This is also equal to the domain of the value function $V_N^*(\cdot)$.

If the current state x is known, then the sequence of optimal predicted control inputs $\mathbf{u}^*(x)$ is obtained through the minimization of (3-13). This process is denoted by $\mathbb{P}_N(x)$:

$$V_N^*(x) = \min_{\mathbf{u}}\{V_N(x, \mathbf{u}) | \mathbf{u} \in \mathcal{U}_N(x)\} \tag{3-18}$$

$$\mathbf{u}^*(x) = \arg\min_{\mathbf{u}}\{V_N(x, \mathbf{u}) | \mathbf{u} \in \mathcal{U}_N(x)\} \tag{3-19}$$

The optimization problem $\mathbb{P}_N(x)$ is solved online at each sampling instant and the first element of $\mathbf{u}^*(x)$, $u_0^*(x)$, is applied to the system [2, 14].

The repeated execution of measuring the state, computing the prediction and optimal control input, and applying it to the plant is regarded as the implicit Model Predictive Control law $\kappa_N(\cdot)$ of the form $\kappa_N(x) = u_0^*(x)$ [2]. Under such a law, the closed loop dynamics of the system can be expressed by

$$x(k+1) = Ax(k) + B\kappa_{\mathbf{N}}(\mathrm{x})$$

$$y(k) = Cx(k) \tag{3-20}$$

3.6 Reference tracking problem

Now that the basis of MPC has been set, the tracking problem can be revisited. For the purposes of this section, assume that no disturbance or uncertainty is present. Following from [2].

3.6.1 The reference tracking problem

Let there exist a reference set point y_r with a corresponding terminal state (x_r, u_r), where the subscript r refers to the reference tracking versions of these variables. Consider first a prediction horizon of a single time step. To be able to track the reference signal without exhibiting an offset, there must exist a feasible terminal state $(x_r, u_r) \in \mathbb{X}\times\mathbb{U}$ for the reference set point y_r, satisfying

$$x_r(k+1) = Ax_r(k) + Bu_r(k)$$

$$y_r(k) = Cx_r(k) \tag{3-21}$$

where the matrices and vectors are the same quantities as previous and have the same dimensions as the regulation problem.

In a linear discrete-time system, the reference state is reachable without an offset

$$rank\begin{bmatrix} I-A & -B \\ C & 0 \end{bmatrix} = n + p \tag{3-22}$$

where n and p are the dimensions of the input and output vectors, respectively. That is to say, if this condition in not met, the system will not be able to reach the set point precisely, but some other stable point in the vicinity. However, if this condition is met, the terminal state is not necessarily unique for the given set point. A method commonly used to ensure a unique terminal state is arrived at is to compare the current system to a unique reference state. This artificial determination basically compares the current state and/or input to a reference value. For a single prediction step, this boils down to the following quadratic optimization is solved

$$(\mathrm{x}_r^*(k), \mathrm{u}_r^*(k)) = \arg \min_{\mathbf{x}_r(k), \boldsymbol{u}_r(k)} (\mathrm{x_r} - \mathrm{x_{rt}})^{\boldsymbol{T}} \mathbf{Q_r} (\mathrm{x_r} - \mathrm{x_{rt}}) + (\mathrm{u_r} - \mathrm{u_{rt}})^{\boldsymbol{T}} \mathbf{R_r} (\mathrm{u_r} - \mathrm{u_{rt}})$$

$$\mathbf{s.t.} \quad \boldsymbol{x_r}(k+1) = Ax_r(k) + Bu_r(k)$$

$$y_r(k) = Cx_r(k)$$

$$x_r(k) \in \mathbb{X}$$

$$u_{\boldsymbol{r}}(k) \in \mathbb{U} \tag{3-23}$$

where $\mathrm{u_{rt}}$ is the desired reference value. On the other hand, if the problem is rank deficient, a feasible steady state can be determined such that the output tracking error is minimised in the least squares sense by solving the quadratic program

$$(\mathrm{x}_r^*(k), \mathrm{u}_r^*(k)) = \arg \min_{\mathrm{x}_r(k), u_r(k)} (\mathrm{y_r}(k) - \mathrm{Cx_{rt}}(k))^T \mathrm{Q_r} (\mathrm{y_r}(k) - \mathrm{Cx_{rt}}(k))$$

$$\mathbf{s.t.} \quad \mathrm{x_r}(k+1) = \mathrm{Ax_r}(k) + \mathrm{Bu_r}(k)$$

$$\mathrm{y_r}(k) = \mathrm{Cx_r}(k)$$

$$\mathrm{x_r}(k) \in \mathbb{X}$$

$$\mathrm{u}_r(k) \in \mathbb{U} \tag{3-24}$$

3.6.2 Nominal MPC for tracking

The concept presented in the preceding can be extended to cover the full horizon. Assuming (3-22) is satisfied for the pair $(x_r(k, y_r(k)), u_r(k, y_r(k)))$ for the desired reference point, a tracking model predictive controller is realized by solving the modified optimal control problem $\mathbb{P}_N(\mathrm{x_r}(k), y_r)$

$$V_N^*(x_r, y_r) = \min_{\mathbf{u}}\{V_N(x_r, y_r, \mathbf{u_r}) |\ \mathbf{u_r} \in \mathcal{U}_N(x_r, y_r)\} \tag{3-25}$$

$$\mathbf{u_r^*}(x_r, y_r) = \arg\min_{\mathbf{u}}\{V_N(x_r, y_r, \mathbf{u_r}) | \mathbf{u_r} \in \mathcal{U}_N(x_r, \mathbf{y_r})\} \tag{3-26}$$

in which the cost function and the set of admissible states is now dependent on the reference point

$$V_N\big(\mathbf{x_r}(k), \mathbf{y_r}(k), \mathbf{u_r}(k)\big) = \sum_{i=0}^{N-1} l\big(x_r(k+i|k) - x_{rt}(k), \mathrm{u}(k+i|k) - u_{rt}(k)\big) + V_f(x_r(k+N|k) - x_{rt}(k)) \tag{3-27}$$

$$\mathcal{U}_N(\mathbf{x_r}(k), y_r(k)) = \{\mathbf{u} | u_i \in \mathbb{U}, \mathbf{\Phi}(i; x, \mathbf{u}) \in \mathbb{X} \text{ for } i = 0,1, \dots, N-1, \mathbf{\Phi}(N; x, \mathbf{u}) \in \mathbb{X}_f(y_r(k))\} \tag{3-28}$$

The terminal and stage cost functions are defined as previously.

The terminal set is also dependent on the set point value of the reference $\mathbb{X}_f(y_s)$. If the system is linear, this can simply be shifted from the origin using

$$\mathbb{X}_f(y_s) = \{x_r(y_s)\} \oplus \mathbb{X}_f \subset \mathbb{X} \tag{3-29}$$

The additional constraint that all states within the shifted terminal set must also be contained in $\mathbb{X}$ limits the set points that can be tracked to

$$\mathcal{Y}_s = \{y_s | x_r(y_s) \oplus \mathbb{X}_f \in \mathbb{X}, \qquad \mathbf{u_r}(y_s) \in \mathbb{U}\} \tag{3-30}$$

For this set of admissible set points, the region of attraction of the tracking controller is

$$\mathcal{X}_N(\mathrm{y_s}) = \{x \mid \mathcal{U}_\mathrm{N}(\mathrm{x}, \mathrm{y_s}) \neq \emptyset\} \tag{3-31}$$

Employing the receding horizon and applying only the first element of the predicted sequence of control inputs, the implicit MPC law is given by

$$\boldsymbol{\kappa}_\mathbf{N}(x, y_r) = \mathbf{u}^*(x, y_r) \tag{3-32}$$

It is often not possible to simply shift the set point reference from the origin without introducing offset to the terminal state. However, when the method is applied, constraint (3-29) must be satisfied. This requirement may result in a small region of admissible steady states. If the size of $\mathbb{X}_f$ is limited by the state constraints, then constraint (3-29) can only be satisfied for steady states $\mathbf{x_s}$ close to the origin.

3.6.3 Offset problem in the presence of uncertainty

In the previous section, disturbances and errors were ignored. In this nominal case, if a feasible steady state exists for the given output setpoint, offset-free tracking is achieved. However, in real-world applications, perfect models do not exist and the system will always be subject to external disturbances. If there is a non-vanishing disturbance present in the environment of the system, the conventional MPC methods described so far – both regulator and tracking – will generally exhibit an offset from the desired outputs.[2]

The predominant approach to handling this offset is to augment the system states with fictitious integrating disturbances (see [2] and the references therein). If the external disturbance is asymptotically constant, then under the correct conditions, this method will allow offset-free MPC to be achieved.

3.7 Implementation of the receding horizon

In conventional MPC, the prediction horizon N has a static length, but receding as explained previously in *section 3.1*. As the state prediction and optimal input sequence depend on the current state measurement, feedback is introduced into the MPC law [52]. This affords a degree of robustness to modelling errors and uncertainty and implies that MPC is an inherently closed loop control method. Furthermore, the advance of the horizon with the current sample point compensates for the truncated infinite horizon, making it seem as if it were infinite.

The linear feedback form of the problem is anticipated in the expected LQ problem. The classic definition of the LQP specifies an infinite horizon. This is, however, impractical in a real-world application where a process has a finite duration. The horizon is truncated to a finite length and becomes the indicated finite prediction horizon N.

Furthermore, in the optimal control definition of the problem, there is no difference between the optimal predicted input sequence and the receding horizon in the absence of disturbances and model errors. Here, though, there can be significant discrepancy between the prediction values and the closed loop responses. This discrepancy increases with reducing horizon length. The horizon, therefore, needs to be long enough to return predictions as close to the closed loop response as possible while still short enough to be feasible. [52]

3.7.1 Prediction horizon selection

The constraints on the horizon may seem rough and there does not appear to be any set rule on how to select the prediction horizon. The recommended practice is to select the prediction horizon size early in the design and to leave it fixed. The size of this horizon should not be used as a tuning parameter. Instead, increasing the sample time or cost function weights or modifying the control horizon or terminal weights should be considered [54].

3.7.2 Control horizon selection

It is also prudent to outline the guidelines for selecting the control horizon n_u - the number of control inputs to be optimized at the current sample step. Recall that regardless of the size of this horizon, only the first element in the optimized sequence will be applied to the system and the rest are ignored. Typically, n_u is chosen so that the $2 \leq n_u \ll N$, for the following reasons [54]:

- Choosing a smaller n_u means that fewer future control values need to be optimized at each step, possibly drastically reducing the computational effort at each step.
- A smaller n_u promotes, but does not guarantee, an internally stable controller.
- If delays are present in the plant model, a small n_u is imperative as some control inputs might not have time before the end of the prediction horizon to apply any affect to the plant outputs. This would result in a singular Hessian matrix in the quadratic problem – i.e. the minimization of the cost function. A small n_u means that fewer computations must be conducted at each time step, resulting in a faster response.

3.8 Sampling time

As shown in the preceding section, the receding horizon implementation is dependent on the selection of the sampling time. As in the case of the prediction horizon, it is recommended to select the sampling time T_s early in the design of the controller and hold the value constant; unlike the prediction horizon, once the initial design steps are undertaken, this is relaxed and T_s becomes a tuning parameter [54]. If the parameter changes, though, other parameters are also likely to need re-tuning. Decreasing the size of T_s may yield better disturbance rejection characteristics in the controller. However, there is a limit to the improvement in disturbance rejection, and eventually reducing the sampling time further will make no difference to this characteristic in the system response. The value of T_s at which performance in terms of disturbance rejection plateaus is dependent on the dynamic characteristics of the plant. It should also be kept in mind that as T_s decreases, the computational effort increases. For process control, $T_s \gg 1\ s$ is common; in other applications, like aerospace, circumstances may call for $T_s < 1\ s$ [54].

3.9 Constraint handling

In practice, all processes are subject to constraints. For example, actuators have a limited range of action and have a limited slew rate, or thrusters have a limited range of force, saturating at each end of the range. Constraints come in two major umbrella types [13, 52, 53]: *equality* and *inequality* constraints. In MPC, equality constraints typically are those which state that the states and control inputs should satisfy model dynamics and are handled implicitly; inequality constraints on the control inputs and the states are imposed explicitly by the online optimization problem. Only the inequality constraints need be briefly enumerated, following [52, 53].

3.9.1 Hard and soft constraints

All constraints are either hard or soft. A *hard constraint* is one which must always be satisfied. If a hard constraint is not feasible, then the problem is infeasible. A *soft constraint* is one which may be violated if necessary so as to permit the problem to remain feasible.

3.9.2 Common constraints

Input saturation constraints

Saturation refers to the inability of the magnitude of a signal to go beyond the minimum or maximum of the permissible range. Input saturation is reached at the boundaries of the input constraint: $\mathbf{u}_{min} \leq \mathbf{u}(k) \leq \mathbf{u}_{max}$.

Input slew rates

In this case, slewing refers to the rate of change of the input. Input slew is therefore the input rate constraint: $\boldsymbol{\Delta}\mathbf{u}_{\boldsymbol{min}} \leq \mathbf{u}(\boldsymbol{k}) - \mathbf{u}(\boldsymbol{k}-\mathbf{1}) \leq \boldsymbol{\Delta}\mathbf{u}_{\boldsymbol{max}}$.

State constraints

State constraints ensure that the states are constrained to the region of attraction. Linear state constraints have the general form $\mathbf{x}_{\mathbf{min}} \leq \mathbf{x} \leq \mathbf{x}_{\mathbf{max}}$ and are active during transient and steady state operation.

Output constraints: saturation, overshoot, and monotonicity

Output saturation is much like input saturation, placing upper and lower bounds on the system output, and are of the form $\mathbf{y}_{\mathbf{min}} \leq \mathbf{y} \leq \mathbf{y}_{\mathbf{max}}$. Constraining the output can be used to prevent overshoot in the system response; while setting either the upper of lower limit to zero results in the output being monotonic.

3.9.3 Incorporating input and state constraints

To incorporate linear input and state constraints into the control law, they must be re-written in the form $\mathbf{A}_{\mathbf{c}}\mathbf{u} \leq \mathbf{b}_{\mathbf{0}} + \mathbf{B}_{\mathbf{x}}\mathbf{x}(k)$ which will be more suitable for inclusion into the optimization problem. To explain this, first consider the example of the input constraints.

The first step is to separate the constraint into its upper and lower constraints:

$$\mathbf{u}_{\mathbf{min}} \leq \mathbf{u} \tag{3-33}$$

$$\mathbf{u} \leq \mathbf{u}_{\mathbf{max}} \tag{3-34}$$

Reorganizing this into matrix form, the constraint becomes

$$\begin{bmatrix} -\mathbf{I} & \mathbf{u}_{\min} \\ \mathbf{I} & -\mathbf{u}_{\max} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{1} \end{bmatrix} \leq \mathbf{0} \tag{3-35}$$

For the state constraints, first consider the state to be of the form $\mathbf{x} = \mathbf{\Theta u} + \theta$, or the input plus some variation. Then, in matrix form the constraint becomes

$$\begin{bmatrix} -\mathbf{\Theta} \\ \mathbf{\Theta} \end{bmatrix} \mathbf{u} \leq \begin{bmatrix} -\mathbf{x}_{\mathbf{min}} \\ \mathbf{x}_{\mathbf{max}} \end{bmatrix} + \begin{bmatrix} \theta \\ -\theta \end{bmatrix} = \begin{bmatrix} -\mathbf{x}_{\mathbf{min}} \\ \mathbf{x}_{\mathbf{max}} \end{bmatrix} + \begin{bmatrix} \mathbf{A}^{i} \\ -\mathbf{A}^{i} \end{bmatrix} x(k) \tag{3-36}$$

Combining (3-35) and (3-36) gives the constraints expressed in the desired form,

$$\mathbf{A_c u} \leq \mathbf{b_0} + \mathbf{B_x x}(k) \tag{3-37}$$

where the constant matrices $\mathbf{A_c}$, $\mathbf{b_0}$, and $\mathbf{B_x}$ can be determined offline.

The optimization problem then becomes the minimization described in (3-18)-(3-19) or (3-25)-(3-26) subject to (3-37).

4 Tube-based Robust Model Predictive Control

The preceding chapter discussed conventional, nominal MPC, which provides strong theoretical results pertaining to nominal stability and feasibility. Nominal MPC, however, does not consider what happens when the predicted evolution of the system differs from the actual system behaviour due to disturbances or modelling errors. Robust MPC builds on nominal MPC to account for these. Open loop frameworks of robust MPC methodologies are generally overly conservative while the closed loop predictions often result in a large spread of predicted trajectories resulting in a controller with a relatively high computational complexity [2]. A formulation of robust MPC which aims to reduce this online complexity is tube-based robust MPC.

Tube-based robust MPC was first proposed by Langson et al. in [25], but [26] by Mayne et al. is widely considered to be the formal birth of the framework. The work conducted by these groups centers on the robust state-feedback regulation problem. It can be understood that Tube-based robust MPC separates the constrained optimal control problem from the problem of ensuring robustness to uncertainty, while the underlying concept of tubes permits extension of the methodology to other problem types [2]. Further work was conducted by Alvarado et al. [31, 32], Limon et al. [3, 33], and Mayne et al. [27, 28, 34, 35] expanding the concept into robust output-feedback, tracking, and nonlinear problems.

Tube-base robust MPC builds on the theory of robust positively invariant (RPI) sets with the aim to solve the nominal MPC problem whilst constraining the discrepancy between the nominal and uncertain system states. By solving this problem under suitably tightened state and control constraints, the uncertain system can be guaranteed to evolve within a tube of trajectories centered on the predicted or reference nominal trajectory. The nominal system should then be controllable in such a way that the original constraints are satisfied at all times [2]. It is important to make the distinction that the nominal system refers here to the unperturbed plant dynamics with no uncertainty or error present.

Amongst the various derivative forms of linear Tube-based robust MPC, the common feature is the form of the control law

$$u = v + K_{dr}(x - z) \tag{4-1}$$

The control law is an LTI feedback controller, where the second term is a disturbance rejection controller with disturbance rejection gain K_{dr} and a bounded deviation of the real system x from the nominal z. This boundary is a robust positively invariant set $\mathcal{Z}$, the selected size and shape of which influences the adjustment of the original constraints $\mathbb{X}$ and $\mathbb{U}$ and the terminal set $\mathbb{X}_f$. The implementation then becomes a slightly modified conventional MPC for the nominal system [2, 26], accompanied by a cluster of additional parameters which permit the reduction in online computational effort. This reduction in complexity is a major selling point to Tube-based robust MPC.

In the succeeding sections, the remaining necessary background theory is outlined, followed by the linear regulation framework as was initially proposed. Subsequently, the state-feedback methodology extensions for tracking is as needed for tracking a reference trajectory. This exposition follows mainly from [1-4, 25, 26, 29, 31-33].

4.1 Uncertainty modelling

No mathematical model of a real process is able to plan for every aspect of reality. The most straight-forward approach to handling uncertainties presented in literature is to rely on the inherent robustness of model predictive control and simply ignore any uncertainty or error [14, 25, 55]. Remember from *section 3.7* that the receding horizon control strategy introduces feedback into the controller through the cyclic prediction and optimization, which leads to a degree of robustness against perturbation, even if the controller has not been explicitly designed for this. It would make sense that this quality would also be present in model predictive control methods. However, the constrained nature of the model predictive control strategy presented in the previous chapter and the implicit form of the control law make robustness and stability analysis a very difficult task [2, 56]. There are stark few approaches to analysing nominal MPC robustness and stability presented in literature [2].

Clearly, then, simply ignoring uncertainty and error in MPC problem formulation is generally a bad idea. In order to account for the discrepancies between the model and the real response, it is necessary to obtain an adequate model of the uncertainty. The way in which the uncertainty is modelled often depends on the technique used to design the controller [13]. In this work, only bounded additive input disturbances are relevant.

In the case of bounded additive input disturbances, the system model is taken to be accurate, but with unknown bounded disturbances acting upon it as follows

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w \\ y(k) &= Cx(k) \end{aligned} \tag{4-2}$$

The state disturbance w directly affects the evolution of the states and includes external disturbances, parameter uncertainty, and unmodelled dynamics. The set $\mathcal{W}$ containing these disturbances are most commonly taken as polytopic. While this is not a strict requirement, polytopic sets can result in a lower computational complexity and cost.

4.2 Robust positively invariant sets

Robust positively and control invariant sets play an important role in robust MPC, appearing in numerous robust methods, see [2, 15, 25, 41, 56] for examples. The actual use of the term RPI has a tendency to vary from author to author. However, in the field of Tube-based robust MPC, the use is consistent in meaning disturbance invariant [2]. This implies that the set is invariant to the realization of the bound disturbance. In this section, the concept of these types of sets will be defined.

Consider a system which is acted upon by an exogenous disturbance

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{4-3}$$

where the disturbance $w \in \mathcal{W}$ and $\mathcal{W}$ is a compact set. Then, a set Ω is robust positively invariant for the system if for every initial state $x(0) \in \Omega$ and all $w \in \mathcal{W}$ the solution at $k > 0$ is $x(k) \in \Omega$ [57]. The minimal robust positively invariant (mRPI) set $\mathcal{F}_\infty$ is an RPI set contained in every closed RPI set of system (4-3), that is $F_\infty \subseteq \Omega$ [58]. The exact set F_∞ is only determinable under certain circumstances. An approximation $\mathcal{Z}$ of the mRPI set is made in practice. It is advantageous, when the system constraints are defined as polytopes, to determine the mRPI in such a way that it is also polytopic. This places certain demands on its determination [26], which will explored later in *section 5.2.2*.

4.3 Tube-based robust MPC for regulation

The solution of a robust MPC optimization problem is dependent on the specific realization of the generic uncertainty. Tube-based robust MPC is motivated by the observation that both the open- and closed loop formulations of the robust control problem in the presence of uncertainty generate a tube of trajectories where each trajectory in the bundle corresponds to a particular realization of the uncertainty [14], illustrated by the following.

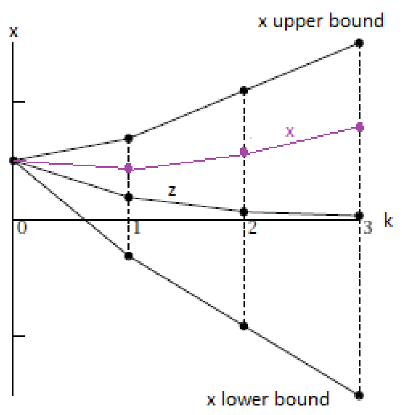


Figure 4-1 State trajectories and state tube, adapted from [14]

In Figure 4-1, certain trajectories of a one-dimensional system are emulated for 3 time steps. The central trajectory, labelled z, corresponds to the nominal trajectory along which the system would evolve with no disturbance present, the extreme plots are the upper and lower bounding trajectories of the bundle of trajectories, and the purple trajectory, labelled x, is the trajectory corresponding to some realization of the system uncertainty. It is important to keep in mind that in the robust framework the state and control constraints must be satisfied at all times by every trajectory within the bundle.

Tube-base robust MPC is a compromise between optimality and simplicity. The uncertain system control problem can be thought of as the nominal model predictive control of a tube, rather than individual trajectories. The center of the tube will be coincident with the nominal system response, when the disturbance is absent. The boundary of the tube then encloses the collective bundle of all trajectories satisfying all of the constraints imposed in the optimization procedure, discussed in the subsequent sections. Through suitable design of the tube, satisfaction of the constraints can be guaranteed for every disturbance realization which will yield a trajectory residing within this bundle [14]. This design process will be discussed later in this chapter and demonstrated in the next chapter.

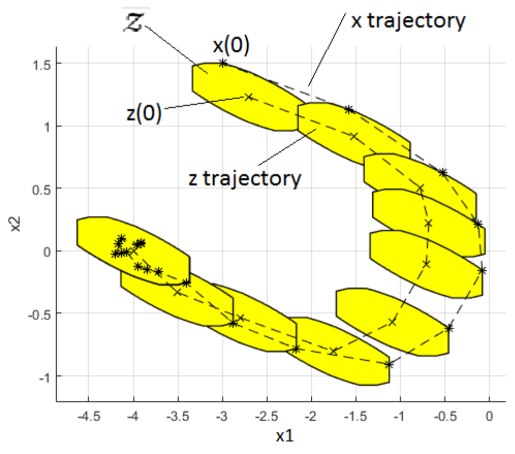


Figure 4-2 Outer bounding tube centered on nominal response

The entire tube, an example of which is illustrated above, need not be considered for each prediction step. As a result of the design process discussed later, the outer boundary of the tube is known at each time step. The step-wise consideration of this boundary means that at the end of the control process the boundary of the entire tube will have been accounted for.

The idea behind Tube-based robust MPC for regulation is then fairly simple: the nominal trajectory at the center of the tube is found using conventional MPC as described in *chapter 3* using appropriately tightened constraints and restricting the size of the tube using feedback elements to steer all trajectories within the tube to the nominal [14]. The resultant controller is in two degrees of freedom – the first DOF consisting of the conventional MPC control in an inner loop called the *nominal controller* and the second a *disturbance rejection controller* in an outer loop restricting the deviation of the actual trajectory from the nominal. The rest of this section is dedicated to the description of this framework following from [2, 25-30].

4.3.1 The system definition

Consider the constrained linear, discrete-time system (DLTI) (4-3), which is simplified here for ease of exposition to

$$x^+ = Ax + Bu + w$$
$$y = Cx \tag{4-4}$$

where $x \in \mathbb{R}^n$ is the current system state, $u \in \mathbb{R}^m$ the control action, x^+ is the successor state, and y is the system output. The unknown, bound disturbances $w \in \mathbb{R}^n$ are contained in the convex and compact set $W \subset \mathbb{R}^n$, and $(A, B) \in \mathbb{R}^{n\times n}\times\mathbb{R}^{n\times m}$ is assumed controllable. The system is subject to hard system and control constraints

$$x \in \mathbb{X} \subseteq \mathbb{R}^n, \qquad u \in \mathbb{U} \subseteq \mathbb{R}^m \tag{4-5}$$

which are polyhedral and polytopic sets, respectively, and both contain the origin. The corresponding nominal system is

$$z^+ = Az + Bv \tag{4-6}$$

where $z \in \mathbb{R}^n$ is the current nominal system state, $v \in \mathbb{R}^m$ the current control action, and z^+ the successive nominal state. The nominal control sequence is of the form $\mathbf{v} = \{v_0, v_1, \dots, v_{N-1}\}$ and the predicted nominal states $\mathbf{z} = \{z_0, z_1, \dots, z_{N-1}\}$. Furthermore, the control input and disturbances are the sequences $\mathbf{u} = \{u_0, u_1, \dots, u_{N-1}\}$ and $\mathbf{w} = \{w_0, w_1, \dots, w_{N-1}\}$, respectively, and $\mathbf{\Phi}(i; x, \mathbf{u}, \mathbf{w})$ is the solution of (4-4) at time step i controlled by $\mathbf{u}$ when the initial state at $i = 0$ is x.

4.3.2 The robust control strategy

As mentioned previously, the control strategy is two-fold: a feedforward conventional MPC element for the control input computed for the nominal problem and an ancillary linear feedback controller acting on the discrepancy between the actual state x and predicted nominal state z. Let this error be signified by

$$e = x - z \tag{4-7}$$

and the feedback controller has the form

$$u = v + K_{dr}e \tag{4-8}$$

where $K_{dr} \in \mathbb{R}^{m\times n}$ is a linear disturbance rejection controller and is chosen such that $A_k = A + BK_{dr}$ is stable [26].

Now, let $\Omega \in \mathbb{R}^n$ be an RPI set for the perturbed system

$$x^+ = A_k x + w \tag{4-9}$$

such that

$$A_K\mathcal{Z}\oplus W \subseteq \mathcal{Z} \subseteq \Omega \tag{4-10}$$

is satisfied. Recall from the introduction of this chapter that $\mathcal{Z}$ represents the RPI set which defines the tube boundary. The operator $\oplus$ indicates the Minkowski set addition. Then, if the current perturbed system state

$$x \in \{z\}\oplus\mathcal{Z} \tag{4-11}$$

and control law (4-8) is applied, it follows that the successor state $x^+ \in \{z^+\}\oplus\mathcal{Z}$ for all admissible disturbance sequences [26]. This implies that if this control law is employed, the uncertain system states $x(i) = \mathbf{\Phi}(i; x, \mathbf{u}, \mathbf{w})$ will be kept close to the predicted states $z(i) = \bar{\mathbf{\Phi}}(i; z, \mathbf{v})$ of the nominal system for all admissible disturbances w and that, if the nominal control problem is solved with tightened constraints, the consistent constraint satisfaction for the uncertain system is ensured.

It is advisable to choose $\mathcal{Z}$ as small as possible so as to reduce the conservativeness of the problem. It would also make sense to choose the disturbance rejection gain K_{dr} to be large, however that would result in a large mapping of $K_{dr}\mathcal{Z}$ and therefore a small tightened constraint set $\bar{\mathbb{U}}$. There is thus a design trade-off in the determination of $\mathcal{Z}$ and K_{dr}.

4.3.3 The cost function

The cost function of the nominal problem has the same general form as the cost function outlined in *section 3.4.* Define the cost function for a trajectory of the nominal system thus as

$$V_N(z, \mathbf{v}) = \sum_{i=0}^{N-1} l(z_i, v_i) + V_f(z_N) \tag{4-12}$$

with the stage cost

$$l(z_i, v_i) = \left||z_i|\right|_Q^2 + \left||u_i|\right|_R^2 \tag{4-13}$$

and terminal cost

$$V_f(z_N) = \left||z_N|\right|_P^2 \tag{4-14}$$

where $Q, R,$ and P are positive definite weighting matrices.

The terminal cost function and constraint sets are commonly chosen to satisfy typical MPC stability assumptions. As such, the following characteristics hold. The tightened terminal constraint set $\mathbb{X}_f$ is a constraint admissible RPI set $\mathbb{X}_f \subset \bar{\mathbb{X}}$ for the closed loop system $z^+ = Az + B\kappa_f(z)$, and it holds that $z^+ \in \mathbb{X}_f$ and $\kappa_f(z) \in \bar{\mathbb{U}}$ for all $z \in \mathbb{X}_f$. By requiring the terminal set to be contained in the set of tightened state constraints ensures that the terminal state x_f of the uncertain trajectory is also contained within the tightened constraint set. Now, the terminal cost is defined by a local control Lyapunov function z^+, satisfying the condition

$$V_f\left(Az + B\kappa_f(z)\right) + l\left(z, \kappa_f(z)\right) \leq V_f(z), \forall z \in \mathbb{X}_f \tag{4-15}$$

The subject of the function V_f in the first term is the successive nominal state z^+. This condition means that the cost of the terminal state in the next step will be smaller than the terminal state cost of the current state and the system driven toward an equilibrium point of minimum cost. It should also be noted that the infinite horizon controller κ_f and the disturbance rejection controller K_{dr} need not necessarily be the same, but since K_{dr} is intended to provide disturbance rejection and not to yield an optimal cost, it can be optimized with respect to κ_f.

4.3.4 The nominal optimal control problem

Now, let the initial state of the nominal system coincide with the actual system state, $z_0 = x_0$. Then, the conventional optimization problem for the nominal controller $\mathbb{P}_N^{r0}(x)$ is as follows

$$V_N^{r0}(x) = \min_{\mathbf{v}}\{V_N(x, \mathbf{v}) \mid \mathbf{v} \in \boldsymbol{\mathcal{U}}_N(\boldsymbol{x})\} \tag{4-16}$$

$$\mathbf{v^{r0}}(\boldsymbol{x}) = \arg\min_{\mathbf{v}}\{V_N(x, \mathbf{v}) \mid \mathbf{v} \in \boldsymbol{\mathcal{U}}_N(x)\} \tag{4-17}$$

with

$$\mathcal{U}_N(x) = \left\{\mathbf{v} \mid v_i \in \bar{\mathbb{U}}, \bar{\mathbf{\Phi}}(i; x, \mathbf{v}) \in \bar{\mathbb{X}}\, for\ i = 0,1, \dots, N-1, \bar{\mathbf{\Phi}}(N; x, \mathbf{v}) \in \mathbb{X}_f\right\} \tag{4-18}$$

and a region of attraction

$$\bar{\mathcal{X}}_N = \{x \mid \mathcal{U}_N(x) \neq \emptyset\} \tag{4-19}$$

As in conventional MPC, the solution to this optimization problem are sequences of predicted optimal control actions and predicted optimal state trajectory $\mathbf{v^{r0}}(x) = \{v_0^0(x), v_1^0(x), \dots, v_{N-1}^0(x)\}$ and $z^{r0}(x) = \{z_0^0(x), z_1^0(x), \dots, z_{N-1}^0(x)\}$, and $z_i^{r0}(x) = \bar{\mathbf{\Phi}}(i; x, \mathbf{v^{r0}}(x)$. The implicit state-feedback control law for the closed loop nominal system $z^+ = Az + B\kappa_N^{r0}(z)$ is of the form

$$\kappa_N^{r0}(x) = v_0^{r0}(x) \tag{4-20}$$

4.3.5 The robust controller

The conventional state-feedback formulation of MPC suggests that a state-feedback robust controller would be the simplest option for regulation. While further work has derived other frameworks for the Tube-based robust regulator, only the state-feedback version will be considered now.

In the preceding, it was assumed that the initial state of the nominal system is the same as the current state of the actual system. This is, however, not required. The control strategy only enforces the actual and predicted states to be “close”. The constraint expressed in (4-11) describes close to be within the bound of the RPI set super-imposed on the nominal state. It is also not necessarily true that the cost (4-12) does not decrease along the actual uncertain state trajectory as required by (4-15). This means that is not possible to control the uncertain system using the conventional model predictive controller outlined in the previous chapter as it would not be possible to establish an exponentially stable robust set Ω. Exponential stability is one of the metrics for assessing stability of model predictive controllers, mentioned in *section 4.1.*

Thus, these preceding assumptions are departed from, allowing for a different nominal initial state to be chosen by casting it as a decision variable z_0 [26]. This is permissible as the states of the nominal system do not have an immediate effect on the actual disturbed system. The rest of this section is devoted to retooling the optimization problem to incorporate the new decision variable.

The modified optimal control problem $\mathbb{P}_N^{r*}(x)$ is, at its base, of the same general form as the conventional problem, while including the new decision variable:

$$V_N^{r*} = \min_{z_0,\mathbf{v}}\{V_N(z_0,\mathbf{v}) \mid \mathbf{v} \in \mathcal{U}_N(z_0), z_0 \in \{x\}\oplus(-\mathcal{Z})\} \tag{4-21}$$

$$\left(z_0^{r*}(x), \mathbf{v}^{\mathbf{r}*}(x)\right) = \arg\min_{z_0,\mathbf{v}}\{V_N(z_0,\mathbf{v}) \mid \mathbf{v} \in \mathcal{U}_N(z_0), z_0 \in \{x\}\oplus(-\mathcal{Z})\} \tag{4-22}$$

where this additional variable is constrained by

$$z_0 \in \{x\}\oplus(-\mathcal{Z}) \tag{4-23}$$

Note that the original and tightened state and control constraints still apply to this controller, unchanged from the nominal problem. Further, as these and the RPI set $\mathcal{Z}$ are all polytopic and convex, they can be expressed as a finite set of linear inequality constraints and applied as briefed in *section 3.9*.

Now, similarly to the preceding problems, the predicted optimal control sequence is $\mathbf{v}^{r*}(x) = \{v_0^{r*}(x), v_1^{r*}(x), \dots, v_{N-1}^{r*}(x)\}$, the predicted optimal nominal state trajectory $\mathbf{z}^{\mathbf{r}*}(x) = \{z_0^{r*}(x), x_1^{r*}(x), \dots, x_{N-1}^{r*}(x)\}$ obtained in the solution of the optimization problem $\mathbb{P}_N^{r*}(x)$ such that $z_i^{r*} = \bar{\mathbf{\Phi}}(i; z_0^{r*}, \mathbf{v}^{r*}(x)$. The region of attraction for this problem is given by

$$\mathcal{X}_N = \{x \mid \exists\, z_0 \; such\ that\ z_0 \in \{x\}\oplus(-\mathcal{Z}), \mathcal{U}_N(z_0) \neq \emptyset\} \tag{4-24}$$

Finally, the modified implicit state-feedback MPC law as applied to the uncertain system at the current state x is of the form

$$\kappa_N^{r*}(x) = v_0^{r*} + K_{dr}(x - z_0^{r*}(x)) \tag{4-25}$$

The controller clearly relies on nominal states and control actions as predicted online by the optimization problem at each time step. Additionally, as the decision variable nominal state is in this case generally different from the actual current state, the applied control move $\kappa_N^{r*}$is also generally different from the first control action in the sequence $\mathbf{v}^{r*}(x)$, as indicated by the inclusion of the additional feedback term $K_{dr}(x - z_0^{r*}(x))$. This term is responsible for counteracting the influence of the disturbance sequence to drive the current actual system state x toward the predicted nominal system. The system trajectory is thus maintained within the sequence of sets

$$\{\mathcal{F}(0), \mathcal{F}(1), \mathcal{F}(2), \dots\} \tag{4-26}$$

$$\mathcal{F}(k) = z_0^{r*}\big(x(k)\big) \oplus \mathcal{Z} \tag{4-27}$$

which describes the desired tube of trajectories.

4.4 The state-feedback tube-based robust MPC for tracking

As motivated for engineering reasons, the tracking problem is of greater relevance to this work, and will be the topic of the remainder of this review.

The first problem in the tracking of a reference is to determine a suitable set point. This could be done using the optimization problem presented in *section 3.6,* however there remains the same problem as in the regulation controller that the conventional tracking MPC controller will be unable to handle the uncertain system. A further difficulty is introduced if the target point is arbitrarily shifted off of the origin, which requires the terminal set to be re-computed for each new steady state in order to ensure feasibility. The work by Alvarado et al. [31, 32] and Limon et al. [3, 4, 33] developed a Tube-based robust model predictive controller for tracking which moves this determination online and introduced an artificial steady state as an additional optimization problem decision variable. The tracking offset is limited by an additional penalizing term in the optimization problem and the terminal set re-computation is avoided using an invariant set for tracking [32].

The state-feedback case for tracking will be outlined in this section, following from [2-4, 31-33]. A practical implementation is conducted for a sample problem in the next chapter to describe how this theory is put to use.

4.4.1 The system definition

Consider for this problem the system modelled by the following

$$x^+ = Ax + Bu + w$$

$$y = Cx + Du \tag{4-28}$$

where, as before, $x \in \mathbb{R}^n$ are the current perturbed system states, $u \in \mathbb{R}^m$ are the control actions, $w \in \mathbb{R}^n$ are the unknown, bound disturbances acting on the system input, $y \in \mathbb{R}^p$ are the system outputs, and x^+ is the successor state. The matrices $A, B, C,$ and D are the appropriately dimensioned state, input, output, and feedthrough matrices, respectively, and pair (A, B) is assumed to be controllable. The following constraints are again applicable

$$x \in \mathbb{X}, \qquad u \in \mathbb{U} \tag{4-29}$$

$$w \in \mathcal{W} \subset \mathbb{R}^n \tag{4-30}$$

where $\mathbb{X} \subseteq \mathbb{R}^n$ is polyhedral and $\mathbb{U} \subseteq \mathbb{R}^m$ is polytopic.

4.4.2 Set point characterization

It is common to parameterize the system such that for a given set point y_s, any permissible terminal state $z_s = (x_s, u_s)$ must satisfy

$$\begin{bmatrix} A - I_n & B \\ C & D \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0_{n,1} \\ y_s \end{bmatrix} \tag{4-31}$$

or in compact form

$$Ez_s = Fy_s \tag{4-32}$$

The controllability of the matrix pair (A, B) is necessary to ensure that the solution to (4-32) is non-trivial. Parameterizing this solution yields

$$z_s = M_\theta \theta \tag{4-33}$$

$$y_s = N_\theta \theta \tag{4-34}$$

where the vector $\theta \in \mathbb{R}^{n_\theta}$ is a parameter vector characterizing any solution and M_θ and N_θ are suitably chosen matrices. In [2, 33], it is advocated that M_θ and N_θ be defined using the singular value decomposition (SVD) of E. This is, of course, just one acceptable method of determining these matrices, but it does have the benefit of generally producing a characterization in which θ has as few parameters as necessary.

If the SVD of E is defined by $E = U\Sigma V$, where $U \in \mathbb{R}^{(n+p)\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, and $V \in \mathbb{R}^{(n+m)\times r}$ are defined in the usual manner, then the matrices M_θ and N_θ are defined as

$$M_\theta = \begin{cases} [V\Sigma^{-1}U^T FG \quad V_\perp] & if \quad r < n + m \\ V\Sigma^{-1}U^T FG & if \quad r = n + m \end{cases} \tag{4-35}$$

$$N_\theta = \begin{cases} [G \quad 0_{p,n+m-r}] & if \quad r < n + m \\ G & if \quad r = m + n \end{cases} \tag{4-36}$$

with

$$G = \begin{cases} \mathbf{I}_p & if \quad r = n + p \\ (F^T U_\perp)_\perp & if \quad r < n + p \end{cases} \tag{4-37}$$

The subscript $\perp$ indicates that the associated parameter is chosen such that, for example, $V^T V_\perp = 0$ and $[V \quad V_\perp]$ is non-singular.

4.4.3 The nominal controller

Now consider, similarly to the regulator case, that the nominal controller disregards the perturbations

$$z^+ = Az + Bv$$

$$\bar{y} = Cz + Dv \tag{4-38}$$

the control law is of the form

$$u = v + K_{dr}(x - z) \tag{4-39}$$

and there exists an RPI set $\mathcal{Z}$ for the perturbed system controlled by (4-39). The determination of the set $\mathcal{Z}$ will be thoroughly explored in the next chapter.

Tightened constraints

The tightened state and input constraints for the nominal system are defined by the Pontryagin set difference of the robust constraint and the minimal robust positively invariant set $\mathcal{Z}$ or $\mathcal{Z}$ scaled by the disturbance rejection gain, respectively

$$\bar{\mathbb{X}} = \mathbb{X} \ominus \mathcal{Z}, \qquad \bar{\mathbb{U}} = \mathbb{U} \ominus K_{dr}\mathcal{Z} \tag{4-40}$$

where the operator $\ominus$ indicates the Pontryagin set difference. Then, any admissible steady state z_s must satisfy the constraints

$$z_s = M_\theta \theta = (x_s, u_s) \in \bar{\mathbb{X}} \times \bar{\mathbb{U}} = \bar{\mathbb{Z}} \tag{4-41}$$

After the application of these constraints, the set of admissible steady states in (x, u)-space is given by

$$\mathcal{F}_s = \{z_s = M_\theta \theta \mid M_\theta \theta \in \bar{\mathbb{Z}}\} \tag{4-42}$$

However, the steady states need to be defined in x-space. This projection is defined by

$$\mathcal{X}_s = Proj_x(\mathcal{F}_s) = \left\{x \in \mathbb{R}^n \mid \exists\, y \in \mathbb{R}^k \;\; such\ that \; \begin{bmatrix} x \\ y \end{bmatrix} \in \mathcal{F}_s \right\} \tag{4-43}$$

where $\mathcal{F}_s \subseteq \mathbb{R}^{n+k}$. Similarly, the set of admissible control actions

$$\mathcal{U}_s = Proj_u(\mathcal{F}_s) \tag{4-44}$$

is projected into u-space. Finally, the set of admissible set points

$$\mathcal{Y}_s = \{y_s = N_\theta \theta \mid M_\theta \theta \in \bar{\mathbb{Z}}\} \tag{4-45}$$

Alternatively, $\mathcal{X}_s$ and $\mathcal{U}_s$ can be defined in terms of the set of admissible parameters θ. In this case, it is important to note that

$$\Theta_s = \{\theta \mid M_\theta \theta \in \bar{\mathbb{Z}}\} \tag{4-46}$$

to achieve the admissible sets

$$\mathcal{X}_s = M_x \Theta_s \tag{4-47}$$

$$\mathcal{U}_s = M_u \Theta_s \tag{4-48}$$

where

$$M_x = [\mathbf{I_n} \quad \mathbf{0_{n,m}}] M_\theta \tag{4-49}$$

$$M_n = [\mathbf{0_{m,n}} \quad \mathbf{I_m}] M_\theta \tag{4-50}$$

This parameterization of all of the admissible sets solves the uniqueness issue outlined in *section 3.6* in that each θ is associated with exactly one output value [2].

The Invariant Set for Tracking

In the most simplistic formulation of the robust controller, the disturbance rejecting controller is a linear feedback controller of the form $u = Kx$ where K is a gain – for example LQR gain – and the terminal set is RPI. The more general tracking problem for arbitrary set points is more complex. [2]

To achieve the goal of maintaining the uncertain system within the neighbourhood of a reference admissible terminal state z_s the control law has the form common to Tube-based robust MPC, mildly adjusted to meet its goal

$$u = u_s + K_\Omega (x - x_s) \tag{4-51}$$

Selecting a larger K_Ω results in a larger domain of attraction, and thereby a larger invariant set for tracking. It is common to choose K_Ω to be the LQR gain K_{lqr} [33].

Furthermore, as the system is constrained, the steady state cannot simply be shifted from the origin of the regulation problem to some other non-zero steady state set. The terminal constraint set therefore needs to be reformulated as an invariant set for tracking [2, 32]:

With some abuse of notation, define an extended state for the closed loop dynamics

$$x^e = (x, \theta) \in \mathbb{R}^{n+n_\theta} \tag{4-52}$$

and a control gain

$$K_\Omega \in \mathbb{R}^{m \times n} \tag{4-53}$$

such that $A + BK_\Omega$ is Hurwitz. Also let

$$K_\theta = [-K_\Omega \quad \mathbf{I}_m] M_\theta \tag{4-54}$$

Then, a set

$$\Omega_k^e \subset \mathbb{R}^{n \times n_\theta} \tag{4-55}$$

is an admissible invariant set if for all

$$(x, \theta) \in \Omega_k^e \tag{4-56}$$

it holds that

$$x \in \bar{\mathbb{X}}, K_\Omega x + K_\theta \theta \in \bar{\mathbb{U}} \tag{4-57}$$

$$\left((A + BK_\Omega)x + BK_\theta \theta, \theta\right) \in \Omega_t^e \tag{4-58}$$

This means simply that for any initial state, the trajectory of the unperturbed system controlled by (4-51) will satisfy $x(i) \in Proj_x(\Omega_k^e)$, and that the use of an invariant set for tracking results in an optimization problem which does not require re-computation of the terminal set if the setpoint changes. The constrained optimal control problem and the robust constraint satisfaction can therefore be handled independently.

The objective function

Recall for the introduction of *section 4.4* that the tube-base robust tracking problem makes use of an artificial steady state

$$\bar{z}_s = (z_s, v_s) \tag{4-59}$$

as an additional decision variable to obtain an increased region of attraction (refer to the introduction of *section 4.4)*. Since the admissible steady state can be parameterized as $\bar{z}_s = M_\theta \bar{\theta}$ where $\bar{\theta} \in \Theta_s$. Therefore $\bar{\theta}$ can be treated as the decision variable.

The cost function then has a similar form to those seen previously

$$V_N(x, \theta; z_0, \mathbf{v}, \bar{\theta}) = \sum_{i=0}^{N-1} l(z_i, z_s, v_i, v_s) + V_f(z_N, z_s) + V_0(\bar{\theta}, \theta) \tag{4-60}$$

which has a slightly modified stage function

$$l(z_i, z_s, v_i, v_s) = \left||z_i - z_s|\right|_Q^2 + \left||v_i - v_s|\right|_R^2 \tag{4-61}$$

and terminal cost

$$V_f(z_N, z_s) = \left||z_N - z_s|\right|_P^2 \tag{4-62}$$

and includes an additional steady state offset cost

$$V_0(\bar{\theta}, \theta) = |\left|\bar{\theta} - \theta\right||_T^2 \tag{4-63}$$

The offset cost term penalizes the deviation of the artificial steady state and the desired steady state and T is the steady state offset weighting matrix.

Harkening again back to the regulation problem, the nominal states and control actions are subject to tightened constraints and the initial state z_0 of the nominal systems must satisfy

$$z_0 \in \{x\} \oplus (-\mathcal{Z}) \tag{4-64}$$

and the predicted terminal state z_N and the steady state parameter $\bar{\theta}$ satisfy

$$(z_N, \bar{\theta}) \in \Omega_k^e \tag{4-65}$$

The optimal control problem then becomes

$\mathbb{P}_N^t(x,\theta)$:

$$V_N^{t*}(x,\theta) = \min_{\mathrm{z}_0,\mathbf{v},\bar{\theta}} \{V_N(x,\theta;z_0,\mathbf{v},\bar{\theta}) | \, \mathbf{v} \in \mathcal{U}_N(z_0,\bar{\theta}), z_0 \in \{x\} \oplus (-\mathcal{Z})\} \tag{4-66}$$

$$\left(z_0^{t*}(x,\theta), \mathbf{v}^{t*}(x,\theta), \bar{\theta}^{t*}(x,\theta)\right) = \arg\min_{\mathrm{z}_0,\mathbf{v},\bar{\theta}} \{V_N(x,\theta;z_0,\mathbf{v},\bar{\theta}) | \, \mathbf{v} \in \mathcal{U}_N(z_0,\bar{\theta}), z_0 \in \{x\} \oplus (-\mathcal{Z})\} \tag{4-67}$$

It should be noted here that the feasible region of $\mathbb{P}_N^t(x,\theta)$ depends only on the current state, and not on the steady state parameter [32]. The set of admissible nominal control actions is given by

$$\mathcal{U}_N(z_0,\bar{\theta}) = \{\mathbf{v} \mid v_i \in \bar{\mathbb{U}}, \bar{\mathbf{\Phi}}(\boldsymbol{i}; z_0, \mathbf{v}) \in \bar{\mathbb{X}} \, for \; i = 0,1,\dots,N-1, (\bar{\mathbf{\Phi}}(N; z_0,\mathbf{v}), \bar{\theta}) \in \Omega_k^e\} \tag{4-68}$$

The set of admissible nominal initial states takes the form

$$\bar{\mathcal{X}}_N = \{z \mid \exists \bar{\theta} \; such \; that \; \mathcal{U}_N(z,\bar{\theta}) \neq \emptyset\} \tag{4-69}$$

and the set of admissible actual initial states the form

$$\mathcal{X}_N = \{x \mid \exists (z_0,\bar{\theta}) \; such \; that \; z_0 \in \{x\} \oplus (-\mathcal{Z}), \mathcal{U}_N(z_0,\bar{\theta}) \neq \emptyset\} \tag{4-70}$$

As is clear at this point, the sequence of predicted optimal control inputs $\mathbf{v}^*(x,\theta)$ is obtained from the solution of $\mathbb{P}_N^{t*}(x,\theta)$. Again, the first element of $v_0^*(x,\theta)$ is applied to the feedforward term in the implicit controller

$$\kappa_N(x,\theta) = v_0^*(x,\theta) + K(x - z_0^*(x,\theta)) \tag{4-71}$$

5 A practical exposition of Tube-based Robust MPC for tracking: The double pendulum

In this final chapter of *Part 1*, a sample problem is presented to illustrate the implementation of the Tube-based Robust MPC for tracking theory presented in the previous chapter. The double pendulum is a well-studied control task. The problem is used as a sample task in literature often. The aim is to here present the design and implementation procedure as clearly and in as much detail as possible, which will require additional inline exposition. The implementation of this controller design and simulation in software is given in Appendix part A. Double Pendulum.

It makes sense, then, to start from the beginning:

5.1 Defining the problem

Let there be a disturbed system of the form of (4-28), with a nominal system described as in (4-38), a disturbance characterized by (4-30), and system and tightened constraints (4-29) and (4-30), respectively. In this example, the double pendulum will be controlled using tracking state feedback control methods as presented in [1, 2, 4, 33]. The disturbed system can be described as follows

$$x^+ = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 & 0.5 \\ 1 & 0.5 \end{bmatrix} u + w$$

$$y = [1 \quad 0]x \tag{5-1}$$

and the nominal system obtained by neglecting the disturbance is given by

$$z^+ = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} z + \begin{bmatrix} 0 & 0.5 \\ 1 & 0.5 \end{bmatrix} v \tag{5-2}$$

The state and control constraints given in [1, 2, 33] for the disturbed system and disturbance bound are given by their H-representations:

$$\mathbb{X} = \{x \mid \|x\|_\infty \leq 5\} = \{x \in \mathbb{R}^n \mid A_x x \leq b_x\} \tag{5-3}$$

$$\mathbb{U} = \{u \mid \|u\|_\infty \leq 0.3\} = \{u \in \mathbb{R}^m \mid A_u u \leq b_u\} \tag{5-4}$$

$$\mathcal{W} = \{w \mid \|w\|_\infty \leq 0.1\} = \{w \in \mathbb{R}^n \mid A_w w \leq b_w\} \tag{5-5}$$

This square problem has two states and two inputs. The constraints and disturbance can therefore be equivalently expressed as $|x_1| \leq 5, |x_2| \leq 5, |u_1| \leq 0.3, |u_2| \leq 0.3,$ and $w \in \mathcal{W} = 0.1\mathcal{B}^2$. These constraints are polytopic or polyhedral (refer to (4-29)), and can as easily be defined as such through their vertices in V-representation. These constraints are implemented in software using the Multi-Parametric Toolbox, see lines 15-24 of the implemented software in the Appendix. The Multi-Parametric Toolbox (MPT) for MATLAB permits working with polyhedral/polytopic constraints in control problems, and it is used in this implementation extensively.

The objective now is to devise a state feedback tube-based Robust model predictive controller which will steer the actual trajectory near the nominal trajectory while robustly satisfying the system and actuation constraints. This controller is of two parts: an LTI feedback control gain K_{dr} for disturbance rejection which ensures that the deviation between the actual and nominal state is bounded and a nominal MPC gain v determined at each time step. The combined form of the control law will then, of course, be

$$u = v + K_{dr}(x - z) \tag{5-6}$$

There are a few quantities which must be determined before this control law can be constructed. This will be the subject of the rest of the chapter.

- First, the tube of trajectories must be addressed. This is a rather involved process which will be tackled in parts. From results obtained in this consideration, the disturbance rejection gain and the tightened constraints for the nominal system can be determined.
- Following this, the nominal MPC problem can be formulated. This will also be conducted in steps covering each important aspect of the construction.

In each of the following sections, the method of implementation will be discussed, pertinent algorithms clearly indicated, and design parameter choices will be explained. Intermediary results will also be provided for understanding. Finally, some results from the simulation of this implementation will be presented.

5.2 The tube of trajectories

Recall from *chapter 4* that the tube, centered on the nominal trajectory, contains all feasible trajectories under which constraints are robustly satisfied for any realization of the uncertainty. Recall, as well, that the tube is constructed by the affine mapping of the minimal robust positively invariant (mRPI) set along the nominal trajectory. Thus, to determine the mRPI is to determine the tube. This section will present the practical determination of the mRPI, the tube of trajectories, and the tightened constraints.

5.2.1 The disturbance rejection gain

The first step in determining the mRPI is to select the disturbance rejection gain. This is, of course, an important parameter in the control law (5-6), compensating the deviation between actual state x and nominal state z and characterizing the closed loop dynamics in the presence of disturbances [33]. This parameter and its determination resembles the Kothare's controller for Min-Max MPC [59], however in tube-based robust MPC it is only determined once at the beginning of the controller design. This difference is important in reducing the computational complexity, and so in simplicity advantage of tube-based robust MPC.

The disturbance rejection gain K_{dr} will be chosen using a disturbance rejection criterion, as indicated in [33], which will ensure the existence of an RPI set $\mathcal{Z}$ such that the tightened constraints are non-empty and which will minimize the size of $\mathcal{Z}$. This robustness criterion will additionally ensure a larger domain of attraction, which is desirable. A practical method of determining the disturbance rejection gain, based largely on the Kothare's controller [59], will be presented here. This method is derived closely following [33].

The form of the controller in the synthesis of this gain is $u = K_{dr}x$, and the design problem can be briefly described as the minimization of the size of an ellipsoid $\mathcal{E}(P,1) = \{x \in \mathbb{R}^n |\, x^T P x \leq 1\}$. P is the terminal weighting matrix of the robust control optimization problem, which, along with the gain K_{dr}, will be determined in this minimization problem. Consider the system and actuation constraints defined in their respective normalized H-representations, such that

$$\mathbb{X} = \{x \in \mathbb{R}^n \mid |h_i^T x| \leq 1, i = 1, \dots, n_{rx}\} \tag{5-7}$$

$$\mathbb{U} = \{u \in \mathbb{R}^m \mid |l_j^T u| \leq 1, j = 1, \dots, n_{ru}\} \tag{5-8}$$

where n_{rx} and n_{ru} are the number of rows in the defining matrices of the corresponding sets, and h_i and l_j are the rows of the matrix derived from the normalization of A_x and A_u with respect to b_x and b_u, as appropriate. The constraints to this optimization problem are detailed in the following list. For optimization using matrices containing unknowns, the constraints need be transformed into Linear Matrix Inequalities (LMI). In the following exposition, the constraints are stated and transformed into this useful form.

(1) The ellipsoid $\mathcal{E}(P,1)$ is an RPI set for the system. This condition is formulated as

$$(x^+)^T P (x^+) \leq 1$$
$$\forall x \in \mathcal{E}(P,1)$$
$$x^+ = A_k x + w$$
$$\forall w \in \mathcal{W} \tag{5-9}$$

The S-procedure [60] is applied to inequality (5-9). When the convexity with respect to w is considered, this modified condition is satisfied if there exists a multiplier $\lambda \geq 0$ such that

$$\left((A_{K,dr})x + w\right)^T P\left((A_{K,dr})x + w\right) + \lambda(1 - x^T P x) \leq 1$$

$$\forall w \in vert(\mathcal{W}) \tag{5-10}$$

with $vert(\mathcal{W})$ indicating the vertices of the disturbance bound.

This is the result of the so-called S-procedure [60] and is an indication of the conditions under which the particular quadratic inequality is non-negative. Applying the Schur complement to the inequality, the linear matrix inequality (LMI) is formed. Applying this step and expanding A_k to make clear the presence of the target variable K_{dr}, this constraint can be written

$$\begin{bmatrix} \lambda P - (A + BK_{dr})^T P(A + BK_{dr}) & -(A + BK_{dr})^T P w \\ -w^T P(A + BK_{dr}) & 1 - \lambda - \omega^T P w \end{bmatrix} > 0, \quad \forall w \in vert(\mathcal{W}) \tag{5-11}$$

The Schur complement is a useful tool for transforming inequalities into LMIs by restructuring a linear equation into a matrix in a manner mimicking Gaussian elimination [61]. Variable changes $W = \gamma P^{-1}$ and $Y = K_{dr}W$ are then made and the LMI becomes

$$\begin{bmatrix} \lambda W & * & * \\ 0 & 1 - \lambda & * \\ AW + BY & \omega & W \end{bmatrix} > 0, \qquad \forall \omega \in vert(\mathcal{W}) \tag{5-12}$$

The constraint is symmetric indicated by $*$ in the matrix.

(2) LMI formulation of the actuation constraint:

For all $x \in \mathcal{E}(P, 1)$, the control law $\left|l_j^T K_{dr} x\right| \leq \rho_j$ for all rows of l and with $\rho_j \in (0,1]$. The magnitude of the parameter ρ_j is chosen to restrict the size of the set of admissible control inputs such that the tightened set $\overline{\mathbb{U}} = \mathbb{U} \ominus K_{dr}\mathcal{Z}$ is not empty. Applying the Schur complement to the condition $l_j^T K_{dr} P^{-1} K_{dr}^T l_j \leq \rho_j^2, j = 1, \ldots, n_{ru}$ yields the LMI

$$\begin{bmatrix} \rho_j^2 & l_j^T K_{dr} \\ K_{dr}^T l_j & P \end{bmatrix} > 0, j = 1, \ldots, n_{ru} \tag{5-13}$$

The same variable change made in (1) is applied to obtain the condition

$$\begin{bmatrix} \rho_i^2 & * \\ Y^T l_i & W \end{bmatrix} > 0, \qquad i = 1, \ldots, n_{ru} \tag{5-14}$$

(3) LMI formulation of the state constraint:

Following similar steps for the state set, for all $x \in \mathcal{E}(P, 1)$, $|h_i^T x| \leq 1$ becomes

$$\begin{bmatrix} 1 & h_i^T \\ h_i & P \end{bmatrix} > 0, i = 1, \ldots, n_{rx} \tag{5-15}$$

The measure used to minimize the size of the ellipsoid $\mathcal{E}(P, 1)$ in the adopted method is a parameter $\gamma > 0$ such that $\mathcal{E}(P, 1) \subseteq \sqrt{\gamma}\,\mathbb{X}$. The minimization problem then becomes the minimization of γ to a value between $(0,1]$. Applying standard operations of LMIs, the minimization problem is fully formulated as follows

$$\min_{Y,W,\gamma} \gamma \tag{5-16}$$

$$s.t. \begin{bmatrix} \lambda W & * & * \\ 0 & 1-\lambda & * \\ AW+BY & \omega & W \end{bmatrix} > 0, \qquad \forall \omega \in vert(\mathcal{W})$$

$$\begin{bmatrix} \rho_i^2 & * \\ Y^T l_i & W \end{bmatrix} > 0, \qquad i = 1, \dots, n_{ru}$$

$$\begin{bmatrix} \gamma & * \\ W h_i & W \end{bmatrix} > 0, \qquad i = 1, \dots, n_{rx}$$

Each of the constraint LMIs are symmetric, indicated by $*$ in the matrices.

In order to reproduce the results given in literature, this application to obtain the mRPI chooses ρ_i to obtain a certain sized tightened control constraint and λ is taken as small as possible such that a solution exists. Both of these quantities should be selected before the optimization is conducted. In this implementation, ρ_i is chosen as dictated in [33]. As the CVX toolbox cannot resolve optimizations where two of the optimization parameters are multiplied together (refer to [62] for a detailed explanation as to why this is), λ is determined external to the optimization itself. In this initial implementation, the λ which resulted in the minimization of γ was determined by experimentation.

The optimization must then be conducted such that each vertex of $\mathcal{W}$ and each half-space of the state and control constraint sets is considered. W, Y, and γ are the optimization variables. The LMI solver will vary W and Y until γ is minimized. In a feasible solution,

$$P = W^{-1} \tag{5-17}$$

and

$$K_{dr} = YW^{-1} \tag{5-18}$$

In this example, $\rho = 0.48$ is chosen to obtain a K_{dr} which would allow the same size set $K\mathcal{Z}$ as for the LQR gain K_{lqr} [33]. Parameter $\lambda = 0.49$ is chosen as this corresponds to the smallest value of γ. This value was determined by experimentation. The code used to conduct this process for this implementation is indicated in Appendix part A. Double Pendulum lines 30-74. The disturbance rejection gain obtained through this implementation is

$$K_{dr} = \begin{bmatrix} -0.1181 & -0.5654 \\ -0.2154 & -0.6462 \end{bmatrix} \tag{5-19}$$

5.2.2 The mRPI set $\mathcal{Z}$

Now that the disturbance rejection gain is known, the mRPI set can be determined. Recall from *section 4.2* that the mRPI set is the smallest RPI set contained in every closed RPI set of system. The set is defined through the iterative Minkowski summations of the set given by the disturbance bound $\mathcal{W}$ with the system dynamics $A_{k,dr}$

$$F_s = \bigoplus_{k=0}^{s} (A + BK_{dr})^k \mathcal{W} \tag{5-20}$$

with s tending to infinity. The disturbance set is effectively grown by an amount at each iteration dictated by the nominal dynamic of the system. F_∞ can only be determined in special cases, so the set $\mathcal{Z}$ is estimated such that $F_\infty \subseteq \mathcal{Z} \subseteq F_\infty \oplus \epsilon \mathcal{B}^n$ for some chosen bound of the error ϵ. To this end, the following functions are calculated

$$\alpha(s) = \min \alpha \quad s.t. \;\; A_K^s \mathcal{W} \subseteq \alpha \mathcal{W} \tag{5-21}$$

$$\beta(s) = \min \beta \quad s.t. \;\; F_s \subseteq \beta \mathcal{B}^n \tag{5-22}$$

through a series of linear programming problems. Once a large enough s is obtained such that $\left(1 - \alpha(s)\right)^{-1} \alpha(s)\beta(s) \le \epsilon$, then the set $Z = \left(1 - \alpha(s)\right)^{-1} F_s$ which is an approximation of F_∞ with an error bound less than ϵ.

This implementation will consider only the practical point of view, following from [33, 58]. The reader is directed to these papers for more detail. The first step is to determine s, followed by the construction of set $\mathcal{Z}$.

The determination of s is an iterative process. First, it should be noted that the support function of a polytopic uncertainty, evaluated at $a \in \mathbb{R}^m$, is given by

$$h_{\mathcal{W}}(a) = \sup_{w \in \mathcal{W}} a^T w \tag{5-23}$$

As $\mathcal{W}$ is polytopic and describable by a finite set of affine inequalities, $h_{\mathcal{W}}(a)$ will be finite and can be calculated through a finite number of linear programs. An initial estimate for α is then given by

$$\alpha(s) = \max_{i \in \mathfrak{I}} \frac{h_{\mathcal{W}}((A^s)^T A_{w,i}^T)}{b_{w,i}} \tag{5-24}$$

for some guess of s. An *a priori* error bound for the approximation of $\mathcal{Z}$ can then also be computed using the support function, through the summation

$$M(s) = \max_{j \in \{1,\dots,n\}} \left\{ \sum_{i=0}^{s-1} h_{\mathcal{W}}\left(\left(A^i\right)^T e_j \right), \sum_{i=0}^{s-1} h_{\mathcal{W}}\left(-\left(A^i\right)^T e_j \right) \right\} \tag{5-25}$$

where e_j is the j^{th} standard basis vector. After each iteration, s is incremented. The value of s which yields the outer ϵ –approximation of the mRPI $\mathcal{Z}$ is obtained when

$$\alpha(s) \le \frac{\epsilon}{\epsilon + M(s)} \tag{5-26}$$

This process is summarized by the following algorithm:

Algorithm 5-1 Determination of parameter s

Inputs: System and input matrices, so that $A_{K,dr} = A + BK_{dr}$

Disturbance $\mathcal{W}$

Algorithm:

```
$s \longleftarrow 1$
Determine initial $\alpha(s)$ using (5-24)
Set $M(s)$ through (5-25)

while $\alpha(s) > \frac{\epsilon}{\epsilon + M(s)}$
    $s \longleftarrow s + 1$
    Determine initial $\alpha(s)$ using (5-24)
    Set $M(s)$ through (5-25)
end
```

Output: parameter s

Now, the construction of $\mathcal{Z}$ is relatively easy. Making use of the zonotopic representation of the polytopic uncertainty set, $\mathcal{W} = H\mathcal{B}^n \oplus \omega_0$, the uncertainty is simply a scaled norm ball centered on ω_0 with H non-singular. Define a matrix $H_z(s) = \left[A_{K,dr}^{s-1}H,\ A_{K,dr}^{s-2}H, \dots, H\right]$ and note that

$$\hat{\alpha}(s) = \left\|H^{-1}A_{K,dr}^{s}H\right\|_\infty = \min \alpha \quad s.t.\ \ A_{K,dr}^{s}H\mathcal{B}^n \subseteq \alpha H\mathcal{B}^n \tag{5-27}$$

$$\hat{\beta}(s) = \|H_z(s)\|_\infty = \min \beta \qquad s.t.\ \bigoplus_{k=0}^{s-1} A_{K,dr}^{k}H\mathcal{B}^n \subseteq \beta\mathcal{B}^n \tag{5-28}$$

By combining these definitions, for this form of uncertainty, it is seen that s is such that

$$\hat{\alpha}(s) = \left\|H^{-1}A_{K,dr}^{s}H\right\|_\infty \leq \frac{\epsilon}{\epsilon + M(s)} \tag{5-29}$$

So, after constructing $H_z(s)$ and B^{ns}, $\mathcal{Z}$ is denoted by

$$\mathcal{Z} = \left(1 - \hat{\alpha}(s)\right)^{-1} H_z(s)\mathcal{B}^{ns} \oplus \left(I_n - A_{K,dr}\right)^{-1}\omega_0 \tag{5-30}$$

These final construction steps are outlined in the following algorithm:

Algorithm 5-2 Construction of mRPI, $\mathcal{Z}$

```
Inputs:      System and input matrices, so that $A_{K,dr} = A + BK_{dr}$
             Disturbance $\mathcal{W}$
             Parameter $s$

Algorithm:   $\mathcal{B}^{n1} \leftarrow \mathcal{B}^n$
             $H_z \leftarrow [\,]$
             for 1 : i : s-1
                 $\mathcal{B}^{ni} \leftarrow \mathcal{B}^{ni} * \mathcal{B}^n$
             end
             for 1 : j : s
                 $H_z \leftarrow [H_z \quad A_{K,dr}^{s-1} H]$
             end
             $\mathcal{Z} \leftarrow \left(1 - \alpha(s)\right)^{-1} H_z(s) \mathcal{B}^{ns}$

Output:      mRPI set $\mathcal{Z}$
```

In this application, set $\epsilon = 0.01$ and s comes to 16. The resultant mRPI set is given here:

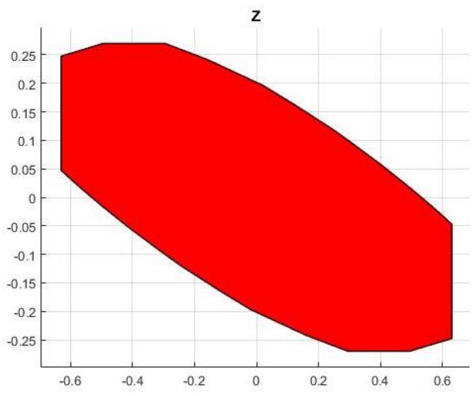


Figure 5-1 Minimal Robust Positively Invariant Set

The determination of the mRPI set implemented in software is given in Appendix part A. Double Pendulum lines 79-139.

5.2.3 Construction of the tube

The practical construction of the tube is a relatively simple procedure. As previously explained, the tube is basically a series of copies of the mRPI set along the nominal trajectory, with the origin of the set co-locating with the nominal state. To build the tube, then, a series of translated mRPI sets is iteratively populated. This simple process is as follows:

Algorithm 5-3 Tube construction

Inputs: mRPI and nominal trajectory

Algorithm:

```
$tube \leftarrow [\,]$
for 1 : i : simulation horizon
    $tube \leftarrow [tube \quad affine\_mapping(\mathcal{Z}, x(i))]$
end
```

Output: Vector of sets describing a tube within which all permissible trajectories will lie

5.2.4 The tightened constraints $\bar{\mathbb{X}}$ and $\bar{\mathbb{U}}$

Finally, the tightened constraints are simple to determine. Recall from *chapter 4* that these take the forms

$$\bar{\mathbb{X}} = \mathbb{X} \ominus \mathcal{Z} \tag{5-31}$$

$$\bar{\mathbb{U}} = \mathbb{U} \ominus K_{dr}\mathcal{Z} \tag{5-32}$$

where $\ominus$ indicates the Pontryagin set difference. This definition has implications, which are important to the understanding of the purpose of the mRPI set. The tightened constraints are those which apply solely to the nominal system. They describe the regions in which the nominal states/trajectory and inputs can exist. They must therefore be an amount and manner smaller than the full system constraints so that the disturbed system will still robustly satisfy these constraints if the nominal state is located along the boundary of the tightened constraint.

The MPT toolbox provides an implementation for the Pontryagin difference. Using this and quantities determined in the preceding, the tightened constraints are determined according to (5-31) and (5-32) through lines 144-146 of Appendix part A. The obtained sets are as follows.

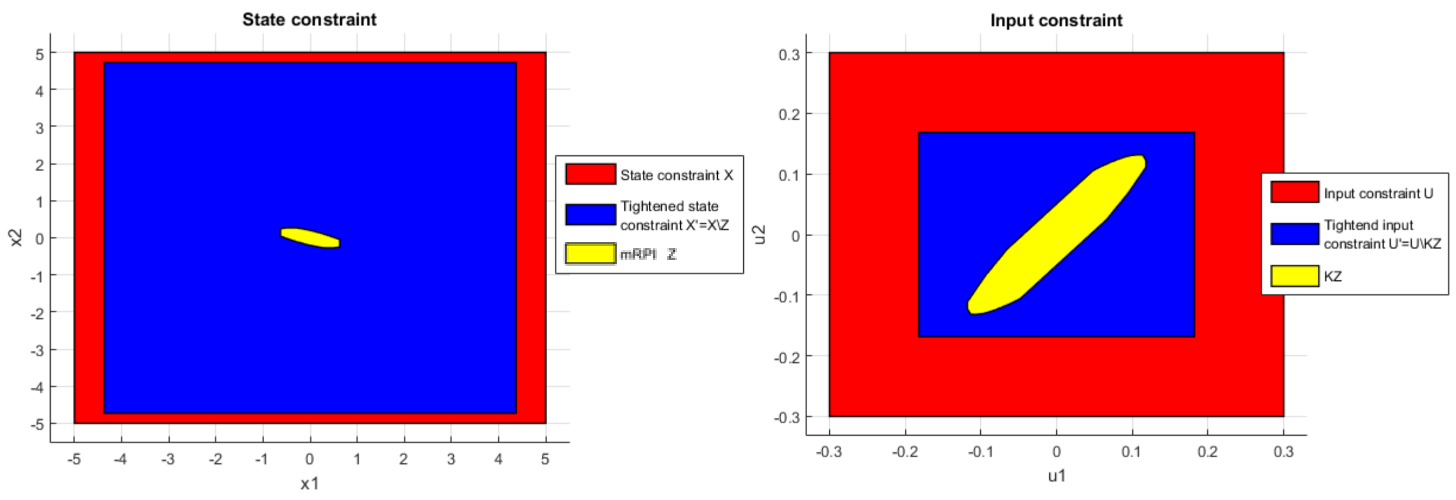


Figure 5-2 Tightened constraints

In the figures, the red sets indicate the robust state and input constraints, respectively. The blue sets are the nominal constraints. The yellow set in state space is simply the mRPI $\mathcal{Z}$ and in input space is the mRPI set scaled by the disturbance rejection gain $K_{dr}\mathcal{Z}$.

5.3 Set point parameterization and the artificial steady state

The set points which can be reached while robustly satisfying the actual system constraints are those terminal states of the nominal system admissible under the tightened constraints (5-31) and (5-32) [32]. The relationship of this subspace of terminal states to the set of admissible set points is characterizable through a parameterization. However, in the literature in which this example has been presented, [1, 4, 33], while published by the same research group following from the same body of work, various parameterizations stemming from at least 2 parameterization methods and yielding θ of different sizes are presented. Each of these parameterizations is correct and yield the same results, however Alvarado, Limon, and the others generally advocate that the parameterization should be conducted such that the parameter θ should be of the smallest number of elements possible to reduce computational complexity later. Technically then, one parameterization would be ‘better’ than others.

Nonetheless, the expressed characterization (4-33)-(4-34) allows the parameterization of the subspace of steady states and inputs via a single decision parameter θ. The implementation in code is straight forward, and the method described by (4-35)-(4-37) can be used. The implementation presented in the Appendix made use of a basic parameterization obtained from the form of the system model with the result verified in [4].

Designing the weighting matrix for the offset cost (refer to (4-63)) is also simple. The weighing matrix T is initially chosen close to the terminal weight, as recommended in [33], and then tuned to achieve a fast transient with minimal optimality loss. In this implementation, $T = 1000 * P_{lqr}$ is taken from [33], where P_{lqr} is the terminal weighting matrix discussed in the following.

5.4 The terminal set and constraints

The final part of the controller design which needs to be considered are the prediction horizon terminal aspects. The terminal set $\mathbb{X}_f$ for the prediction horizon truncated to N steps replaces the terminal set of the infinite horizon $N \to \infty$ problem. It is customary to imitate the infinite horizon cost for the actual system through the infinite horizon cost of the unconstrained system, determined through the solution for P in the Riccati equation. This forms the basis of LQR MPC. Using the unconstrained LQR parameters, in particular the gain K_{lqr}, has the added bonus of allowing a larger terminal set to be determined. The terminal cost must then also be addressed. Therefore, the LQR infinite horizon gain and terminal weighting matrices need to be considered here.

5.4.1 LQR gain and terminal weight

These parameters are dependent on the LQR weighting matrices, which are taken in this implementation from [33] as $Q = I_2$ and $R = 10\, I_2$. For this discrete problem, the MATLAB function *dlqr* is utilized to obtain the gain K_{lqr} and terminal weight matrix P_{lqr}. This function makes use of the system and input matrices and the weighting matrices Q and R to determine the LQR gain K_{lqr} and the terminal weight P_{lqr}.

There is one caveat to using this available function. The function *dlqr* assumes that $A_{K,lqr} = A - BK_{lqr}$. This design methodology expects $A_{K,lqr} = A + BK_{lqr}$. So, to obtain the expected results, the gain obtained from the *dlqr* function be multiplied by -1.

5.4.2 Terminal set $\mathbb{X}_f$

The terminal set $\mathbb{X}_f$ is the set of states within which the final predicted state on the prediction horizon must reside. In order to obtain the largest region of attraction, encompassing as much of the state constraint set as possible, the terminal set $\mathbb{X}_f$ should be as large as possible. In tube-based robust MPC for tracking, this set is called the Invariant Set for Tracking, which is chosen as the Maximal Robust Positively Invariant Set (MRPI). This set is determined using the closed-loop dynamics of the extended state

$$x^e = (x, \theta) \tag{5-33}$$

encompassing, with some stretching of notation, the nominal system state x and an admissible terminal state θ. To describe these dynamics, first consider the nominal control law

$$u = K_{lqr}(x - z_s) + v_s \tag{5-34}$$

where z_s and v_s indicate the desired terminal state and actuation of the system. Let this be re-written as

$$u = K_{lqr}x + L\theta \tag{5-35}$$

where $L = [-K_{lqr} \quad I_m]M_\theta$.

Provided that $A + BK_{lqr}$ is Hurwitz, two important results occur. First, the closed-loop system will evolve to the terminal state and input described by $z_s = M_\theta \theta$. Second, the invariant set for tracking is the set of states and inputs which are admissibly stabilizable by this control law.

Then, for the extended state x^e, the closed loop system is posed as

$$\begin{bmatrix} x \\ \theta \end{bmatrix}^+ = \begin{bmatrix} A + BK_{lqr} & BL \\ 0 & I_{n\theta} \end{bmatrix} \begin{bmatrix} x \\ \theta \end{bmatrix} \tag{5-36}$$

or $x^{e+} = A_e x^e$. The constraint set for this system is given by

$$\mathbb{X}^e_{\lambda_{MRPI}} = \{x^e = (x,\theta) \mid (x, Kx + L\theta) \in \bar{\mathbb{X}} \times \bar{\mathbb{U}}\, , M_\theta \theta \in \lambda_{MRPI} \times (\bar{\mathbb{X}} \times \bar{\mathbb{U}})\} \tag{5-37}$$

where $\lambda_{MRPI} \in (0,1)$ will be discussed shortly.

The MRPI is determined using this closed loop system matrix, the LQR gain, the relationship of terminal states and inputs to the parameterized θ-space, and the nominal system constraints. It should be noted here, that due to the unitary eigenvalues of A_e, the MRPI set is not finitely determinable, but can be arbitrarily closely approximated [1-3, 32, 63-65]. This convex, finitely determined polyhedron is obtained by applying a scaling factor λ_{MRPI} arbitrarily close to 1 to the definition of the MRPI set,

$$\Omega_{\infty,\lambda}^{e} = \{x^e : A_e^i x^e \in \mathbb{X}_{\lambda_{MRPI}}^{e}, \forall i \geq 0\} \tag{5-38}$$

The determination of this set is therefore conducted practically as follows:

Algorithm 5-4 Determination of MRPI as the Invariant Set for Tracking

Inputs: Nominal constraints $\overline{\mathbb{X}}, \overline{\mathbb{U}}$,
Closed-loop system A_e,
LQR gain K_{lqr},
State and input parameterization matrix M_θ,
Scaler λ_{MRPI}

Algorithm:

```
Set $k \leftarrow 0$
Construct the constraint set for the closed loop system
    $\Omega_0 \leftarrow \left\{ \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_x M_\theta & 0 \\ 0 & 0 & A_u M_\theta \end{bmatrix} x \leq \begin{bmatrix} b_x \\ \lambda_{MRPI}\, b_x \\ \lambda_{MRPI}\, b_u \end{bmatrix} \right\} = \{A_\Omega x \leq b_\Omega\}$
Calculate 2 pre-sets for an initial comparison
    $\Omega_1 \leftarrow Pre(\Omega_0) \cap \Omega_0$
    $\Omega_2 \leftarrow Pre(\Omega_1) \cap \Omega_1$
while the current and next pre-sets are not equal $\Omega_{k+1} \neq \Omega_k$
    $k \leftarrow k + 1$
    Determine the next pre-set $\Omega_{k+1} \leftarrow Pre(\Omega_k) \cap \Omega_k$
end
```

Output: Maximal Robust Positively Invariant Set $\Omega_\infty = \Omega_k = \Omega_{k+1}$

For more information on this algorithm, the reader is directed to [63, 64].

In the preceding algorithm, the function $Pre(\cdot)$ indicates the "predecessor" set, or 1-step set. In practice, the pre-set is defined by formulating the polytope

$$\begin{bmatrix} & A_\Omega A_e & \\ A_x & & 0 \\ A_u K_{lqr} & A_u[-K_{lqr} & I]M_\theta \end{bmatrix} x \leq \begin{bmatrix} b_\Omega \\ b_x \\ b_u \end{bmatrix}$$

The algorithm basically states that the computation of pre-sets continues until two consecutive pre-sets are equal.

This process will yield the invariant set for tracking in extended space Ω^e. It is desirable to know this set in state space, Ω_x, or input space,Ω_u, to be able to apply the result. These must be found through a projection from the extended space onto the desired space. In this implementation, the appropriate functions from the MPT toolbox are made used to obtain the projections. The resultant invariant set for tracking is given below.

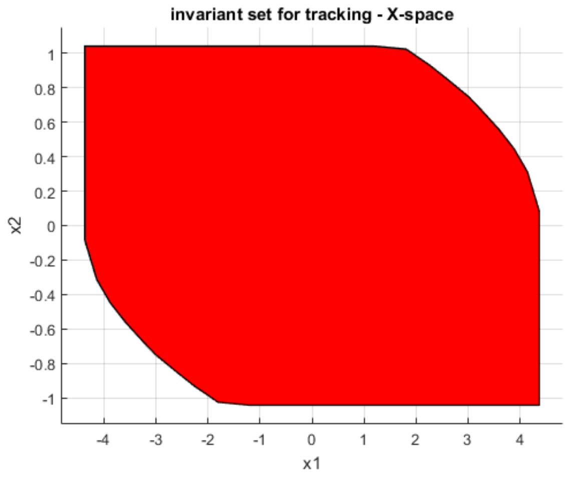


Figure 5-3 Invariant set for tracking projected onto X-space

This result was obtained using the implementation in software in Appendix part A. Double Pendulum lines 169-184 and 421-475. For more detail on this set and its determination, the reader is referred to [1-3, 32, 63-65].

5.5 Nominal model predictive controller

Now that all of the design parameters have been addressed, the nominal model predictive controller can be constructed. This controller was laid out in *section 5.5*, the reader is referred to this for the theory. The important results here are the cost function

$$V_N(x,\theta;z_0,\mathbf{v},\bar{\theta}) = \sum_{i=0}^{N-1}(\|z(i)-x_s\|_Q^2 + \|v(i)-u_s\|_R^2) + \|z(N)-x_s\|_P^2 + \left\|\bar{\theta}-\theta\right\|_T^2 \tag{5-39}$$

and optimization problem

$$\min_{z,\mathbf{v},\theta} V_N(x,\theta;z_0,\mathbf{v},\bar{\theta}) \tag{5-40}$$

$$s.t.\quad z \in x \oplus (-\mathcal{Z}) \tag{a}$$

$$z(i) \in \bar{\mathbb{X}}, \qquad i = 0,\dots,N \tag{b}$$

$$v(i) \in \bar{\mathbb{U}}, \qquad i = 0,\dots,N-1 \tag{c}$$

$$(z(N),\bar{\theta}) \in \Omega^e \tag{d}$$

While controller implementation through the multi-parametric toolbox is well documented in the toolbox manual and auxiliary documentation [7, 66], the remainder of this section will outline the practical aspects of constructing the implementation using the Multi-Parametric Toolbox.

The controller object must be called at each time step. The nominal controller object therefore only handles the prediction and optimization steps along a single prediction horizon. To construct a controller object, the optimization variables – the nominal and actual state, the nominal input, steady state characterized as θ, and the artificial steady state $\bar{\theta}$ – are defined as variables which can be manipulated by the optimization problem across the prediction horizon. The objective function and constraints are then considered simultaneously. Each call of the controller object encompasses one prediction cycle. Each begins with a prediction cost of 0. For the first $N-1$ prediction steps, the cost is determined according to the first term of (5-39) and the constraints (5-40)(b-c) and the nominal system dynamic (5-2) are defined. The terminal and offset costs are then determined according to the second and third terms of (5-39), respectively, and are added to the result of the summation in the previous step. The nominal and terminal state set constraints are determined for (5-40)(a and d). The nominal output can then be defined for the extended state (5-33). The set of constraints, cost, extended state, and output form are sent to the MPT optimizer function, which returns the controller object. The implementation of this controller in software can be found in Appendix part A. Double Pendulum lines 196-216 and 476-544.

5.6 Controller simulation

Simulating the disturbed system is fairly straight forward. First, a disturbance sequence is chosen. Then, the controller object constructed in the preceding sections is applied at each time step in the simulation horizon to obtain the controlled nominal response and the corresponding input. The successive actual state, is simply obtained by applying the current nominal state obtained from the controller and the current state of the disturbance to the dynamics described by the state equation in (5-1).

The results of such a simulation are presented here.

The initial and terminal states of the simulation was selected from the presentations of this example in literature [1, 2, 4, 33] as a diagnostic against which to compare this implementation. The assumption is that, as these works have been reviewed and accepted, comparable quality and form of results would indicate a correct understanding and implementation of the theory. Some parameters were modified in testing. The parameters used in the simulation are summarized in Table 5-1 at the end of this section. The simulation and the results presented here were designed to illustrate the ability of the controller to bring the system to a desired set point and to handle online changes in set point. The simulation was conducted and the results obtained using lines 223-259 and 545-576 of the software implementation in Appendix part A. Double Pendulum.

To this end, the simulation has two parts: the initial tracking problem, followed by a change in set point and a second tracking. The initial state is taken to be $x(0) = [-3 \quad 1.5]^T$ and the first set point $x_{s,A} = [-4 \quad 0]^T \in \Omega_x$. The second set point is chosen to demonstrate the ability of the controller to handle set points not located in the terminal set by tracking the nearest permissible set point. After successfully reaching the first set point, the simulation will change over to the second, $x_{s,B} = [4 \quad -0.5]^T \notin \Omega_x$.

The following figure demonstrates the progression of the controlled system in state space.

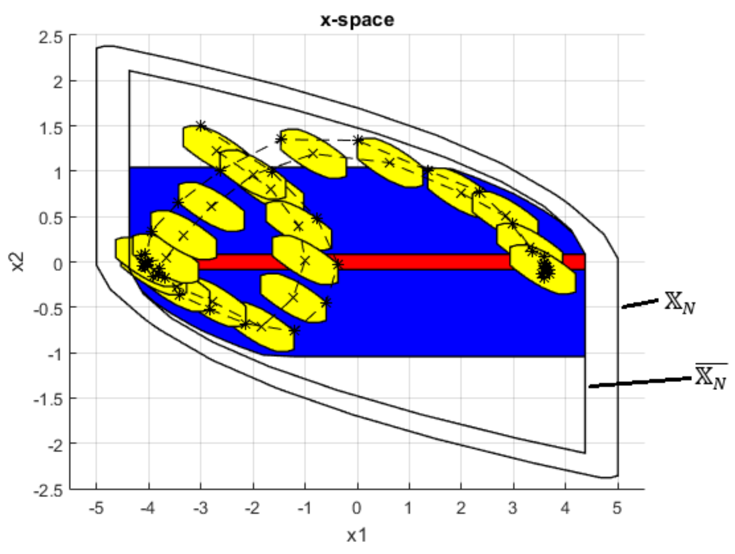


Figure 5-4 Controlled evolution pendulum system

This figure depicts the projection of the invariant set for tracking on state space Ω_x (blue), the set of admissible terminal states $\mathbb{X}_s$ (red), the tube of trajectories (yellow), and the boundaries of the nominal and robust regions of attraction $\overline{\mathbb{X}_N}$ and $\mathbb{X}_N$, respectively marked in the figure. The nominal and actual states are also indicated as trajectories with the markers x and $*$, respectively.

The projection of the invariant set for tracking represents the set of points in state space for which there exists a value for the parameterization variable θ such that there exists an associated steady state described in the MPRI set Ω^e. The set of admissible terminal states (red) is the representation of all terminal states which are relatable to the characterization of the target state (i.e. $y_{t,A} = -4$ or $y_{t,B} = 4$) through the parameterization of θ. The output equation of the system model and parameterization through N_θ of the set points means that the output is dependent only on state x_1. However, the set of admissible terminal states is obviously 2-dimensional. This is as a result of the set of admissible terminal state's definition, which is reliant on the definition of θ – involving the the state and input constraint and mRPI sets, which are all of course 2-dimensional.

The state space results indicate that after approximately 12 time steps in simulation, the system enters the neighbourhood of the first target steady state $x_{s,A}$. It will remain there until time step 21, when the target steady state is changed to $x_{s,B}$. The system will steadily progress toward this target state, but it is not within the set of admissible steady states, making the state technically infeasible. However, the offset constraint of the cost function (5-39) and the terminal set constraint of the optimization problem (5-40)(d) mean that the artificial steady state will drive the system to the neighbourhood of the nearest feasible set point, retaining problem feasibility.

Similarly, the input space should be considered. The evolution in this space is illustrated in the following figure.

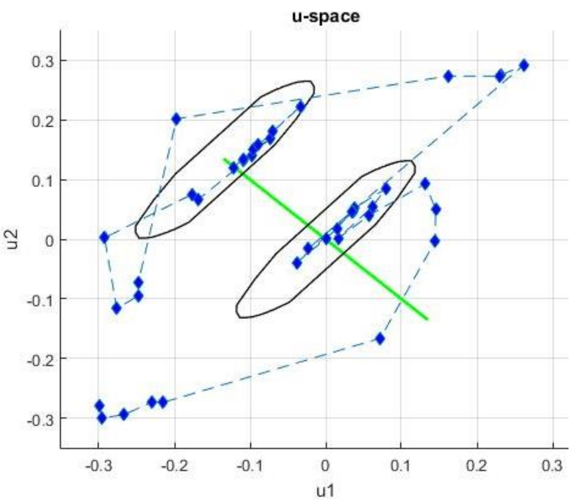


Figure 5-5 Input evolution of pendulum system

The (blue) time series of control inputs are presented here as a form of trajectory. The two regions, marked out in black, indicate the boundaries of the set $u_{s,A} \oplus K_{dr}\mathcal{Z}$ and $u_{s,B} \oplus K_{dr}\mathcal{Z}$ – for the first and second parts of the simulation, respectively – analogously to the yellow tube in *Figure 5-4.* The green line specifies the 1-dimensional set of admissible steady state inputs. The control input generated for the uncertain system at steady state, with any admissible realization of the uncertainty, must exist in the intersection of the of the black region and the green line. Sure enough, the results indicate that the final inputs are located on the green line and within the admissible control input.

Table 5-1 Double pendulum simulation parameters

Parameter	Simulation Value
$\boldsymbol{Q}$	I_2
$\boldsymbol{R}$	$10\, I_2$
$\boldsymbol{P}$	$P_{lqr}: solution\ to\ Riccati\ eq.for\ the\ LQR$
$\boldsymbol{T}$	$1000\, P$
$\boldsymbol{Prediction\ horizon}$	$10\ steps$
$\boldsymbol{Simulation\ horizon}$	$20\ steps$
$\boldsymbol{x_0}$	$\begin{bmatrix} -3 \\ 0.5 \end{bmatrix}$
$\boldsymbol{Steady\ state}$	$\theta_1 = \begin{bmatrix} -4 \\ 0 \end{bmatrix}\ ;\ \theta_2 = \begin{bmatrix} 4 \\ -0.5 \end{bmatrix}$
$\boldsymbol{Disturbance}$	$uniformly\ distributed\ pseudorandom\ noise\ within\ specified\ bounds$

Part 2: Application of tube-based robust MPC to a satellite rendezvous maneuver

6 Platform and RPO formulation

In this rendezvous problem, the goal is to bring a chaser spacecraft from its initial position in space to a docking point on a target spacecraft along a provided reference trajectory while robustly satisfying actuation constraints. To this end, this chapter is dedicated to the definition of the system and an initial characterization of the control problem, with the aid of the preceding theory.

The rendezvous maneuver, considered in 3 dimensions, is to be carried out between a chaser spacecraft and a target satellite. The target craft is considered as a sphere of radius r_t [m] centered on the center of mass of the craft, which tumbles with an angular velocity ω_t. The target body frame follows this same motion. A docking point is located at a position on this sphere defined by $\mathbf{r}_d^{\mathbf{O}}$ in the orbital frame of the target. The orbit of the center of mass of the target is approximated to be circular with orbital radius R_t, measured from the center of the Earth. The relative motion between the spacecraft is then appropriately described using the CWH equations, previously discussed in *chapter 2*. The chaser spacecraft is modelled as a point mass and must approach the target to dock at the docking point. As is typical to the rendezvous phase of RPO maneuvers, the chaser and target spacecraft are within the same orbit. [20, 21]

The reference trajectory is provided by a motion planner, which makes use of a selected inertia and rotational velocity state to determine the trajectory optimized to a set of costs [67]. The motion planner devises a relative trajectory between the two spacecraft with respect to the target orbital frame [67]. However, there is an amount of uncertainty in the motion of the target spacecraft resultant from uncertainty in the inertia or spin state of the target craft. The uncertainty in the motion of the target can be described by an arc drawn on a spherical shell, or the upper most bound of a scaled unitary ball $\mathcal{B}^n = \{b \in \mathbb{R}^n : \|b\|_p \leq 1\}$, as represented in Figure 6-1.

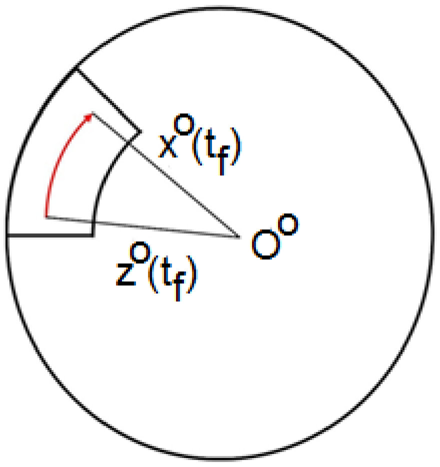


Figure 6-1 Effect of uncertainty in motion of the target on the position of the docking point

The above figure captures a realization of the uncertainty at the final time step in the maneuver. The center of mass of the target is coincident with O^O in the figure. The position of the docking point if nominal motion of the target is followed is indicated by the radius to point z^O. When the nominal motion is additionally affected by an uncertainty, described by the red arc, the docking point will be located at x^O. This uncertainty must be accounted for in the robust control of the chaser to the docking port, and this idea will be returned to later.

Fortunately, the reference trajectory is defined relatively with respect to the two spacecraft, and, under robust reference tracking control methods, it can be validly nominally tracked irrespective of the actual position of the docking point in absolute space, permitting that the actuation constraints are not violated. As the boundary of the uncertainty is characterizable, there exist robust control methods which will allow the reference trajectory to be tracked within this uncertainty bound and actuation constraints. The aim of this work is to design a controller using the theory of Tube-based robust Model Predictive Control for tracking, as presented in the *chapters 4 and 5*, to control the approach of the chaser satellite while guaranteeing constraint satisfaction and tracking the provided reference trajectory. The development of this control problem shall be the topic of the next chapter.

In this chapter, some characteristics of the two spacecraft will be explained, the system model will be presented in state space representation, and the constraints of the system and the uncertainty will be characterized.

6.1 The target satellite

This work tackles the task of bringing a chaser craft to the disabled, tumbling Envisat so that robotic docking and deorbiting maneuvers can be conducted. The rest of this section will outline pertinent details about the satellite in preparation of the following work.

Figure 6-2 Envisat [68]

Envisat, depicted above, is an unresponsive, tumbling satellite which serves as the target in active debris removal scenarios investigated by ESA [69]. Launched in 2002, Envisat was the largest civilian Earth observation mission. In April of 2012, contact with Envisat was unexpectedly lost, and in May, after a month of efforts to regain control of the satellite, the Envisat mission was declared over [68]. Now, Envisat is the largest single piece of space debris owned by ESA. A large body of work has been conducted under the ESA e.Deorbit project, focused on the re- or de-orbiting of the satellite to prevent adverse collisions, destruction, or the creation of more high velocity debris. The work can hopefully be more widely deployed to help clean up the orbits about Earth.

Following from e.Deorbit specifications, Envisat is chosen as the target spacecraft in this work for a number of reasons [69-71]: Envisat is located in the densely populated 600-800 km altitude band near the polar region; maintaining a sun-synchronous orbit at a nominal reference mean orbit altitude of 800 km, or about 7171 km from the center of the Earth. The spacecraft has an approximate mass of 8000 kg. The size and location of the craft gives it the highest risk of collision with other debris out of all ESA owned orbiting objects. The large size of Envisat is also representative of many heavy orbital debris objects, such as late rocket stages. Furthermore, Envisat's complex shape and uncertain tumbling state complicate its capture. This means that in the case of robotic capture, as is of interest here, the position and attitude of the chaser must be synchronised with the motion of the target for capture, while avoiding the solar panel – which is locked into an inconvenient position, partially blocking access to one of the strongest, stiffest, most stable external points on the satellite (see Figure 6-2).

The tumbling nature of the satellite has precluded its attitude evolution from being reliably known, but motion planning methods have been used to estimate the evolution. The estimate commonly uses a satellite rotation at an angular velocity of 3.5 deg/s almost aligned with the orbital momentum [71]. The effects of the Earth's magnetic field on the rotation of the satellite are difficult to determine, but could be slowing it. On the other hand, the spin could also have been slowed or sped up due to impacts with other debris or micrometeorites which have already occurred. [71]

6.2 The chaser satellite

The e.Deorbit project proposed many methods for the de- or re-orbiting of orbital debris (see [69]). This work focuses on a robotics-based proposal. This requires the chaser to approach the docking point on the target such that the terminal state is within a specific distance of the docking point. The re-orbit or de-orbit mission end goal makes no difference as to how the chaser will maneuver to the proximity of the docking port and the scope of this work terminates with the end of the rendezvous phase – that is, at the beginning of the docking procedures.

Under the body of the e.Deorbit project, the selected grasping or docking point used in the robotic docking designs varies by research group [69]. These scenarios are illustrated in the following figure.

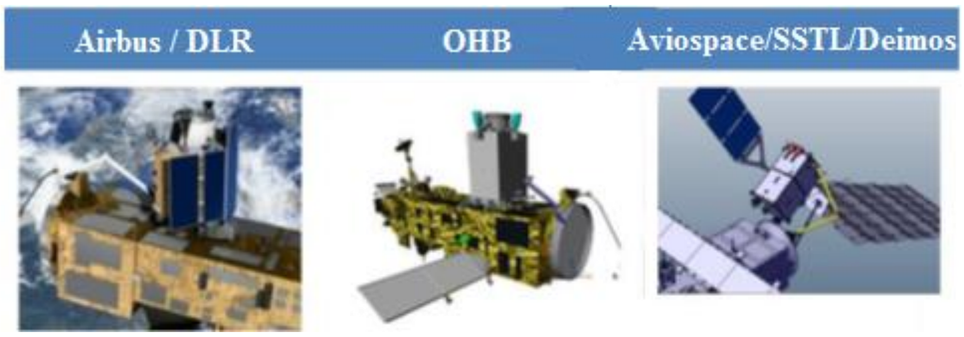


Figure 6-3 Grasping point proposals under the e.Deorbit Phase A, adapted from [69]

The Airbus Defence and Space and DLR partnership proposed grasping Envisat's Hold Down Release Mechanism (HDRM). OHB systems devised a method which suggested clamping onto the sides of the target, such that the chaser is held to the top face of the target. SSTL, Aviospace, and Deimos devised a clamping mechanism for the Launch Adapter Ring (LAR) of Envisat.

In this work, a motion planner designed for optimizing a rendezvous path to Envisat is utilised to provide the optimal trajectory from the initial position of the chaser to the target. The motion planner was devised with docking through grasping the Launch Adapter Ring (LAR) in mind. By the end of Phase A of the ESA project, the LAR had been agreed upon as the designated grasping point, as the it can withstand large forces and the structure is a common feature for many other possible target satellites [69]. This would permit the extension of the development conducted under this project to be easily applied to other missions with little to no redesign of the docking procedures. It should also be noted that the trajectories provided by motion planner were created with hardware imposed constraints on the accessible thruster force and velocity of $[-65 \quad 65]\,N$ and $[-1 \quad 1]\,m/s$, respectively. These constraints cannot be violated in the control implementation as the real system would not be capable of supplying more thrust force or attaining greater velocities.

It should also be noted that the proposed chaser spacecraft will make use of LiDAR and a multispectral camera to determine the relative position of the chaser to the target. As of the publication of [69], this was still an open task. Nonetheless, this work can assume the provision of these measurements, which is vital to the relative motion control task.

6.3 The rendezvous model

As a circular orbit is approximated for the progression of the relative motion system, the CWH equations will be used as outlined in *chapter 2* to set out the system model. In accordance with this preceding exposition, the x-axis refers to radial or R-bar direction, the y-axis to the along-track or V-bar direction, and the z-axis to the cross-track or H-bar direction. The cross-track dynamics have often been ignored in previous works, for example [16-18, 20, 21], and the problem constrained to the radial/along-track-plane, as the cross-track dynamics are decoupled from the interdependent radial and along-track dynamics. In this work, however, the third dimension is included, expanding on the works [15, 20-24, 41]. The nominal system model is here developed.

The target is on a circular orbit about the Earth, with an orbital rate $n = \sqrt{\mu/R_t^3}$ (rad/s), where μ is the gravitational constant of the Earth and $R_t = 7171$ [km]. The origin of the reference Hill frame is affixed to the center of mass of the target and revolves with orbital rate n with respect to the inertial reference frame centered on the center of the Earth. Following from the development in *section 2.2* up to the definition of the *set of equations* (2-5), the base-model for the relative motion problem is given in continuous time by

$$\dot{z}(t) = A_c z(t) + B_c v(t) \tag{6-1}$$

where the nominal state vector

$$z = \begin{bmatrix} \delta x^O & \delta y^O & \delta z^O & \delta \dot{x}^O & \delta \dot{y}^O & \delta \dot{z}^O \end{bmatrix}^T \tag{6-2}$$

within which, $x^O, y^O,$ and z^O refer to the axes of the target orbital frame, the nominal input vector

$$v = \begin{bmatrix} v_x^O & v_y^O & v_z^O \end{bmatrix}^T \tag{6-3}$$

the components of which correspond to the accelerations applied to translational thrusters of fixed-direction of the chaser, and

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & 3n^2 & 0 & 0 & 0 \end{bmatrix} \tag{6-4}$$

$$B_c = \begin{bmatrix} 0_{3\times 3} \\ I_3 \end{bmatrix} \tag{6-5}$$

As a discrete-time model is desired, this system is discretized and becomes

$$z(k+1) = Az(k) + Bv(k) \tag{6-6}$$

which is discretized using the sampling time T_s.

Note that the system retains its linear dependence on the input control actions. Under MPC terminology, therefore, linear control methods are applicable to this problem. This system model will be used in the control strategy following.

6.4 Rendezvous constraints

As has been indicated in the *Part 1* of this work, constraints are integral to the development of MPC problems and system description. Thus, system constraints will be enumerated here, but will be numerically defined later in the controller and simulation designs.

The state and input constraints for the above system are fairly simple to outline. The first three components of the system state correspond to the position and the next three to the velocity of the chaser relative to the target. It is desirable that these system properties remain within hard limits and are set using inequality constraints in the form of polyhedral set constraints, which will be defined later in *chapter 6*. The acceleration inputs are physically constrained by the nature of the thrusters utilized by the chaser spacecraft. These constraints can be defined by an inequality constraint

$$u_{min} \leq u \leq u_{max} \tag{6-7}$$

where u_{min} is the minimum applicable thrust and u_{max} the maximal. In case, the thrusters are characterized by their thrust force capabilities in newtons. The lower and upper constraints on the acceleration inputs must then be determined using

$$\frac{F_{min}}{m_c} \leq u \leq \frac{F_{max}}{m_c} \tag{6-8}$$

where $F_{(\cdot)}$ is derived from the force available from the thruster and m_c is the mass of the chaser spacecraft, to which the thrusters are attached.

In RPO problems, there are typically some further constraints, namely line of sight and soft-docking constraints. However, a line of sight constraint is typically only useful in the case of a cooperative target and the soft docking constraint is simply an additional acceleration constraint which limits the relative acceleration over time, with the aim that the relative acceleration between the target and the chaser at the time of docking is negligible. The same outcomes are achieved by requiring the system to follow a provided, pre-determined optimal trajectory. The use of reference tracking techniques imposes further constraints on the system to limit the discrepancy between the current system state and the desired trajectory state. The form of the model will then constrain the position and velocity of the chaser relative to the target.

6.5 Uncertainty characterization

The goal of Tube-based robust MPC is to steer a disturbed system in an admissible evolution to an admissible steady state while guaranteeing robust satisfaction of the state and control constraints in the presence of a possibly unknown, but bounded, uncertainty. As was shown in *chapters 4 and 5*, the definition of the uncertainty is important to the design of the control law. To this end, the uncertainty pertaining to this problem will now be characterized.

6.5.1 Uncertainty boundary

Previous applications of these control methods to satellite rendezvous have concentrated on navigation and thruster timing errors. In this application, the uncertainty lies in the inertia and spin state of the target spacecraft. This uncertainty has been characterized simply as an angular uncertainty, described as in the axis-angle representation. That is, the uncertainty can be visualized as forming an arc along the surface of a sphere centered on the origin of the orbital frame (refer to Figure 6-1). Therefore, as the target moves through space, the uncertainty introduced into its motion will not result in some large deviation of its path, but rather a deviation of the docking point from the expected point on the sphere over-bounding the satellite to some other point along the surface of the sphere [72]. This will be shown to be the case with the following.

First, it is necessary to determine how the target will behave under the presence of a disturbance in its inertia or spin state. Consider, then, the general problem, where the point mass chaser would be brought together with the center of mass of the target, as illustrated in the following image.

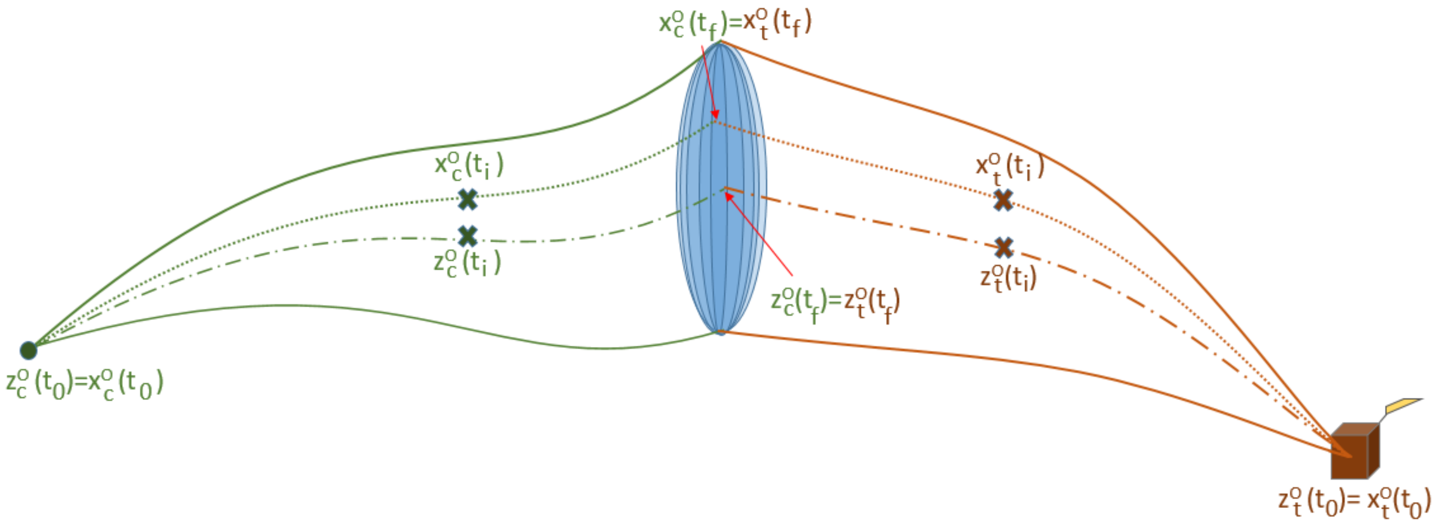


Figure 6-4 Nominal and disturbed motion

The spacecraft are depicted moving to a common position in space at which they are to rendezvous. This may be a possible view external to this defined system. In reality, it is assumed that the spacecraft are in the same orbit, travelling in the same direction, with the chaser spacecraft initially at a known position phased a known distance behind the target craft. In the above illustration, x_c^O and x_t^O represent the actual positions of the chaser and target in the target orbital frame (refer to Figure 2-2) and z_c^O and z_t^Orepresent the evolutions of the nominal positions of the chaser and target in the target orbital frame when all uncertainty is ignored. The nominal position of the chaser relative to the target z_r^O is equivalent to the expected state dictated by the reference trajectory. If the uncertainty in the target's motion was not present, the motion of the chaser relative to the target would follow this trajectory. However, because of the presence of the uncertainty, the chaser will deviate from this path, but still track the shape of the reference trajectory which is defined based on the relative motion of the target and chaser. The trajectory will lie within the tube extending from the initial position of the chaser toward the terminal set where the target and chaser meet. This is the same concept described in *section 4.3*. The evolution of the set of desired terminal states is described by the possible motion of the docking point through space.

Now, the relationship between frames of the chaser and the target needs to be shown. First, note that

$$z_c^O(0) - z_t^O(0) = z_r^O(0) \qquad (6\text{-}9)$$

$$z_c^O(t_f) - z_t^O(t_f) = 0 \qquad (6\text{-}10)$$

where the initial time $t_0 = 0$, t_f is the final time of the maneuver. Therefore, it can clearly be understood that

$$z_c^O(t_i) - z_t^O(t_i) = z_r^O(t_i) \tag{6-11}$$

which simply states that the difference of the positions of the chaser and the target in the undisturbed system is the nominal relative position at time t_i as posed in the orbital frame. Clearly, the goal is to determine the position of the chaser in the presence of uncertainty. The uncertainty $\pm\Delta x_{(\cdot)}^O$ is incorporated as follows

$$z_c^O(t_i) \pm \Delta x_c^O(t_i) = x_r^O(t_i) + z_t^O(t_i) \pm \Delta x_t^O(t_i) \tag{6-12}$$

such that the nominal and real states of each body are related by

$$x_{(\cdot)}^O(t_i) = z_{(\cdot)}^O(t_i) \pm \Delta x_{(\cdot)}^O(t_i) \tag{ 6-13}$$

Now, the inertia of the target is defined in the body frame of the target. This is advantageous to motion planning and other processes, as the property is constant in this frame. The target motion, incorporating the uncertainty, must be transformed from the body frame into the inertial frame. This is written as

$$z_c^O(t_i) \pm \Delta x_c^O(t_i) = x_r^O(t_i) + A^{bO}(\phi) x_t^b(t_i) \tag{6-14}$$

with superscript b indicating the target body-fixed frame, $A^{bI}(\phi)$ is the rotation matrix which transforms the body frame to the inertial frame, and $x_t^b(t_i)$ indicating the target motion in the target body-fixed frame at time t_i incorporating the uncertainty, as in the illustration above. It is not the subject of this thesis to fully characterise this rotation matrix, but is used here as a device to illustrate the basis of the evolution of the uncertainty. It is desired to describe the subject of the rotation matrix ϕ. This is easiest understood from the mathematical basis of the description of the motion of a rigid body. Consider then, the equation of motion of the rigid body of the target:

$$I\dot{\omega} + \omega \times I\omega = 0 \tag{6-15}$$

where I is the inertia matrix of the rigid body and ω the angular velocity. Let $\bar{I} = I \pm \Delta I$ describe the inertia of the target incorporating the uncertainty. Then the equation of motion becomes

$$\bar{I}\dot{\omega} + \omega \times \bar{I}\omega = 0 \tag{6-16}$$

Rewriting this in terms of $\dot{\omega}$ gives an integrable equation such that

$$\omega(\bar{I}) = \int \dot{\omega}\, dt = \int \bar{I}^{-1}\, (-\omega \times \bar{I}\omega)\, dt \tag{6-17}$$

The subject of the rotation matrix is an angular position. However, the angular velocity ω cannot be integrated in time to obtain a time evolution of the orientation of the target. To obtain this evolution, the equation

$$\dot{\phi} = C^{-1}(\phi)\omega(\bar{I}) \tag{6-18}$$

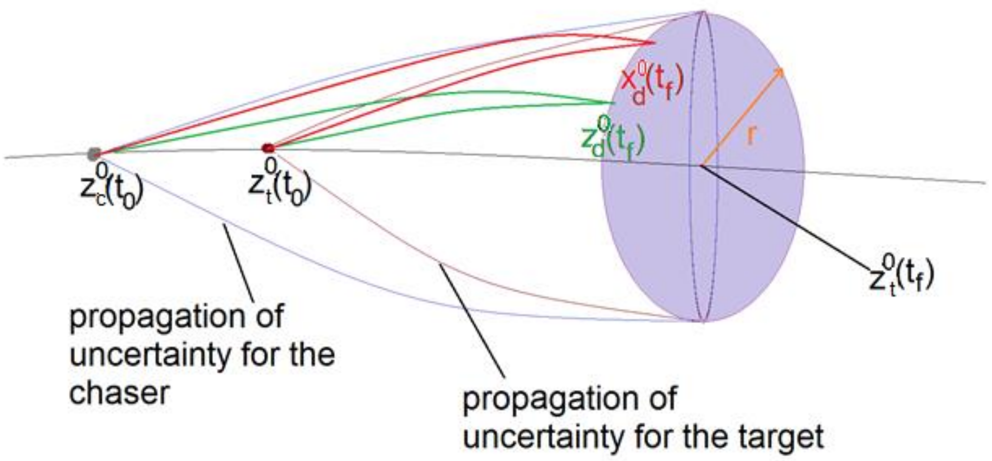
is used, where the matrix $C(\cdot)$ is a function of ϕ, which be determined through

$$\phi(\bar{I}) = \int \dot{\phi}(\bar{I})\, dt = \int C^{-1}(\phi)\, \omega(\bar{I})\, dt \tag{6-19}$$

Therefore, the transformation matrix $A(\phi)$ is dependent on the inertia, which simply influences a rotation in the inertial frame.

Now, recalling the definition of the chaser position incorporating uncertainty (6-12), it can easily be understood that the chaser position uncertainty is directly influenced by the uncertainty in the inertia of the target, and can similarly be defined.

The problem at hand requires that the chaser meets a docking point on the target, rather than the target's center of mass. The control problem is therefore a tracking problem rather than a regulation problem. The above result is extended to the present problem, and is be understood with the aid of the following figure.



$x_d^0(t_f)$
$z_d^0(t_f)$
r
$z_c^0(t_0)$
$z_t^0(t_0)$
$z_t^0(t_f)$
propagation of uncertainty for the chaser
propagation of uncertainty for the target

Figure 6-5 Propagation of uncertainty

By virtue of the preceding exposition, the worst-case uncertainty bound of the docking point can be defined in the orbital frame as a unitary norm ball, scaled by the radius of the sphere over-bounding the target – that is $r\mathcal{B}^n$, or $4.6662\,\mathcal{B}^n$ for this problem. The radius is determined from the magnitude of the vector displacing the grasping point from the center of mass of the target. The spherical bound is drawn by the motion of the docking point with every realization of the worst-case uncertainty in the motion of the target. The relevant final positions of the chaser are located on this sphere, represented by the purple sphere in Figure 6-5, when the nominal motion of the target brings the center of mass of the target to $z_t^0(t_f)$. Of course, it was shown above that the same would be true for the position of the disturbed $x_t^0(t_f)$ – as only the pose of the target will change. In this figure, the center of mass of the target and the nominal position of the docking point are therefore differentiated. The latter is noted by z_d^0 for some possible predicted motion of the target. The position of the docking point in the presence of a non-zero disturbance of the target motion is denoted by x_d^0 . Trajectories which could possibly be followed by the docking point and the chaser to the nominal and disturbed set points are given in green and red, respectively.

The trajectories of the docking point to any worst-case uncertainty set point will be contained in the propagation of uncertainty of the target. As in [37], the worst-case uncertainty, or likewise the uncertainty boundary, for the chaser can be translated or propagated back from the docking point uncertainty back to the chaser and implemented as a state disturbance. This uncertainty bound can thus be likewise defined using the $\infty - norm$ of the scaled sphere. As the motion of the chaser is to be controlled in this task, this is the desired uncertainty bound to be defined in the controller design – the exact set definition shall be addressed in the next chapter. Luckily, this is a typical polytopic form for the uncertainty in problems controlled through this method – which requires the uncertainty to be bounded, closed, and convex – and literature suggests this to be the easiest form to handle in the progression of the problem.

6.5.2 Disturbances applied to the system

As presented in *chapter 4,* the disturbed system is given by

$$x^+ = Ax + Bu + w \tag{6-20}$$

where a state disturbance presents some unknown, but bound, additional actuation to the system. This must be distinguished from the instigating position disturbance. Let this position disturbance be denoted by $\delta\mathrm{x}$. For this application, the current state disturbance $\delta\mathrm{x}$ is given by the difference between the nominal state z defining the position and velocity of the chaser relative to the target and the relative state of the chaser measured by instrumentation located on the chaser x.

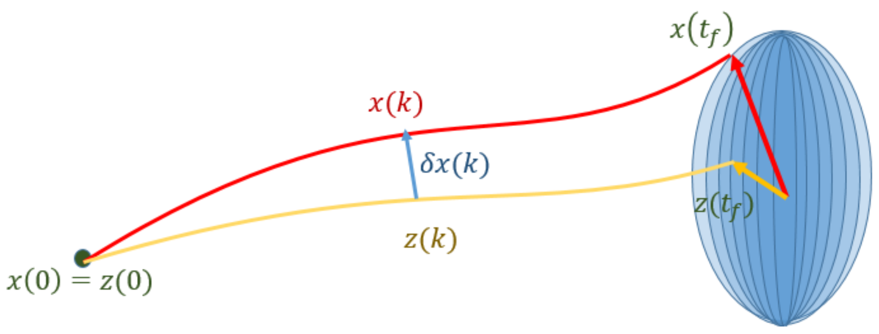


Figure 6-6 Position disturbance at time k

The figure illustrates that the real state of the chaser relative to the docking point, measured by instrumentation on-board the chaser, differs from the reference at time k any an amount δx, that is

$$\delta x = x - z \tag{6-21}$$

This disturbance will in some manner instigate an additional actuation to the state equation to obtain the real system (6-20). To see how this effect is introduced, predict the next real state of the chaser:

$$x^+ = z^+ + \delta x^+ \quad (6\text{-}22)$$

$$= Az + Bv + \delta x^+ \quad (6\text{-}23)$$

$$= A(x - \delta x) + Bv + \delta x^+ \quad (6\text{-}24)$$

$$= Ax - A\delta x + Bu - BK_{dr}\delta x + \delta x^+ \quad (6\text{-}25)$$

$$= Ax + Bu + (\delta x^+ - A\delta x - BK_{dr}\delta x) \quad (6\text{-}26)$$

$$= Ax + Bu + w \quad (6\text{-}27)$$

As will be discussed more in the next chapter, there is assumed to be only a position disturbance applied to the docking point. It can then be shown that for a position disturbance of

$$\delta x = [\Delta x \quad \Delta y \quad \Delta z \quad 0 \quad 0 \quad 0]^T \quad (6\text{-}28)$$

the state disturbance

$$w = \delta x^+ - A\delta x - BK_{dr}\delta x \quad (6\text{-}29)$$

with A and *B* as above and K_{dr} determined in the next chapter, reduces to

$$w = \delta x \quad (6\text{-}30)$$

This is an important result which will be made use of later.

7 Controller formulation

In general terms, MPC is a frequently chosen method for the control of satellite rendezvous as it has the particular advantage that robust control ideas – including system and control constraints and the robust satisfaction thereof – can be easily incorporated [2, 13, 15, 52, 73], as should be now evident. Furthermore, the pointwise-in-time treatment and the inherently slow nature of the spacecraft rendezvous problem, as well as its expression as a linear model, make the problem a suitable candidate for MPC [15]. Furthermore, MPC controllers have the distinct advantages of being inherently closed loop in operation and that the model or circumstantial specifications can be done remotely, where infrastructure permits, and can even be done on the fly [50].

Reconsider, then, the system to be controlled: In summary, a chaser spacecraft is required to rendezvous with a target spacecraft utilizing their relative dynamics for navigation. A pre-determined reference trajectory is provided by a motion planner tailored to this process. This trajectory is an optimized path from a start position to the specified final position – which coincides with the point where the docking procedure is set to begin. The chaser is required to follow this provided reference trajectory while robustly satisfying the system constraints and converging to the defined terminal state.

The goal of the control problem is thus to determine a sequence of control actions which guarantee rendezvous precision while tracking the provided reference. It is crucial that the actuator constraints are satisfied and the size of the arrival set should be minimized to benefit the rendezvous precision. Therefore, the control strategy which follows makes use of tube-based robust MPC for tracking to robustly satisfy constraints, track a reference trajectory, and satisfy the requirements of an RPO problem. This chapter will detail how tube-based robust MPC for tracking can be used in the framework of this problem. The implementation of this design process in software is provided in Appendix part B. Satellite Rendezvous controller design.

7.1 The system

The undisturbed nominal system for this task can be described by the rendezvous model described in *section 6.3*

$$z^+ = Az + Bv \tag{7-1}$$

where the state $z \in \overline{\mathbb{X}} \subseteq \mathbb{R}^6$, successive state $z^+ \in \overline{\mathbb{X}} \subseteq \mathbb{R}^6$, and control input $v \in \overline{\mathbb{U}} \subseteq \mathbb{R}^3$ vectors and the state A and input B matrices are defined as in *section 6.3.* When the orbital rate n is chosen to match that used by the motion planner, n and the discretized system matrices at a sampling time of $T_s = 0.5\ s$ become

$$n = 0.0012\ [rad/s] \tag{7-2}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0.5 & 0.003 & 0 \\ -2.16\times10^{-10} & 1 & 0 & -0.0003 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.5 \\ 2.16\times10^{-10} & 0 & 0 & 1 & 0.0012 & 0 \\ -1.296\times10^{-9} & 0 & 0 & -0.0012 & 1 & 0 \\ 0 & 0 & -7.2\times10^{-7} & 0 & 0 & 1 \end{bmatrix} \tag{7-3}$$

$$B = \begin{bmatrix} 0.125 & 5\times10^{-5} & 0 \\ -5\times10^{-5} & 0.125 & 0 \\ 0 & 0 & 0.125 \\ 0.5 & 0.0003 & 0 \\ -0.0003 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{7-4}$$

This model describes the normal expected dynamics of the rendezvous maneuver. However, under normal operation, the chaser will deviate from this trajectory due to disturbances, and the perturbed system is represented by

$$x^+ = Ax + Bu + w \tag{7-5}$$

again with the state $x \in \mathbb{X} \subseteq \mathbb{R}^6$, successor state $x^+ \in \mathbb{X} \subseteq \mathbb{R}^6$, and control $u \in \mathbb{U} \subseteq \mathbb{R}^3$ vectors defined analogously to (7-1) and the state A and control B matrices defined as in (7-3) and (7-4), respectively.

The output $y \in \overline{\mathbb{X}} \subseteq \mathbb{R}^6$ vector and output C and feedthrough D matrices are now brought into consideration:

$$y = Cx + Du \tag{7-6}$$

As it is desirable to track all components of the state and no feedthrough is introduced, the respective matrices are given by

$$C = I_6 \tag{7-7}$$

$$D = \mathbf{0}_{6\times3} \tag{7-8}$$

The state disturbance w is bounded and belongs to the polytopic compact set $\mathcal{W}$. Under the framework of Tube-based robust MPC, the boundary of this uncertainty is required. As elaborated in *section 6.5,* the boundary for the uncertainty in the relative position of the chaser can be described by the upper most bound of the scaled unitary ball $r_t\mathcal{B}^n = \{b \in \mathbb{R}^n : \|b\|_\infty \le r_t\}$. It is assumed that there is no uncertainty on the relative velocity. With the radius of the docking port from the center of mass of the target being $r_t = 4.66627\ m$, the relevant bound of the disturbance is described by

$$\mathcal{W} = \begin{Bmatrix} \{b \in \mathbb{R}^3 : \|b\|_\infty \le 4.66627\} \\ \mathbf{0}_{3\times1} \end{Bmatrix} \tag{7-9}$$

This set can be visualized as follows.

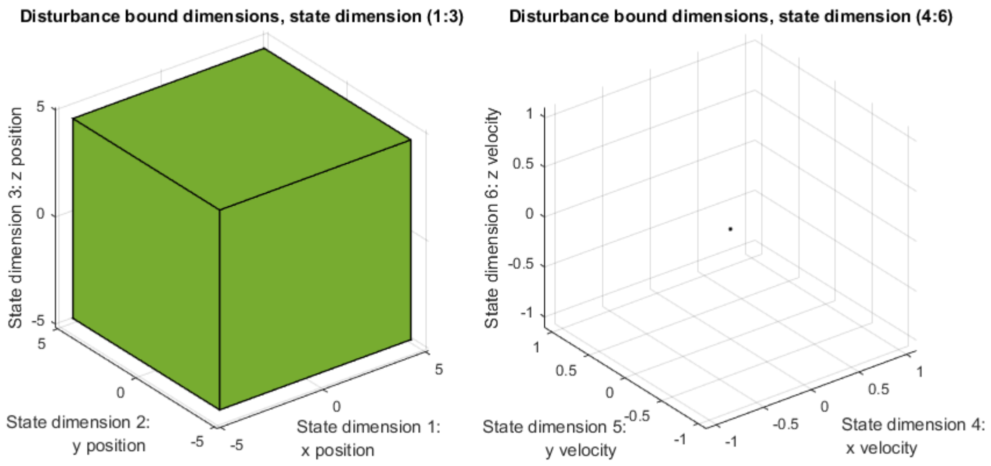


Figure 7-1 Disturbance bound, (left) dimensions 1-3, (right) dimensions 4-6

Figure 7-1 represents the set boundaries for the system states rather than the true bounds of the disturbance in space. The left figure depicts the elements of the state vector corresponding to the maximum disturbance in relative position. The set is defined using the infinity-norm; this set thus translates to maximal disturbance in each dimension, which is required for the subsequent calculations. The implications should be clear. Each position component is limited in the range $[-4.66627 \quad 4.66627]$ m. The right figure indicates the boundary on elements of the state vector corresponding to the relative velocity of the spacecraft.

Finally, the state and control constraints can be defined. The robust constraints can be defined for velocity states and input accelerations based on the hard constraints outlined in *section 6.2.* The position state constraints are chosen to be within the common range of a rendezvous maneuver, as discussed in *chapter 2*. Therefore, the robust state and control constraints are polyhedral, defined in their H-representations as

$$\mathbb{X} = \left\{ x \in \mathbb{R}^6 \mid \begin{bmatrix} I_6 \\ -I_6 \end{bmatrix} x \leq \begin{bmatrix} 100 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \\ 100 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \end{bmatrix} \right\} \tag{7-10}$$

$$\mathbb{U} = \left\{ u \in \mathbb{R}^3 \mid \begin{bmatrix} I_3 \\ -I_3 \end{bmatrix} u \leq [0.043\overline{3} * \mathbf{1}_{6\times1}] \right\} \tag{7-11}$$

The position constraints are chosen from the definition of the terminal rendezvous phase in *section 2.1*. The maneuver may begin at any point within these bounds.

These sets are defined using the Multi-Parametric Toolbox, as in *chapter 5*, for the subsequent set operations. Lines 32-89 in Appendix part B. Satellite Rendezvous controller design define these constraints. The nominal system constraints will be determined in the following as the appropriate robust sets are determined.

7.2 Minimal robust positively invariant set

Recall that there are several steps in the approximation of the mRPI set. The algorithm for each step has been laid out in *section 5.2,* and the full process need not be reiterated here. Rather reference will be made to this preceding exposition and the application of the process in software. Therefore, at the end of this section, the disturbance rejection gain K_{dr} and the mRPI set $\mathcal{Z}$ will be known.

First, the disturbance rejection gain is determined using optimization problem given by equation (5-16), implemented using the CVX toolbox as in *section 5.2.1* in lines 99-142 and 469-616 of Appendix part B. Satellite Rendezvous controller design.

To determine the S-procedure parameter λ, a form of L-curve parameter determination was implemented. The obtained curve is presented in the following figure.

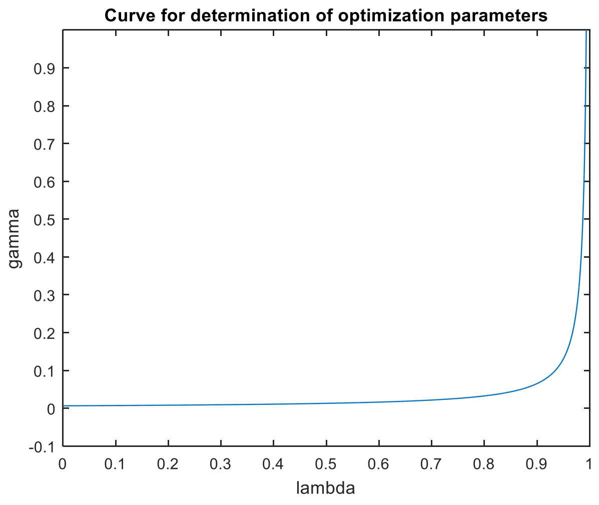


Figure 7-2 Determination of appropriate λ for minimization of γ

There exists significant variance in the optimization parameter γ present in the curve before knee. Therefore, a value of lambda was selected from along the curve where the variance in γ becomes insignificant.

The parameter is then set to $\lambda = 1\times10^{-5}$. The design parameter ρ for the weighing of the actuation constraint (refer to *section 5.2.1*) is arbitrarily set to $\rho = 0.4$, as the selection makes insignificant difference to the disturbance rejection gain or the tightening to the actuation constraints. This test was conducted using lines 633-765 in Appendix part B. Satellite Rendezvous controller design.

Applying these choices to the optimization problem in (5-16), the disturbance rejection gain is found to be

$$K_{dr} = \begin{bmatrix} -4.32\times10^{-6} & 0 & 0 & 0 & -2.4\times10^{-3} & 0 \\ 0 & 0 & 0 & 2.4\times10^{-3} & 0 & 0 \\ 0 & 0 & -4.32\times10^{-6} & 0 & 0 & 0 \end{bmatrix} \tag{7-12}$$

Now, (7-3),(7-4), and (7-12) are substituted respectively into $A_{K,dr} = A + BK_{dr}$, and $A_{K,dr}$, $\mathcal{W}$ defined as in (7-9), and the error bound $\epsilon = 0.01$ are respectively passed as inputs or set as a parameter to Algorithm 5-1. Initially the same error bound from *chapter 5*. Altering this value made no noticeable difference to performance or result, and so the bound was left unchanged. In this case, the algorithm returns $s = 3$. Passing these parameters to Algorithm 5-2 returns the mRPI set $\mathcal{Z}$ given below. These steps correspond to lines 144-217 in Appendix part B. Satellite Rendezvous controller design.

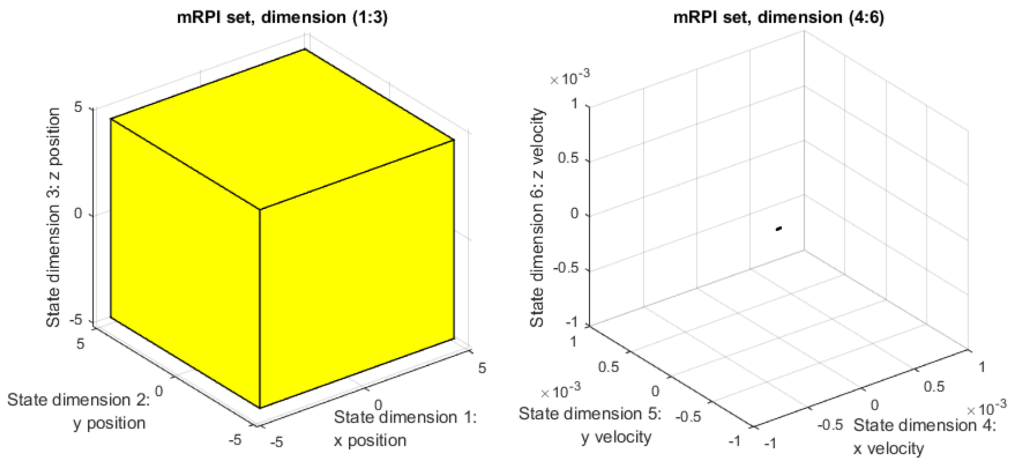


Figure 7-3 Minimal Robust Positively Invariant set for the rendezvous maneuver, (left) dimensions 1-3 corresponding to relative position, (right) dimensions 4-6 corresponding to relative velocity

The plot on the left of Figure 7-3 depicts the first three dimensions of the mRPI corresponding to the relative position states. The plot on the right depicts the same for the last three dimensions corresponding to the relative velocities. In the long term, this means that the actual relative position can be at a position ± 4.6662 m in any dimension relative to the nominal relative position due to the disturbance. This very small difference to the disturbance bound is due to the very small magnitude of the values in A as compared to the magnitude of the disturbance bound given by $\mathcal{W}$, from which this set is determined. As there is to be no disturbance to the nominal in the relative velocity, it is expected these dimensions of the mRPI are of very little contribution.

The H-representation of the mRPI is given as an intermediary result in the following which will be referenced in *section 7.5*

$$\mathcal{Z} = \left\{ x \in \mathbb{R}^6 | \begin{bmatrix} I_6 \\ -I_6 \end{bmatrix} x \leq \begin{bmatrix} 4.666200621821147 * \mathbf{1}_{3\times1} \\ \mathbf{0}_{3\times1} \\ 4.666200621821147 * \mathbf{1}_{3\times1} \\ \mathbf{0}_{3\times1} \end{bmatrix} \right\} \tag{7-13}$$

7.3 Nominal constraints

The nominal constraints can now simply be determined. Recall from *sections 4.3* and *5.2.4* the relationship between the nominal and robust constraints, re-produced here:

$$\bar{\mathbb{X}} = \mathbb{X} \ominus \mathcal{Z} \tag{7-14}$$

$$\bar{\mathbb{U}} = \mathbb{U} \ominus \oplus K_{dr} \mathcal{Z} \tag{7-15}$$

From the preceding results, the robust state constraints determined using lines of Appendix part B. Satellite Rendezvous controller design are given by their H-representations as

$$\bar{\mathbb{X}} = \left\{ z \in \mathbb{R}^6 \mid \begin{bmatrix} I_6 \\ -I_6 \end{bmatrix} z \leq \begin{bmatrix} 95.333799378178853 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \\ 95.333799378178853 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \end{bmatrix} \right\} \tag{7-16}$$

Similarly, for the robust input constraints,

$$\bar{\mathbb{U}} = \left\{ v \in \mathbb{R}^3 \mid \begin{bmatrix} I_3 \\ -I_3 \end{bmatrix} v \leq \begin{bmatrix} 0.043313142013789 \\ 0.043333300000000 \\ 0.043313142013532 \\ 0.043313142013789 \\ 0.043333300000000 \\ 0.043313142013532 \end{bmatrix} \right\} \tag{7-17}$$

Practically, the nominal constraints for each state and input is indicated in the following table:

Table 7-1 Nominal state and input constraints

Constrained aspect	**Minimum constraint**
States 1-3: relative x, y, and z position [m]	± 95.333799378178853
States 4-6: relative x, y, and z velocities [m/s]	± 1
Input 1: x acceleration [m/s^2]	± 0.043313142013789
Input 2: y acceleration [m/s^2]	± 0.043333300000000
Input 3: z acceleration [m/s^2]	± 0.043313142013532

As can be seen from Table 7-1 the nominal state constraints as compared to the robust constraints have been tightened by the margin granted by the mRPI set, or in the case of the input constraints by that dictated by the mRPI set scaled by the disturbance rejection gain. It should be noticed that none of the hardware constraints mentioned in *section 6.2* are violated by these robust constraints. The ensuing controller is therefore constraint feasible.

7.4 Invariant set for tracking and admissible sets

Recall from *sections 4.4.3, 5.3,* and *5.4* that the invariant set for tracking takes the place of the terminal set in tracking applications and is determined using numerous parameters. In this section, the determination of this set is outlined.

First, the LQR gain K_{lqr} and terminal weighting P_{lqr} are determined using the MATLAB function *dlqr.* Recall from *section 5.4.1* that this function takes the system dynamics and the weighting matrices Q and R as inputs and the obtained K_{lqr} must be negated. Initial values of the weighting matrices were chosen; then tuned based on performance and values cited in literature for MPC of rendezvous maneuvers [15, 20-22] to

$$Q = 10\times10^{5} * I_6 \tag{7-18}$$

$$R = 100 * I_3 \tag{7-19}$$

The obtained LQR parameters are

$$K_{lqr} = \begin{bmatrix} -4.319784011\times10^{-6} & 0 & 0 & 0 & -0.002399880006 & 0 \\ 0 & 0 & 0 & 0.002399880005482 & 0 & 0 \\ 0 & 0 & -4.319784011\times10^{-6} & 0 & 0 & 0 \end{bmatrix} \tag{7-20}$$

$$P_{lqr} = 1\times10^{6} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.000000000575978 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.000000000575970 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{7-21}$$

These will be reference later in section 7.5.

Then, the terminal states and set points are parameterized using the method expressed in chapter 4 to obtain the matrices M_θ and N_θ.

$$M_\theta = \begin{bmatrix} -0.185551526369166 & 0.018117071393572 & 0.047743479379178 & 0.706328547648448 & 0.005820151954137 & 0.032651318626743 \\ -0.002300266364531 & -0.182713737993363 & 0.060394004339685 & -0.007604460874062 & 0.706008414024590 & 0.038656066811025 \\ -0.051013496976408 & -0.057658466070907 & -0.176380728691161 & -0.032282424810556 & -0.038964665502406 & 0.705293981181349 \\ 0.185551526369166 & -0.018117071393571 & -0.047743479379178 & 0.706328547648448 & 0.005820151954136 & 0.032651318626743 \\ 0.002300266364531 & 0.182713737993363 & -0.060394004339685 & -0.007604460874062 & 0.706008414024590 & 0.038656066811025 \\ 0.051013496976408 & 0.057658466070907 & 0.176380728691161 & -0.032282424810556 & -0.038964665502406 & 0.705293981181349 \\ 0.185546807312485 & -0.018555662630504 & -0.047598740020594 & 0.706343747015220 & 0.004125706617421 & 0.032558403012700 \\ 0.002745590027817 & 0.182670257022019 & -0.060508588690195 & -0.005909272359706 & 0.706022382389280 & 0.038734429975730 \\ 0.051013717354715 & 0.057658715155481 & 0.176381490655910 & -0.032282285350481 & -0.038964497175050 & 0.705290934311349 \end{bmatrix} \tag{7-22}$$

$$N_{theta} = \begin{bmatrix} -0.556654579107498 & 0.054351214180714 & 0.143230438137534 & 0.706328547648448 & 0.005820151954136 & 0.032651318626743 \\ -0.006900799093592 & -0.548141213980089 & 0.181182013019056 & -0.007604460874062 & 0.706008414024590 & 0.038656066811025 \\ -0.153040490929224 & -0.172975398212721 & -0.529142186073484 & -0.032282424810556 & -0.038964665502406 & 0.705293981181349 \\ 0.556654579107498 & -0.054351214180714 & -0.143230438137534 & 0.706328547648448 & 0.005820151954137 & 0.032651318626743 \\ 0.006900799093592 & 0.548141213980089 & -0.181182013019056 & -0.007604460874062 & 0.706008414024590 & 0.038656066811025 \\ 0.153040490929224 & 0.172975398212721 & 0.529142186073484 & -0.032282424810556 & -0.038964665502406 & 0.705293981181348 \end{bmatrix} \tag{7-23}$$

The parameterization method is implemented in software in Appendix part B. Satellite Rendezvous controller design in lines 238 and 298-329.

The system dynamics and parameters K_{lqr} and M_θ are then used to form the closed loop system for the extended state

$$\begin{bmatrix} x \\ \theta \end{bmatrix}^+ = \begin{bmatrix} A + BK_{lqr} & BL \\ 0 & I_6 \end{bmatrix} \begin{bmatrix} x \\ \theta \end{bmatrix} \tag{7-24}$$

with $L = [-K_{lqr} \quad I_3] M_\theta$, such that $x^{e+} = A_e x^e$. Finally, the parameters K_{lqr} and $\lambda_{MRPI} = 0.99$, the matrices M_θ and A_e, and the nominal constraints are passed to Algorithm 5-4 to obtain the invariant set for tracking. The scaling parameter λ_{MRPI} is chosen arbitrarily close to 1 as the design methodology allows. The first six dimensions of set pertain to state space and the next six to the parameterized θ-space.

The invariant set for tracking Ω^e describes the terminal constraint for each prediction step of the nominal controller. For the nominal states in this problem, this means that the final step of each prediction horizon must lie within the projection of this set into state space Ω_x, given by

$$\Omega_x = \{x \in \overline{\overline{\mathbb{X}}} | 1\times10^3 *$$

$$\left(\begin{bmatrix} 5.246339631\times10^{-6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001219679685189 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.001219679685189 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.017090189112486 \\ 0 & 0 & 0 & 0 & 0 & 5.15069147091\times10^{-4} \\ 0 & -0.017090189112502 & 0 & 0 & 0 & 0 \\ 0 & 5.246339631\times10^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.017090189112502 & 0 & 0 & 0 \\ 0 & 0 & 5.246339631\times10^{-6} & 0 & 0 & 0 \\ -0.017090189112502 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -8.47376763781\times10^{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 8.47376763781\times10^{4} & 0 \end{bmatrix} x \le \begin{bmatrix} 0.000500153489806 \\ 0.001219679685189 \\ 0.001217523942679 \\ 0.017090189112486 \\ 0.000515069147091 \\ 1.629272660186442 \\ 0.000500153489806 \\ 1.629272660186442 \\ 0.000500153489806 \\ 1.629272660186443 \\ 0.000847376763781 \\ 0.000848420429103 \end{bmatrix} \right) \Bigg\} \tag{7-25}$$

The state-related dimensions of the invariant set for tracking is of the same magnitude as the nominal set constraint. Practically, this implies that any nominal position or velocity is a permissible state at the end of the prediction horizon.

Similarly, for the parameterized θ-space:

$$\Omega_\theta = \{\theta \in \mathbb{R}^6 | 1\times10^3 *$$

$$\left(\begin{bmatrix} 0.001189681852441 & 0.00134464866731 & 0.004113361452975 & -0.00075284944968 \\ 0.004327223038681 & -0.0004225058681 & -0.001113419479532 & -0.01647219629 \\ -9.83220914\times10^{-7} & 9.6\times10^{-8} & 2.5298842\times10^{-7} & 3.74277169\times10^{-6} \\ 5.3644213025\times10^{-5} & 0.004261043344668 & -0.001408440728517 & 0.000177342644035 \\ -1.2188905\times10^{-8} & -9.68183727\times10^{-7} & 3.2\times10^{-7} & -4.0295357\times10^{-8} \\ 6.4029363914\times10^{-5} & 0.004260017061755 & -0.00141110887128 & -0.000137808976049 \\ -0.004327116638817 & 0.000432734562646 & 0.001110044968777 & -0.016472564657425 \\ -2.7031595\times10^{-7} & -3.05527046\times10^{-7} & -9.34625678\times10^{-7} & -1.71061677\times10^{-7} \\ -0.001189688705037 & -0.001344656412518 & -0.004113385146035 & 0.000752853786114 \\ 0.001189679135061 & 0.001344645595966 & 0.004113352057549 & 0.000752854234714 \\ -6.39702969\times10^{-5} & -0.004256087201038 & 0.001409807125011 & 0.000137681847432 \\ 0.00432711663882 & -0.000432734562646 & -0.001110044968778 & 0.016472564657436 \end{bmatrix}\right. \tag{7-26}$$

$$\left.\begin{bmatrix} -0.000908684126199 & 0.016447964758324 \\ -0.000135731007539 & -0.000761457159473 \\ 3.084\times10^{-8} & 1.73\times10^{-7} \\ -0.016464730495353 & -0.000901493111712 \\ 3.741\times10^{-6} & 2.04835\times10^{-7} \\ 0.016465008830619 & 0.000903318007343 \\ -9.6215\times10^{-5} & -0.000759290927449 \\ -2.0647\times10^{-7} & 3.737289615\times10^{-6} \\ 0.000908689360242 & -0.016448059498994 \\ 0.000908689901699 & -0.016448069299844 \\ -0.016449819879387 & -0.000902484697546 \\ 9.6215149211\times10^{-5} & 0.000759290927450 \end{bmatrix} x \leq \begin{bmatrix} 0.000999997120004 \\ 2.201034477567907 \\ 0.000500113608763 \\ 2.201034477567907 \\ 0.000500113608763 \\ 0.001000461675305 \\ 0.001 \\ 0.000500113608763 \\ 0.001 \\ 2.201034477568637 \\ 0.000999538750589 \\ 0.001 \end{bmatrix}\right)\Bigg\}$$

The polytope indicated in the above set of inequalities indicates the terminal constraints for the parameter θ. As this parameter exists in parameterized space, it does not have an exact physical meaning. It does mean that the artificial steady state in the offset cost term of the objective function must be optimized to exist within this set.

The software implantation of the Invariant Set for Tracking is implemented in the lines of 240-251 of Appendix part B. Satellite Rendezvous controller design.

Incidentally, in this task, the nominal region of attraction is similarly sized:

$$\bar{\mathbb{X}}_{ra} = \left\{ \begin{bmatrix} I_6 \\ -I_6 \end{bmatrix} x \leq \begin{bmatrix} 95.333799378178867 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \\ 95.333799378178853 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \end{bmatrix} \right\} \tag{7-27}$$

It would be expected, then, that the region of attraction for the robust states would be similarly sized to the robust state constraint:

$$\mathbb{X}_{ra} = \left\{ \begin{bmatrix} I_6 \\ -I_6 \end{bmatrix} x \leq \begin{bmatrix} 100 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \\ 100 * \mathbf{1}_{3\times1} \\ \mathbf{1}_{3\times1} \end{bmatrix} \right\} \tag{7-28}$$

Recalling that a large region of attraction is one of the benefits of tube-based robust MPC, these preceding results indicate that the design process has yielded control gains and constraints which produce such desirably large regions of attractions.

The set of admissible terminal states is given by

$$\mathbb{X}_a = \left\{ \begin{bmatrix} I_6 \\ -I_6 \end{bmatrix} x \le \begin{bmatrix} 95.3337993781788 * \mathbf{1}_{3\times1} \\ 0.043828731493523 * \mathbf{1}_{3\times1} \\ 95.3337993781788 * \mathbf{1}_{3\times1} \\ 0.043828731493523 * \mathbf{1}_{3\times1} \end{bmatrix} \right\} \tag{7-29}$$

Recall from the steady state/set point parameterization that the steady states are defined for all states and inputs. The steady states of the states and inputs must reside in these sets. The practical implications of the above is that the states pertaining to relative position and the inputs may reach steady states at any point within the nominal constraints. The velocity steady state set, on the other hand, is tighter than the nominal constraint, with each velocity steady state constrained to ± 0.043 m/s.

These sets are determined in lines 253-267 of Appendix part B. Satellite Rendezvous controller design.

7.5 Tube-based robust model predictive controller

Now that all of the necessary parameters have been determined, the tube-based robust model predictive controller can be defined. The nominal controller is considered first, then integrated into the robust controller.

Recall from *section 5.5* that the objective function is of the form

$$V_N(x,\theta;z_0,\mathbf{v},\bar{\theta}) = \sum_{i=0}^{N-1} (\|z(i) - x_{ref}(i)\|_Q^2 + \|v(i) - u_{ref}(i)\|_R^2) + \|z(N) - x_{ref}(N)\|_{\mathrm{P}}^2 + \|\bar{\theta} - \theta_{ref}\|_T^2 \tag{7-30}$$

which is minimized in the optimization problem

$$\min_{z,\mathbf{v},\theta} V_N(x,\theta;z_0,\mathbf{v},\bar{\theta}) \tag{7-31}$$

$$s.t.\quad z \in x \oplus (-\mathcal{Z}) \tag{a}$$

$$z(i) \in \bar{\mathbb{X}}, \qquad i = 0,\dots,N \tag{b}$$

$$v(i) \in \bar{\mathbb{U}}, \qquad i = 0,\dots,N-1 \tag{c}$$

$$(z(N),\bar{\theta}) \in \Omega^e \tag{d}$$

After some tuning, the design parameters were chosen as summarized in Table 7-2. In this table, large matrices and sets which have been previously defined are indicated simply by their equation number.

Table 7-2 Nominal predictive controller design parameters

Parameter	Simulation Value
State weighting matrix Q	$10 \times 10^5 * I_6$
Input weighting matrix R	$100\ I_3$
Terminal weighting matrix P	P_{lqr} (7-21)
Offset weighting matrix T	$1000\ P$
LQR gain K_{lqr}	K_{lqr} (7-20)
Nominal state constraints $\bar{\mathbb{X}}$	$\bar{\mathbb{X}}$ (7-16)
Nominal input constraint $\bar{\mathbb{U}}$	$\bar{\mathbb{U}}$ (7-17)
mRPI $\mathcal{Z}$	(7-13)
Invariant set for tracking Ω^e	Ω_x (7-25), Ω_θ (7-26)
Prediction horizon N	$40\ steps$
Sample time T_s	$0.5\ s$

The predictive controller makes up a large part of the robust controller, handling the most calculation intensive part of the control action. The following figure demonstrates how the predictive controller factors into the robust model predictive control action.

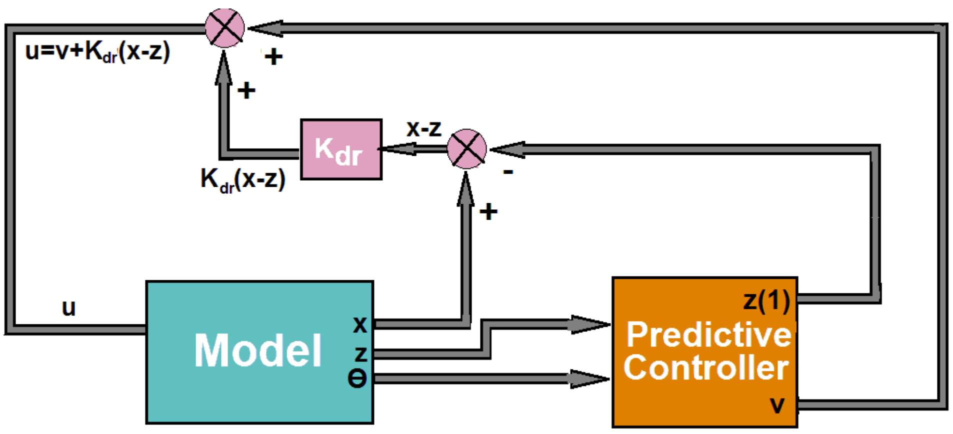


Figure 7-4 Block diagram of the robust controller

As can be seen, to complete the robust controller, the nominal input must be slightly modified. To do this, the first predicted state in the series determined by the optimization problem z must be subtracted from the actual state x, scaled by the disturbance rejection gain K_{dr}, and finally added to the nominal optimal control input v. It should be remembered that the output of the model is equal to the system state. The robust input

$$u = v + K_{dr}(x - z) \tag{7-32}$$

is then passed to the system model to close the control loop. It is important to ensure that the robust control constraints are not violated.

The remaining parameters pertinent to the robust controller are outlined in Table 7-3.

Table 7-3 Additional robust controller design parameters

Parameter	Simulation Value
Disturbance rejection gain $\boldsymbol{K_{dr}}$	K_{dr} (7-12)
Robust state constraints $\bar{\mathbb{X}}$	$\mathbb{X}$ (7-10)
Robust input constraint $\bar{\mathbb{U}}$	$\mathbb{U}$ (7-11)

This controller is initialized in software using in lines 330-468 of Appendix part B. Satellite Rendezvous controller design. Now that the robust controller has been fully formulated, the next chapter is devoted to the formulation a MATLAB/Simulink simulation.

8 Simulation

In this chapter, a simulator for the tube-based robust model predictive controller designed in *chapter 7* is developed. For the purposes of generality, the formulation will only discuss the structure of the simulator; specific simulation values will then be applied to obtain the reported results.

In *chapter 5,* the controller development and simulation was conducted using only the Multi-parametric and the CVX toolboxes. However, the satellite rendezvous control task requires reference tracking and a modified objective function and constraint set as compared to standard MPC, which requires nuances of the MPT to be influenced in a manner which have not yet been reliably manipulated by the author. Therefore, it was decided to implement this simulation using the Model Predictive Control Toolbox. Much like the MPT, the Model Predictive Control Toolbox, allows the user to create a custom nominal model predictive controller, with the desired system constraints, prediction and terminal weights, and set custom terminal constraints for a desired system model. However, unlike the MPT, the Model Predictive Control toolbox very clearly and transparently handles reference trajectories.

In the rest of this chapter, the simulator will be formulated and explained and applied to a set of simulation parameters, and the simulation results will be analysed.

8.1 Simulator formulation

Figure 7-4 serves as a basic schematic for the simulator, but a few more aspects need to be considered for simulation using the Model Predictive Control toolbox. The modified block diagram is shown below.

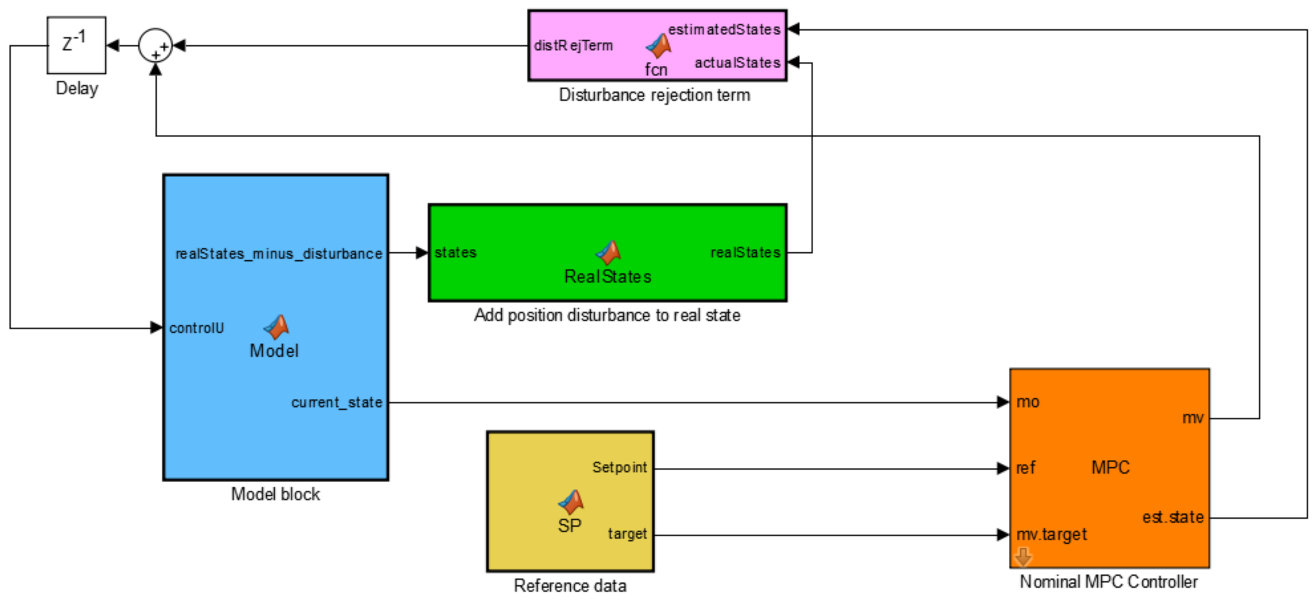


Figure 8-1 Block diagram for simulation in Simulink

The above figure shows the five major parts to the simulation: the system model, the set point disturbance, the nominal MPC controller, the construction of the robust control law, and the reference trajectory. Each of these is discussed in the following.

The orange MPC controller block refers to the nominal MPC Controller block from the set of Model Predictive Control Toolbox Simulink functions. This block serves as the prediction controller in the simulation. The block applies an mpc object from the MATLAB workspace as a model predictive controller on the inputs to the block at the current state node and outputs the optimized control action. The reference node refers to the state references and takes in a reference vector the length of the prediction horizon. Likewise, the target node refers to the target values of the control action and requires a vector the length of the prediction horizon. The reference and target nodes are built-in required attributes of the controller which permit tracking of the reference trajectory. In this application, the estimated state viewing option is also activated.

The yellow reference trajectory block parses a loaded reference file provided by the motion planner to obtain the reference and target vectors at each simulation step. The block provides both a reference and a target the length of the prediction horizon to the controller at each simulation step.

The robust control law is constructed from the outputs of the controller block summed together with the disturbance rejection term provided by the pink disturbance rejection term block. The pink block takes in the current state provided by the system model block and the estimated state vector from the nominal controller. The first state in the series of estimated states is subtracted from the current real state, and the difference is multiplied by the disturbance rejection gain K_{dr} (7-12) determined in *chapter 7.* The robust control input is passed to the blue system model block.

The system model block handles the coordination of the nominal and disturbed system models subject to the robust control inputs and provides the model outputs. At each time step the real and nominal system outputs are determined and sent to the appropriate subsequent block.

Now recall the discussion on the form of the bounded state disturbance from *section 6.5.2*. While the state disturbance can be shown to be of the same form as the set point disturbance in this problem, in these simulations the bound disturbance is applied as set point disturbance δx rather than the additive disturbance w. This mimics where the uncertainty enters the system. The addition of the disturbance to the state is therefore removed from the model block to an external summation conducted by the green block in the figure above.

As noted in the above, the MPC block makes use of a mpc object defined in the MATLAB workspace as the designated model predictive controller. This mpc object is created using the Model Predictive Control Toolbox for a specific system model, sampling time, and prediction and control horizons. Once the controller object is created for the nominal system (7-1) and (7-6), the various system attributes can be set. In this case, the weighting matrices, system and input constraints, terminal constraints, and terminal sets are defined as indicated in Table 7-2 and Table 7-3. The Model Predictive Control Toolbox also makes provision for a user defined initialization of the controller. The initial conditions are taken from the reference trajectory for the initial plant and last move arguments, respectively corresponding to the initial state and inputs, and from the problem definition to define the initial disturbance and noise.

8.2 Common simulation components

Several simulations were conducted using the above simulator to test the capability of the controller to robustly guide the chaser to the correct point on the target. Each of the simulations makes use of the same simulator. The only changes made between simulations pertains to the disturbance w and the setpoint. This section details the definition of the common components.

The simulations of course make use of a common controller. The robust control law uses the disturbance rejection gain K_{dr} (7-12). The nominal mpc controller object is defined as indicated in the preceding section, using the parameters $\overline{\mathbb{X}}, \overline{\mathbb{U}}, Q, R, P, T,$ and Ω^e determined in *chapter 7.* The initial conditions of the controller are taken from the motion planner as

$$x = [39 \quad 39 \quad 4.23 \quad 0 \quad 0 \quad 0]^T \tag{8-1}$$

$$u = [0 \quad 0 \quad 0]^T \tag{8-2}$$

with initial noise and disturbances set to 0. The Simulink simulator and the MATLAB function called in the execution are included in Appendix part C. Satellite Rendezvous simulator. The common functions are given in lines 1-385. As this thesis is not a tutorial on the Model Predictive Control Toolbox, the reader is referred to the Model Predictive Control Toolbox manual and MATLAB documentation for further details [9, 10].

Each simulation also makes use of the same reference trajectory. This is done to maintain uniformity of conditions across the trials and to give meaning to the results. The nominal set point is given by this reference trajectory as

$$x = [1.97 \quad 0 \quad 4.23 \quad 0 \quad 0 \quad 0]^T \tag{8-3}$$

$$u = [0 \quad 0 \quad 0]^T \tag{8-4}$$

The nominal control action should drive the nominal state and inputs from the initial condition in (8-1) and (8-2) to the steady states values in (8-3) and (8-4), while following the reference between. Following from the definition of the disturbance (see sections 6.5 and 7.1), the robust position states are permitted to deviate from the nominal relative position trajectory but should maintain the same target accelerations and reference velocities.

The reference trajectories provided by the motion planner are of more conservative constraints than the nominal constraints $\bar{\mathbb{X}}$ and $\bar{\mathbb{U}}$ determined in the preceding chapter. Both the nominal and robust operations should therefore be constraint feasible. To ensure that no constraints are violated, constraint handling is included in the system model block. This also means that there exists a much larger set of points which are reachable without breaching the input constraints. As has been previously discussed, though, there exists a finite set of trajectories which can be robustly controlled while tracking a specific trajectory under the nominal dynamics. For a successful rendezvous, and therefore a truly robust tracking, the real terminal state of the relative position should lie on the sphere over bounding the target described in *section 6.5.*

8.3 Simulation results and analysis

In this section, several simulations are presented: The first simulation verifies the nominal operation of the controller. Then, three simulations of various guaranteed-to-work set points are presented to verify correct operation of the robust controller. Finally, two simulations of points which demonstrate the limitations of the controller are presented to demonstrate non-compliance with terminal conditions or failed docking. Additional simulation results are presented for the case of shortened rendezvous times.

Recall that the goal of Tube-based robust MPC is to steer a disturbed system in an admissible evolution to an admissible final state – that is, points contained within (7-29) – while guaranteeing robust satisfaction of the state and control constraints in the presence of a possibly unknown, but bounded, uncertainty. In each of the simulations a set point and disturbance are defined. In this task, the terminal set point in the target orbital frame is inherently defined by the manifestation of the uncertainty. Referring to *section* 6.5, the arc prescribed on the over-bounding sphere starts at the nominal set point and ends at this new set point. For the purposes of analysis, it is assumed that the uncertainty is known. In the following, it is assumed that the desired set point in is known before the start of the rendezvous maneuver and remains constant until the end of the maneuver. The disturbance in the set point has been distributed linearly along the trajectory as the state disturbance δx, rather than w, as discussed in *section* 6.5 and *8.2*. The rest of this chapter presents the simulation results.

8.3.1 Nominal operation

In this first simulation, the disturbance is set to zero. The resulting system response is the nominal response for the setpoint (8-3). The state and input responses are as follows.

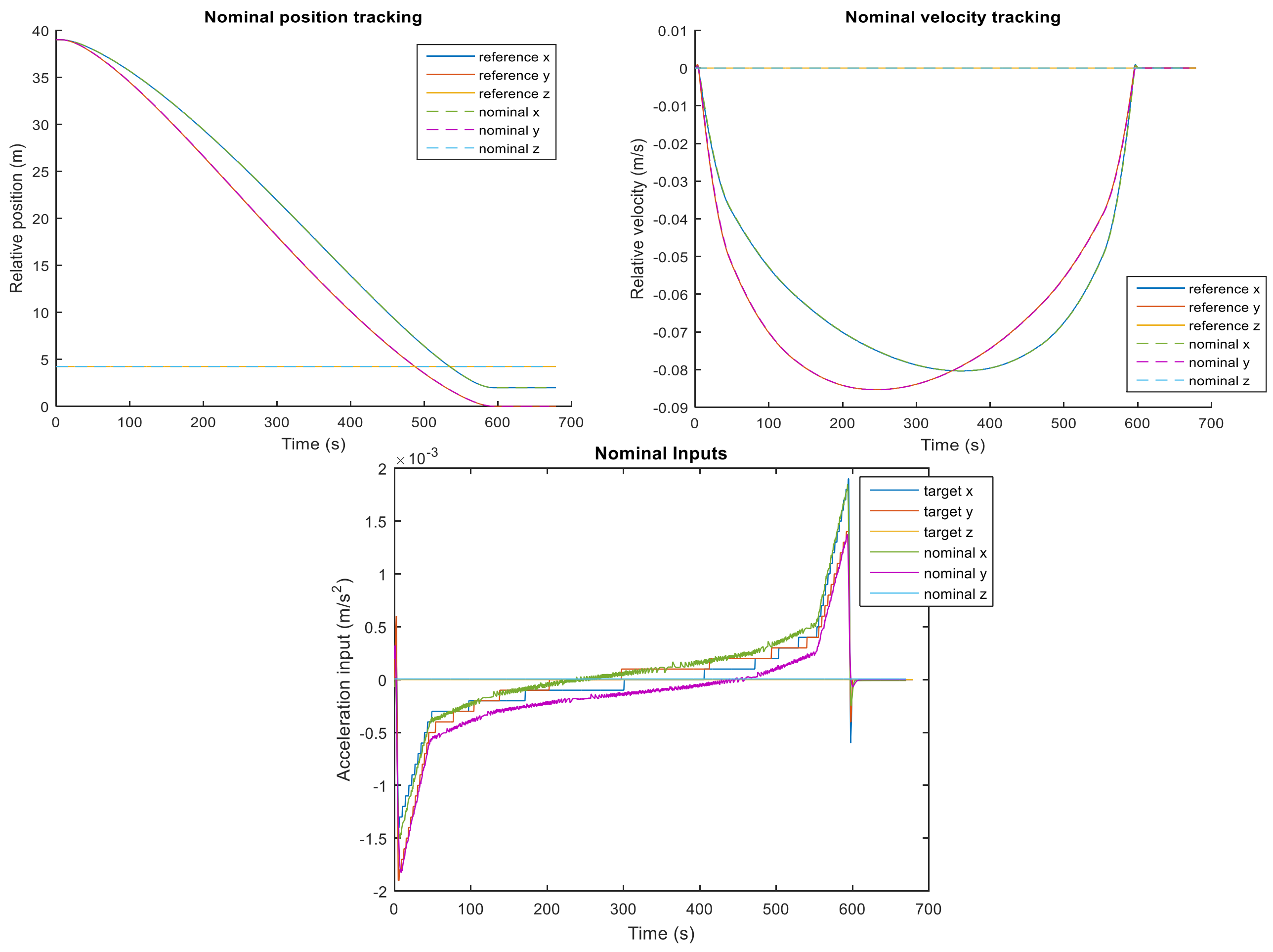


Figure 8-2 Simulation of the nominal system, (left) relative position states, (right) relative velocity states, (below) control inputs

In the timeseries in the preceding figure, the nominal state and input results from the simulator, are plotted along with the reference and target series, respectively. The reference series are plotted in dark blue for the radial, red for the in-track, and yellow for the cross-track directions, and the simulated series are respectively green, purple, and light blue. As can be seen, the controller produces a series of inputs which very closely track the target values and shape of the series. This results in nominal state progressions which very clearly track the reference trajectory. With satisfactory nominal control, the robust controller has a solid foundation on which to build.

8.3.2 Robust operation with successful rendezvous

Three simulations were conducted in demonstration of successful rendezvous maneuvers. The three set points were chosen to show the robust control capabilities as defined in the tube-based robust MPC framework. Each of the points are therefore selected on the spherical shell on which the docking port could exist if the origin of the frame were coincident with the center of the target. The set points chosen for simulation are depicted in the following figure.

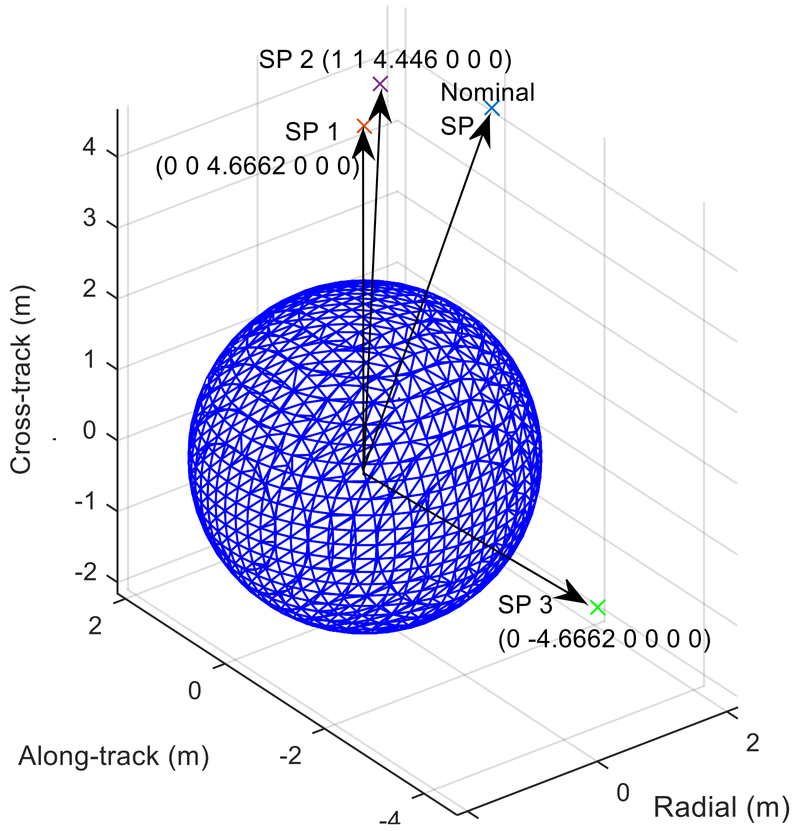


Figure 8-3 Set points used in simulations with successful rendezvous

The sphere is reduced to a unit sphere for the visibility of the relationship of the set points. The first two points that will be simulated, SP1 and SP2, are located close to the nominal set point. The third set point, SP3, is a greater distance from the nominal, near the boundary of the tube. The results of the simulations to these set points are presented in the following.

In the first of these four simulations, the disturbance is given by

$$\delta x = [-1.64\times10^{-3}\,t \quad 0 \quad 3.63\times10^{-4}\,t \quad 0 \quad 0 \quad 0] \tag{8-5}$$

where t indicates the simulation time step, and a corresponding set point of

$$x(t_f) = [0 \quad 0 \quad 4.6662 \quad 0 \quad 0 \quad 0] \tag{8-6}$$

The set point is located near the nominal steady state described – refer to Figure 8-3 – and is therefore located solidly in the terminal tube set.

As it is desired that the form of the nominal trajectory should be followed by the robust trajectory, the relative position states are studied. First, consider the nominal and real trajectories returned by the simulation.

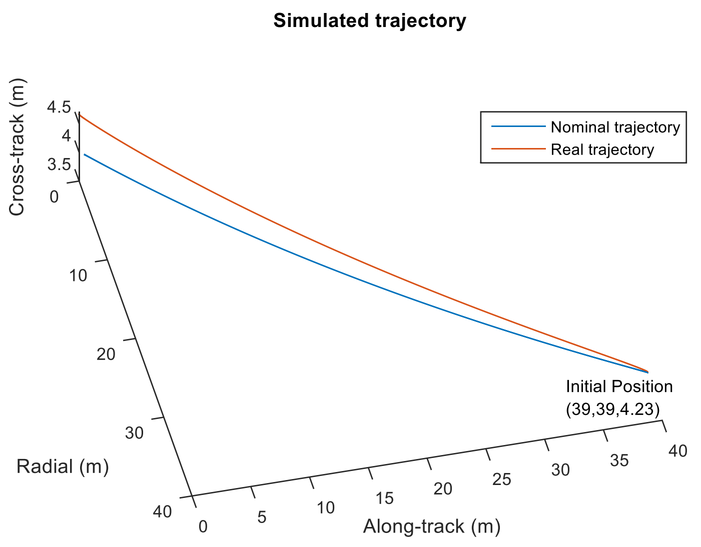


Figure 8-4 Simulated trajectories for steady state (0,0,4.6662,0,0,0)

In this plot, the nominal is given by the blue trajectory and the real trajectory by the red. Clearly the form of the real trajectory, controlled by the full robust controller, maintains the form of the nominal trajectory. This is a touch complex to view in three dimensions. In the following, the trajectory is broken down in to one-dimensional timeseries.

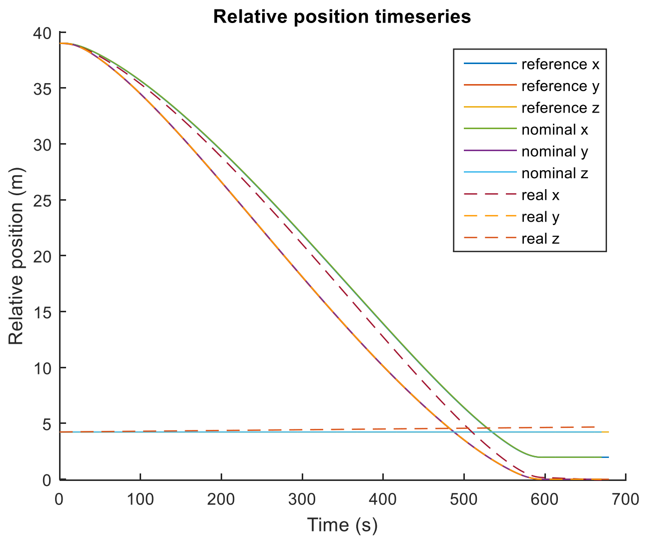


Figure 8-5 Relative position timeseries for steady state (0,0,4.6662,0,0,0)

In this figure, the reference and nominal components are represented as in Figure 8-2 and the real position components by the red, yellow, and orange dashed series. Clearly, each real component follows the form of its corresponding reference to the new steady state.

Furthermore, the containment of the real trajectory within the tube can be shown.

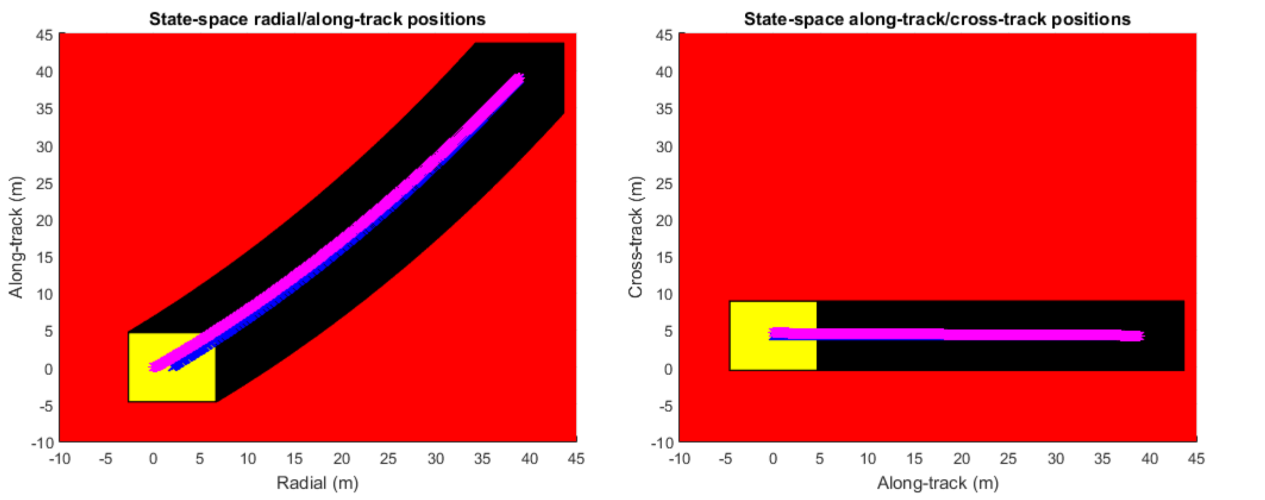


Figure 8-6 Location of the trajectories within the tube, (left) radial/along-track dynamics, (right) along-track/cross track dynamics, steady state (0,0,4.6662,0,0,0)

In Figure 8-6, red indicates the set of admissible steady states, the black prism with a yellow front face represents the tube of trajectories, the blue line represents the nominal trajectory, and the magenta the real trajectory. Recall from *section 7.4* that the set of admissible states will extend far past the presented axes. As can be seen here, the robust trajectory is contained well within the tube boundary indicating the trajectories whose evolution can be robustly controlled by the controller.

For completeness, the velocity states should also be presented.

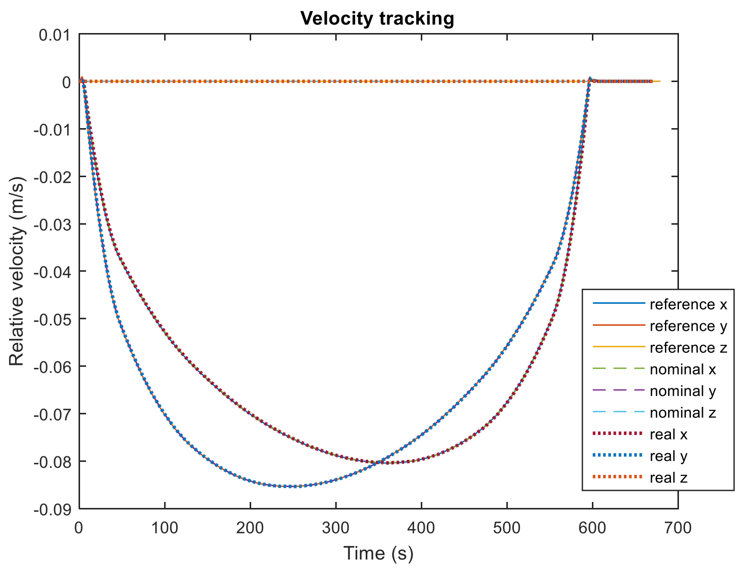


Figure 8-7 Velocity tracking for steady state (0,0,4.6662,0,0,0)

In Figure 8-7, the only addition as compared to the nominal simulation Figure 8-2 is the dotted robust values. The robust relative velocity values track the nominal. Clearly, no substantial difference exists between the nominal and true values. This is due to the zero-disturbance applied to the velocity-related states. With no disturbance applied to the velocity states, there should exist a zero-difference between the nominal and robust relative velocities.

Finally, the robust control action is displayed below.

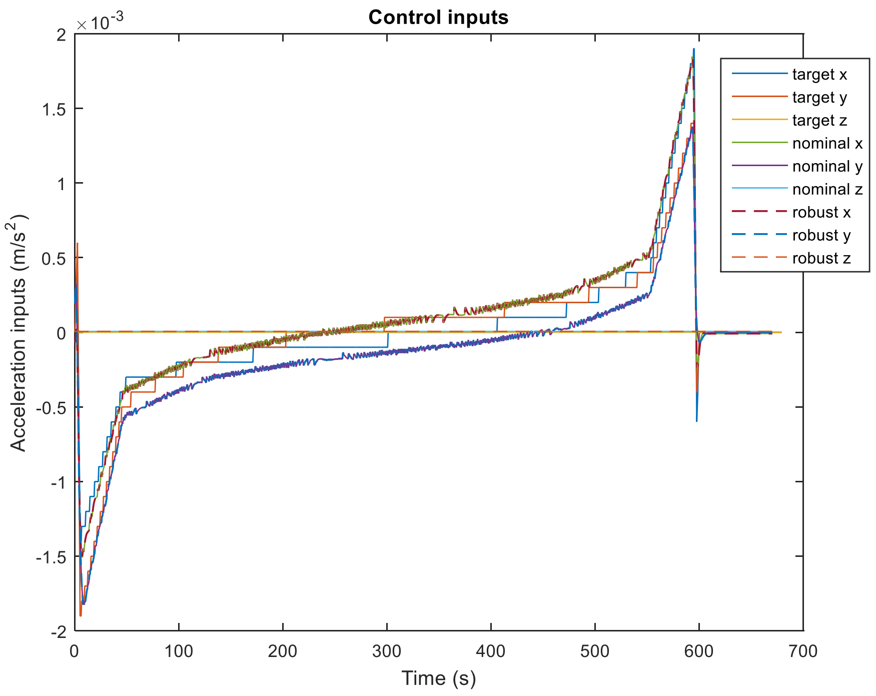


Figure 8-8 Control action for steady state (0,0,4.6662,0,0,0)

The target and nominal inputs are identical to those in Figure 8-2. Figure 8-8 additionally displays the robust control actions as the dashed series. As desired to maintain the properties of the reference trajectory and anticipated, the robust control inputs track the form of the targets. However, to produce the real position trajectory, an amount dictated by the disturbance rejection term is added to the nominal inputs. This is a fairly small additional acceleration:

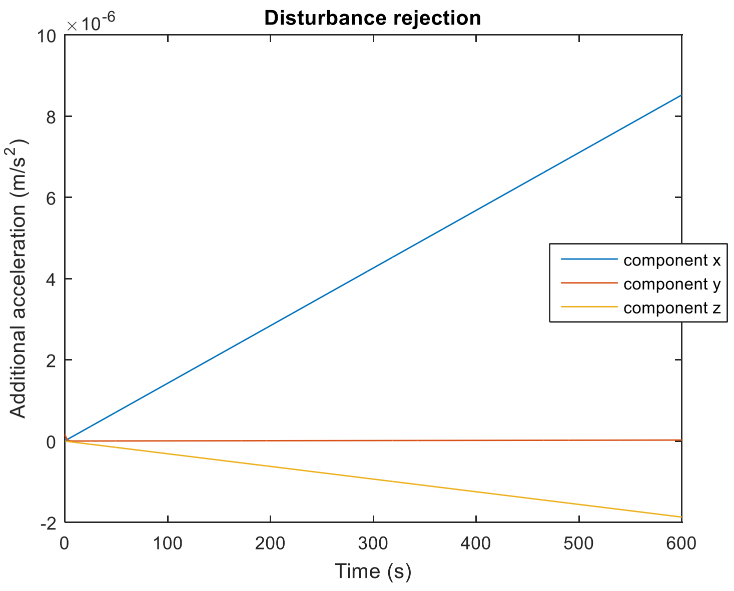


Figure 8-9 Additional acceleration from disturbance rejection term for steady state (0,0,4.6662,0,0,0)

With a simulation duration of 600 s and the very small state disturbance (8-5), this small additional term is all that is required to make up the difference in position of the set points. Validation through inverse dynamics will provide support for this finding in *section 8.3.5*.

In the next simulation, the disturbance is given by

$$\delta x = [-8.076{\times}10^{-4}\,t \quad 8.326{\times}10^{-4}\,t \quad 1.798{\times}10^{-4}\,t \quad 0 \quad 0 \quad 0] \tag{8-7}$$

and a corresponding steady state of

$$x(t_f) = [1 \quad 1 \quad 4.446 \quad 0 \quad 0 \quad 0] \tag{8-8}$$

The final state is also located near the set point described by the nominal and therefore solidly in the terminal tube set. As this final state is rather close to the preceding, it would be expected that the results are similar. This shall be shown in the following.

As before, the first step will be to confirm that the form of the nominal trajectory is followed by the robust trajectory. Therefore, consider the nominal and real trajectories returned by the simulation.

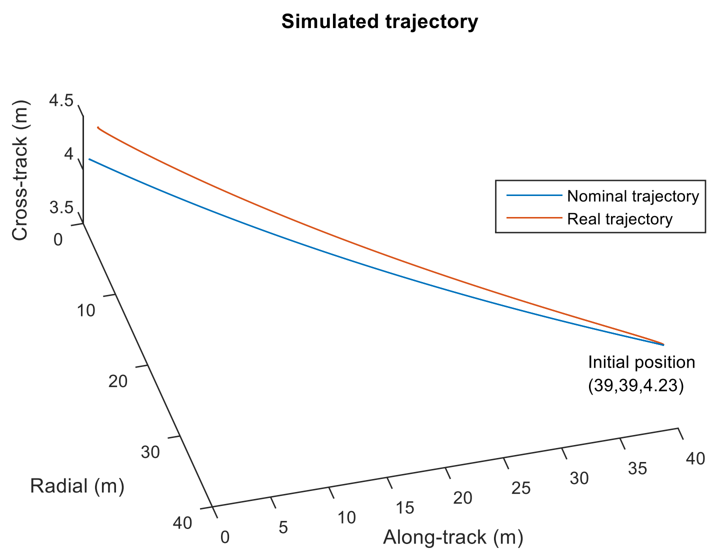


Figure 8-10 Simulated trajectories for steady state (1,1,4.446,0,0,0)

The nominal is given by the blue trajectory and the real trajectory by the red. Again, the form of the real trajectory, controlled by the full robust controller, maintains the form of the nominal trajectory. For ease of understanding, the trajectory is broken down in to one-dimensional components in the following figure.

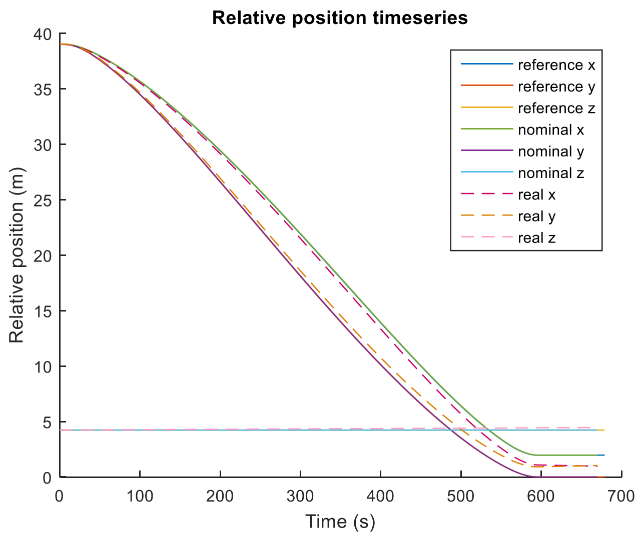


Figure 8-11 Relative position timeseries for steady state (1,1,4.446,0,0,0)

The reference and nominal trajectories appear as before and the real position components by the red, orange, and pink dashed series. Clearly, each real component follows the form of its corresponding reference to the new steady state.

Furthermore, the containment of the real trajectory within the tube can be shown.

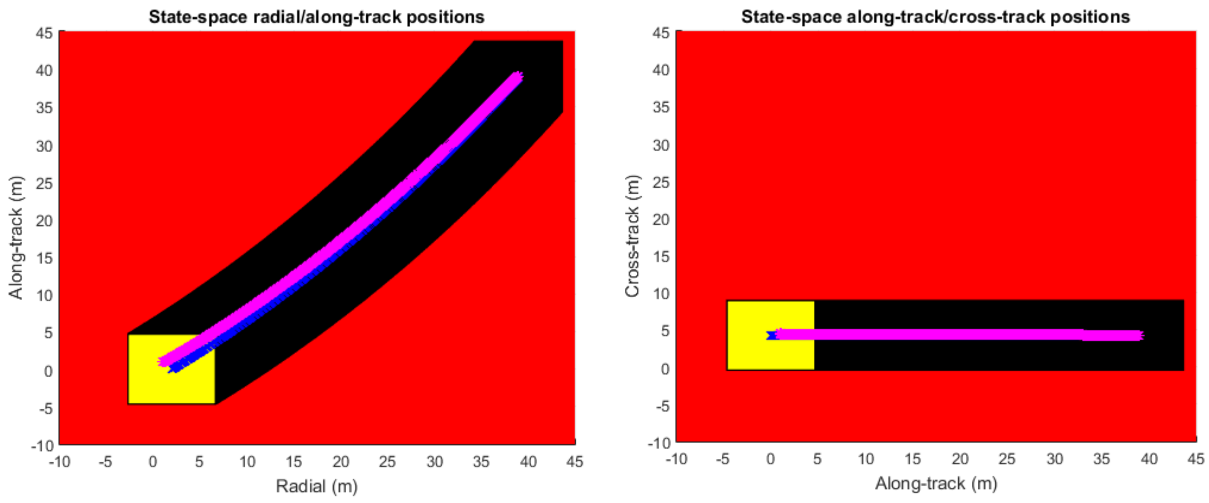


Figure 8-12 Location of the trajectories within the tube, (left) radial/along-track dynamics, (right) along-track/cross track dynamics, steady state (1,1,4.446,0,0,0)

In the above figure, red indicates the set of admissible steady states, the black prism with a yellow front face represents the tube of trajectories, the blue line represents the nominal trajectory, and the magenta the real trajectory. As before, the robust trajectory is contained well within the tube boundary indicating the trajectories whose evolution can be robustly controlled by the controller.

The evolution of the velocity states is given by the following.

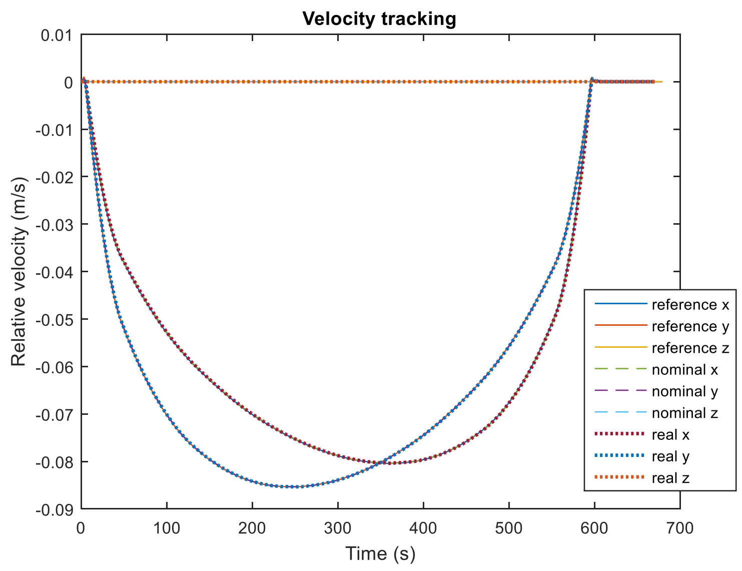


Figure 8-13 Velocity tracking for steady state (1,1,4.446,0,0,0)

Again, the real velocities are indicated by the dotted series. As expected, the real velocities track the form of the nominal. The difference between the nominal and robust velocities is similar to the preceding simulation, as it is a result of the robust system dynamics.

Finally, the robust control action is displayed below.

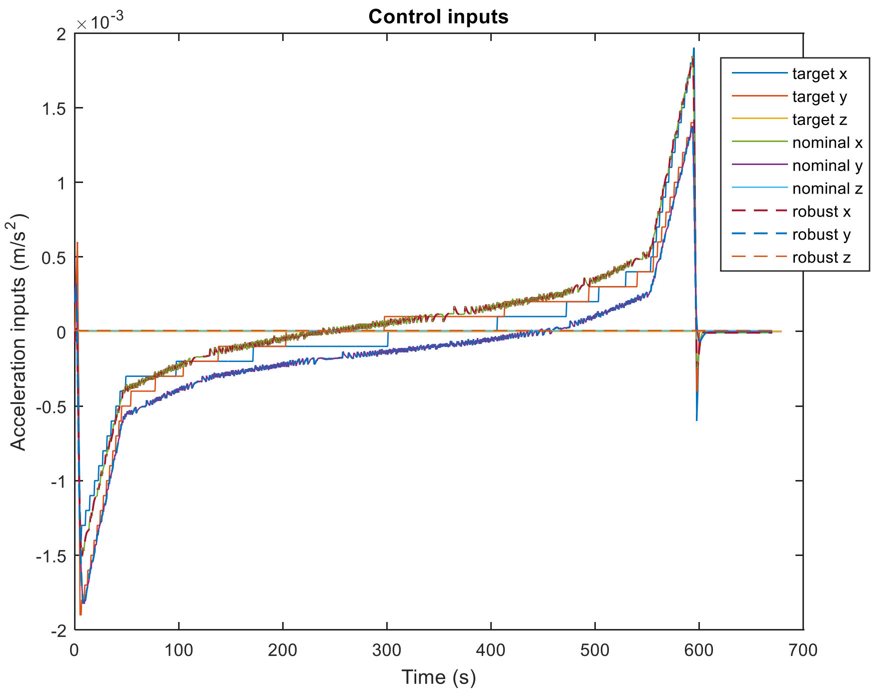


Figure 8-14 Control action for steady state (1,1,4.446,0,0,0)

The target and nominal inputs are identical to those in Figure 8-2, and the robust control actions are similar to those in Figure 8-8. As in Figure 8-8, the robust control inputs track the form of the targets. The additional acceleration given by the disturbance rejection term is given as follows:

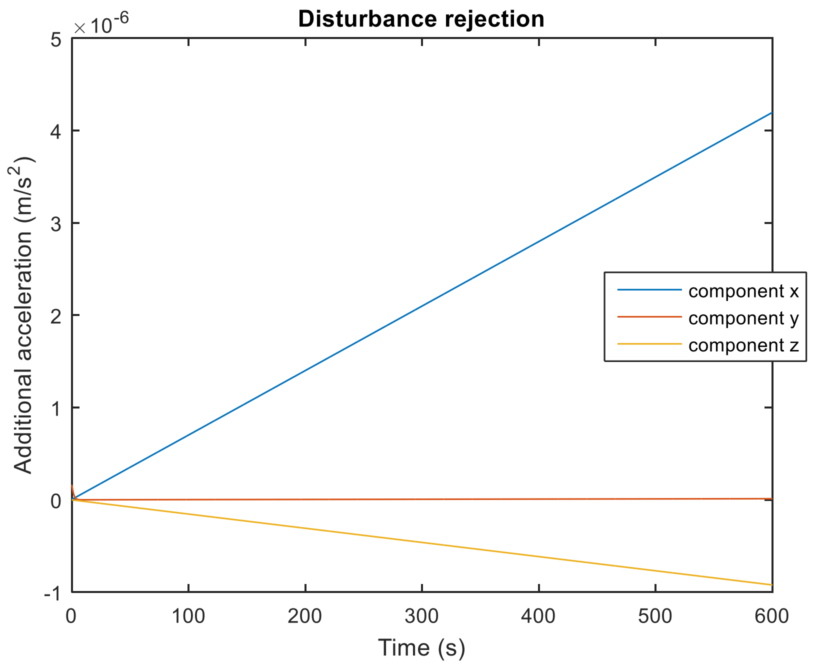


Figure 8-15 Additional acceleration from disturbance rejection term for steady state (1,1,4.446,0,0,0)

This is again of the same form as for the previous set point, but the growth in the radial and cross-track acceleration components are only about half of that in preceding case. This is likely due to the distance of the set point from the nominal set point being about half the size as in the first simulation. This would suggest that the greater the disturbance in the motion of the target, the greater the additional acceleration required from the robust controller. This is supportive evidence for a logical result.

The next simulation implements a disturbance is given by

$$\delta x = [-1.64{\times}10^{-3}\,t \quad -3.885{\times}10^{-3}\,t \quad -3.522{\times}10^{-3}\,t \quad 0 \quad 0 \quad 0] \tag{8-9}$$

and a corresponding steady state of

$$x(t_f) = [0 \quad -4.6662 \quad 0 \quad 0 \quad 0 \quad 0] \tag{8-10}$$

The final state is located at a greater distance from the nominal set point than the preceding two simulation steady states. As will be seen later, (8-10) is located at the boundary of the final iteration of $\mathcal{Z}$ in the tube. This simulation will demonstrate the consequences, if any, of the real steady state being located at the boundary of the robust control capability.

Again, it should be confirmed that the form of the nominal trajectory is followed by the robust trajectory. Consider, then, the nominal and real trajectories returned by the simulation.

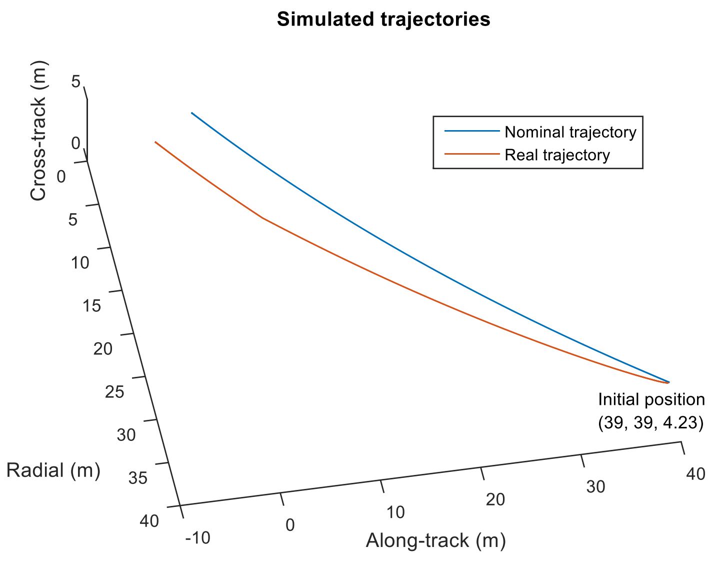


Figure 8-16 Simulated trajectories for steady state (0, -4.6662,0,0,0,0)

The nominal is given by the blue trajectory and the real trajectory by the red. Again, the form of the real trajectory, controlled by the full robust controller, maintains the form of the nominal trajectory. The trajectory is still contained within the tube, shown below, though the steady state lies on the boundary of the tube.

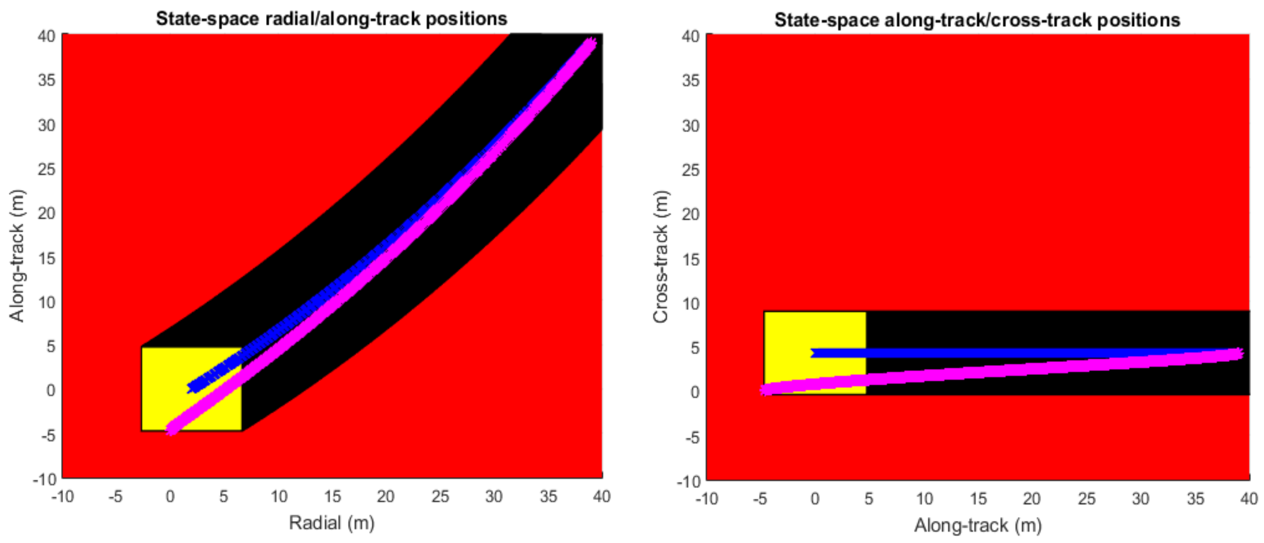


Figure 8-17 Location of the trajectories within the tube, (left) radial/along-track dynamics, (right) along-track/cross track dynamics, steady state (0,-4.6662,0,0,0,0)

In the above figure, the same color scheme is used as previously. This figure highlights the difference between this simulation and the preceding two: the set point is located on the boundary of the tube, albeit, this is the only point in the simulation when the boundary is encountered. This is an important property of the simulation, however. The implications become clear when the trajectory is broken down into one-dimensional components.

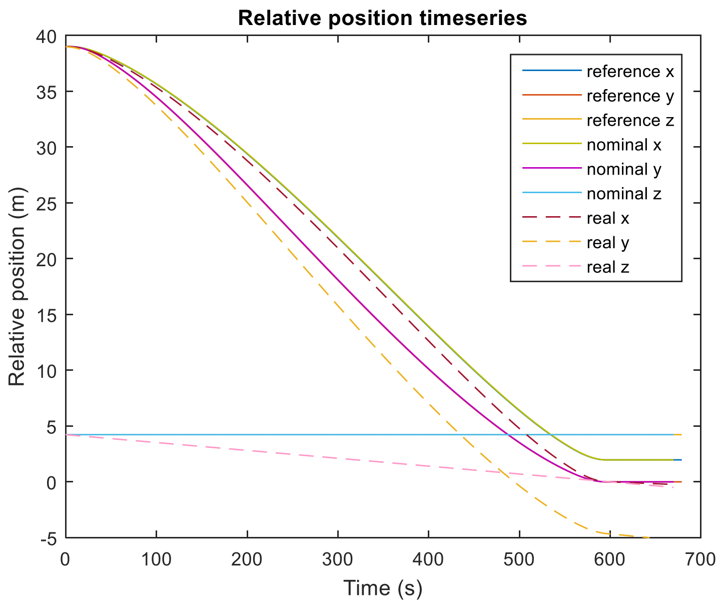


Figure 8-18 Relative position timeseries for steady state (0,-4.6662,0,0,0,0)

Each real component still follows the form of its corresponding reference to the new final state. However, the set point is barely reachable in the allotted simulation horizon. This can be seen by the real along-track and cross-track series struggling to remain steady after the 600 s mark.

The velocity states and acceleration inputs still track the forms of the reference and targets, respectively.

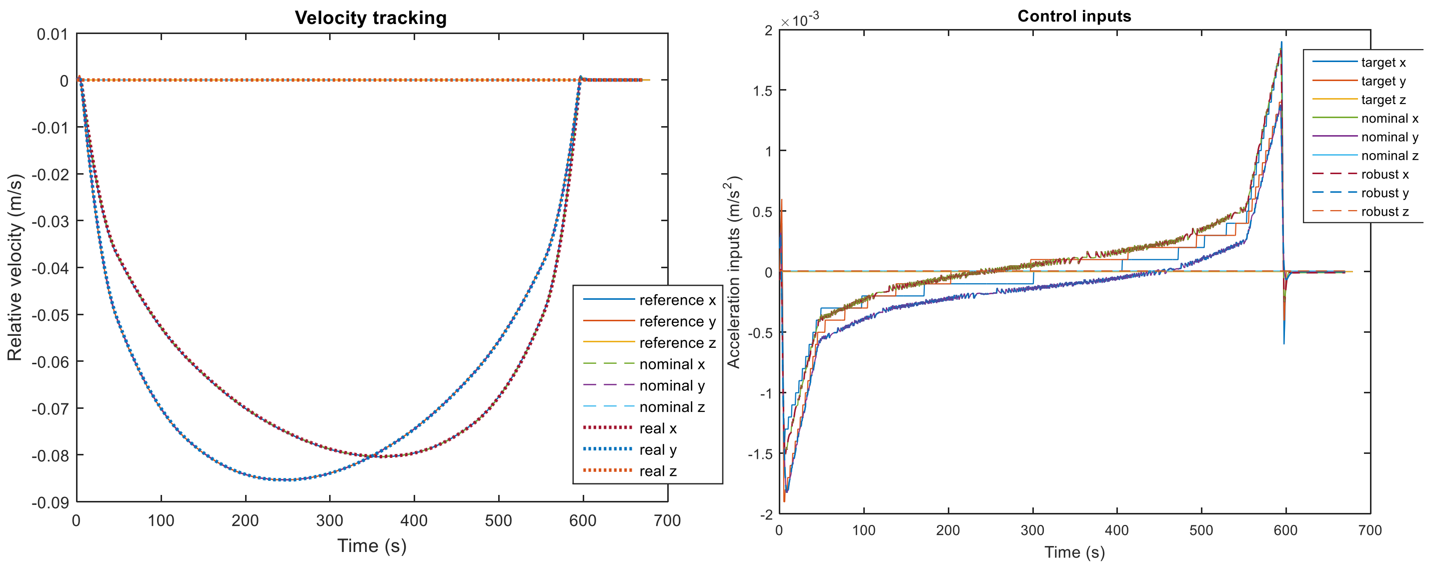


Figure 8-19 Velocity and input tracking for steady state (0,-4.6662,0,0,0,0)

This is desirable to maintain the obstacle avoidance and optimality properties. However, observe the disturbance rejection term profile for this simulation:

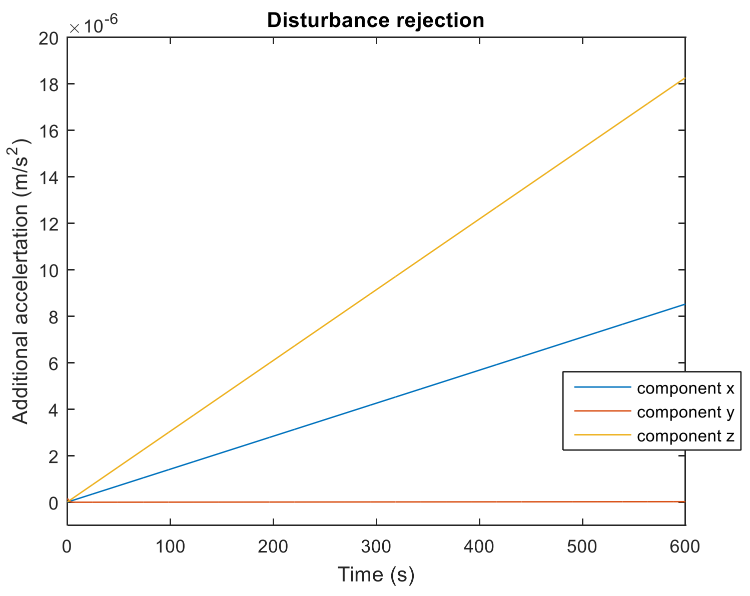


Figure 8-20 Additional acceleration from disturbance rejection term for steady state (0,-4.6662,0,0,0,0)

In comparison to the preceding two simulations, the additional acceleration added by the feedback term is more complex. Bear in mind that the distance of the set point from the nominal set point is much greater than either of the previous simulations. First, the along-track component is non-zero, unlike in the preceding two simulations, although by a very small amount as compared to the other two components. Now, note that the growth in the radial component is approximately the same as in the first scenario, despite the much greater disturbance. This is easily explained by breaking down the disturbance into components. In the preceding two simulations, the disturbances were mostly in the radial and cross-track position components, resulting in a large growth in the radial and cross-track disturbance rejection terms and negligible overall growth in the along-track disturbance rejection term. In this scenario, on the other hand, the disturbance is predominantly in the along- and cross-track position components. The decoupled nature of the cross-track component from the radial and along-track components and the quadratic weighting on the position in the determination of the cross-track acceleration accounts for the growth in the cross-track disturbance rejection term. The along-track and radial components are inter-related. The along-track component is defined by its linear coupling with the radial velocity only and, with no radial velocity disturbance and very small component introduced by the system dynamics, exhibits comparatively little growth. The radial acceleration component, on the other hand, applies a quadratic weighing on the radial position and a linear weighting on the along-track velocity. With the large weighting on the radial position, the disturbance in this component being the same as in the first scenario, the growth in the radial disturbance rejection term is comparable to that in the first simulation. The growth is larger in the radial component than the cross-track component due to the slightly greater weighting on the radial position.

Clearly, then, disturbances in the radial position will cause the greatest additional control effort, followed closely by disturbances in the cross-track. Disturbances in the along-track position of the set point will be of little consequence to the requisite robust control action. An addendum to this analysis can be made in the event that velocity uncertainties are incorporated in future work: it would be expected that disturbances in velocities will be of lesser consequence than position disturbances as they are of linear weighting in the system dynamics. Velocity disturbances in the cross-track will cause no additional control action to become necessary, in the along-track will require some additional control action on the radial component, and velocity disturbances in the radial will cause an overall additional control action to finally become necessary in the along-track acceleration.

This also explains why the controller is beginning to struggle to reach the set point under the simulation requirements. In fact, the tube is the region in which the nominal dynamics are guaranteed to be feasible. If the reference parameters are to be tracked, only the states within the tube, so within a bounded disturbance from the nominal, will be reachable while also tracking the reference. So, as the set point reaches the bound of the tube, the limit of the nominal dynamics is reached and the ability of the controller to drive the chaser to the set point as a steady state with soft docking and zero-velocity comes into question.

8.3.3 Testing the limits of the implemented controller

In these final simulations, some limitations of this controller will be demonstrated. Three cases are presented: The relationship between the nominal set point and these test points is illustrated below. In the first, a trajectory contained in the tube will fail to complete the rendezvous. In the second, the steady state will be admissible according to the problem constraints, but the trajectory will leave the tube. In the third case, the actuation constraints will be stressed by decimating the time allowed for the chaser to rendezvous with the target.

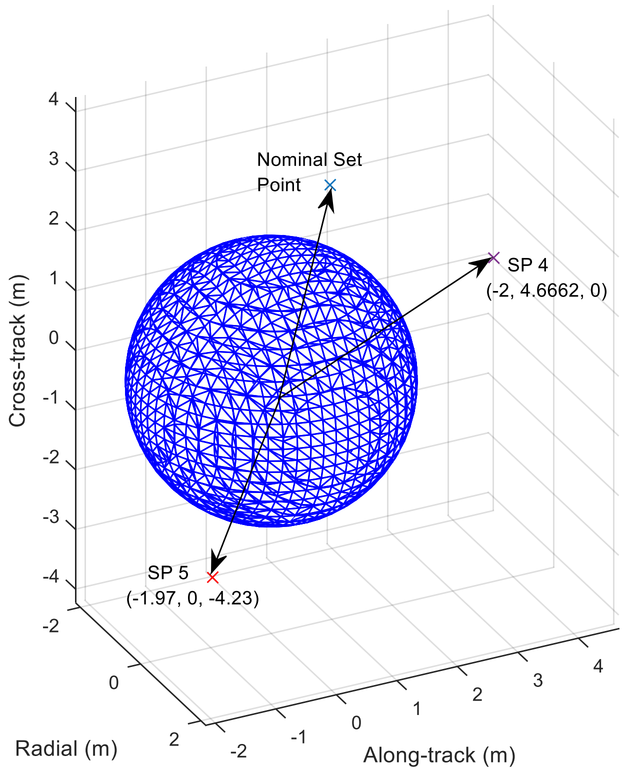


Figure 8-21 Set points used in limitations of the controller simulations

The first simulation implements a disturbance is given by

$$\delta x = [-3.05\times10^{-3}\,t \quad 3.885\times10^{-3}\,t \quad -3.522\times10^{-3}\,t \quad 0 \quad 0 \quad 0] \tag{8-11}$$

and a corresponding steady state of

$$x(t_f) = [-2 \quad 4.6662 \quad 0 \quad 0 \quad 0 \quad 0] \tag{8-12}$$

This will result in a trajectory which is contained within the tube, but approaches a point not located on the spherical shell described by the rotation of the grasping point. This is evidenced by the following figure.

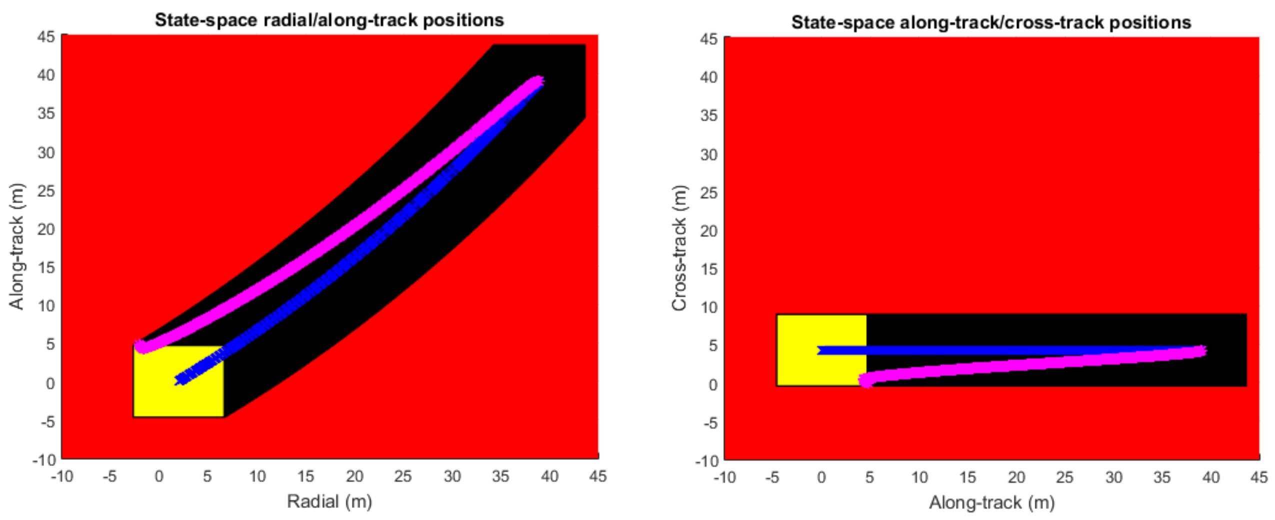


Figure 8-22 Trajectory contained within tube, failed rendezvous

The trajectory is clearly contained in the tube for the duration of the maneuver. However, by simple geometry, the steady state is not located in the desired region. As the point is located within the tube, the nominal dynamics will also still be valid, resulting in the nominal actuation to follow the form of the reference, as previously and shown in the following figure.

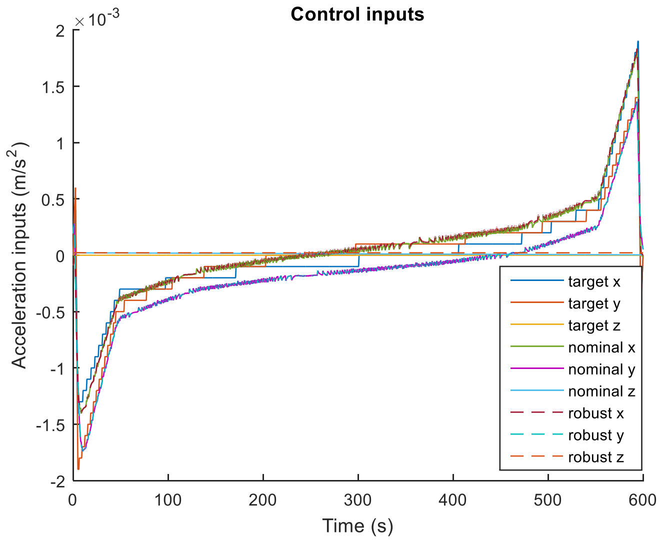


Figure 8-23 Actuation time series for a point in the tube, but not on the sphere

Attempting to maneuver to this point will therefore result in a false positive for successful rendezvous. This is because the controller views this point as an acceptable point to steer the chaser to. It would be fairly simple to maneuver the chaser from this point to the nearest acceptable steady state, but this would violate the simulation requirements.

In the second simulation, the disturbance is given by

$$\delta x = [-3.28\times10^{-3}\,t \quad 0 \quad -7.044\times10^{-3}\,t \quad 0 \quad 0 \quad 0] \tag{8-13}$$

and a corresponding final state of

$$x(t_f) = [-1.97 \quad 0 \quad -4.23 \quad 0 \quad 0 \quad 0] \tag{8-14}$$

This is a rather extreme case. Not only does the steady state correspond to a possible manifestation of the uncertainty in the satellite motion, but it is located on the far side of the target and clearly outside of the tube of trajectories. In this simulation, the state constraints imposed by the tube are stressed.

Without constraint (7-31)(a), the chaser would follow the following trajectory.

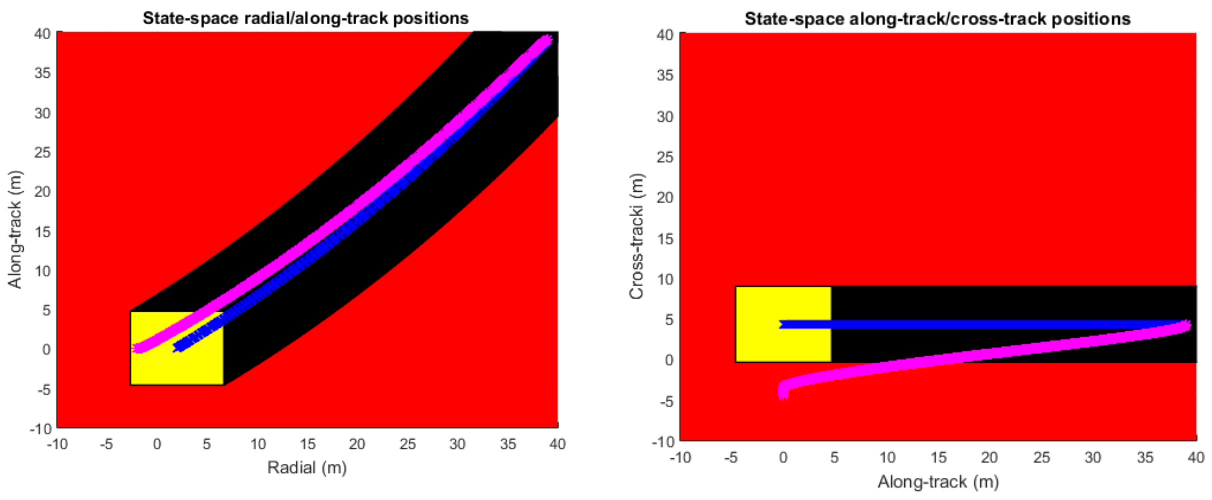


Figure 8-24 Trajectory departs from tube

The tube is departed from as a direct result of the state uncertainty bound being violated – the set point would require a terminal state disturbance of over 8 m.

Despite departing from the tube in the cross-track, it is clear that the real trajectory does follow the form of the nominal.

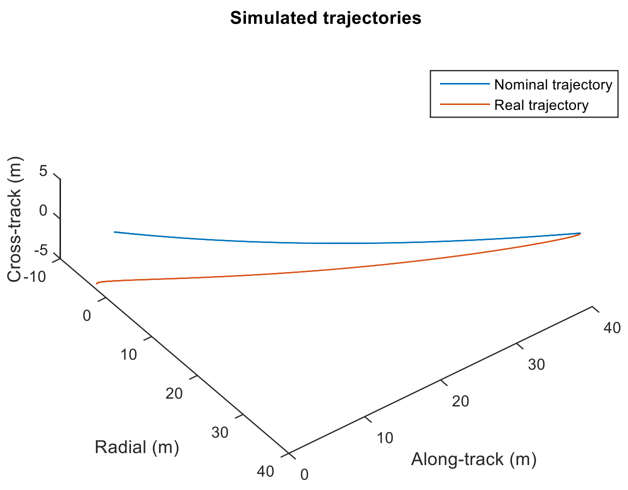


Figure 8-25 Simulated trajectories for steady state (-1.97,0,-4.23,0,0,0)

The actuation required for this maneuver would have the following form.

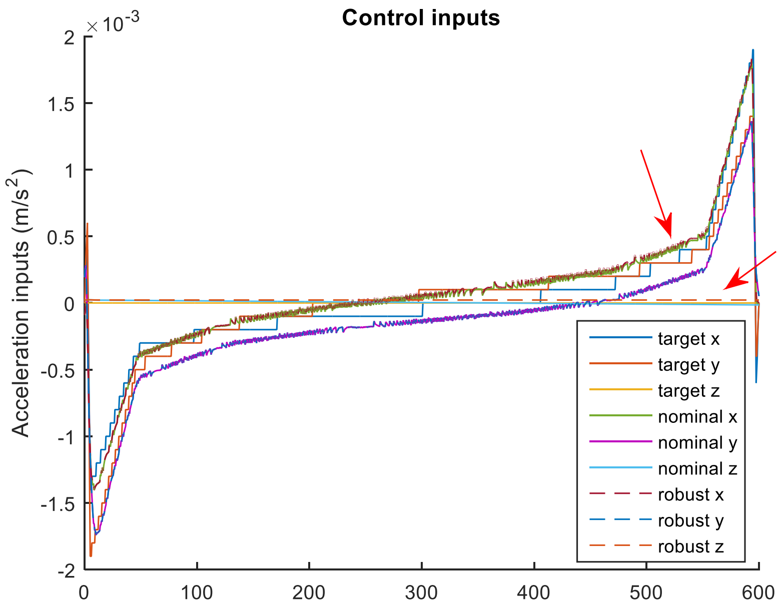


Figure 8-26 Control inputs without tube constraint

The actuation certainly follows the form of the nominal actuation, but the additional acceleration applied by the disturbance rejection term is now large enough to cause a deviation from the nominal series to become visible. Some points where this is best visible in the radial and cross-track series are highlighted in the figure by the arrows.

The state tracking requirements and all constraints other than that the trajectory remain in the tube (which was omitted in the results of this simulation so far) are still adhered to. However, by tracking the reference as required the chaser would have to pass through the sphere over bounding the grasping point. So, while the point could be reached despite departing from the tube, the reference conditions mean that a collision would likely occur between the target and chaser spacecraft, which is undesirable for obvious reasons.

However, the controller has a constraint which requires the trajectory of the chaser to remain in the tube. The application of this constraint results in the following chaser trajectory.

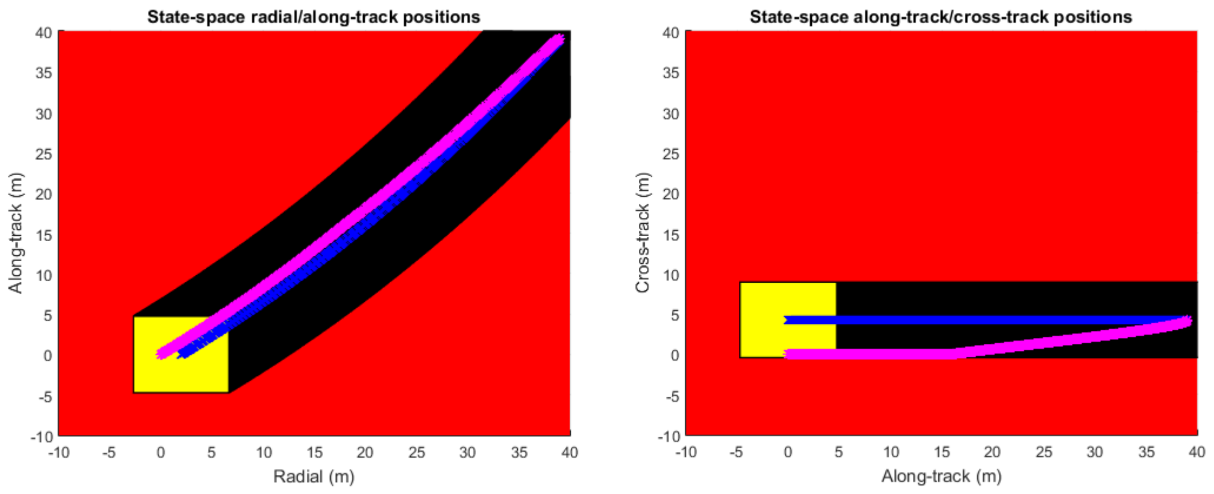


Figure 8-27 Trajectory remains in tube

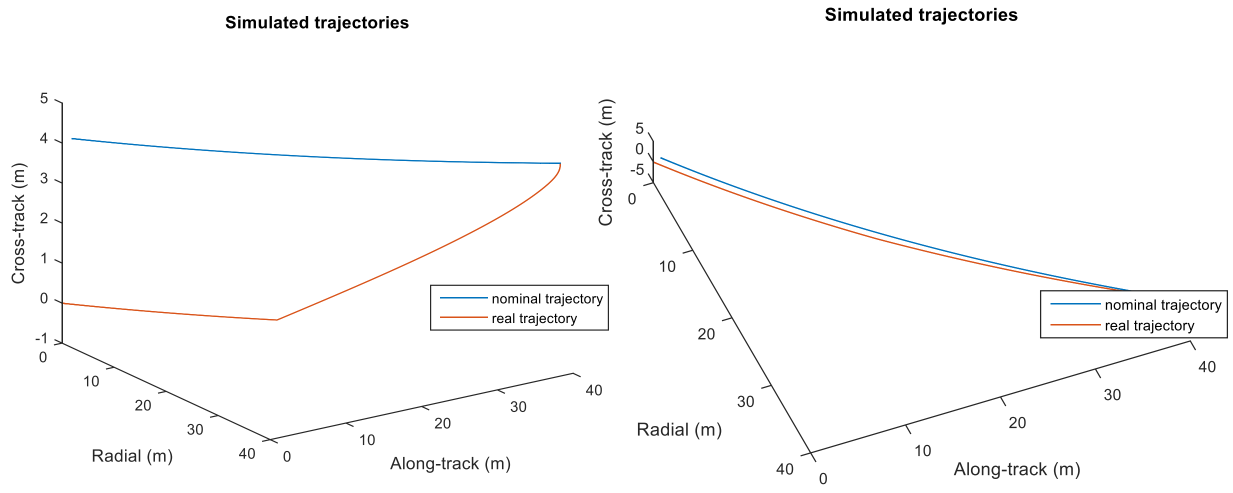


Figure 8-28 Trajectory for set point (-1.97,0,-4.23,0,0,0)

Now the trajectory remains clearly inside of the tube, despite attempting to maneuver to the required set point. The state constraint imposed by the tube, despite being stressed for most of the simulation, holds and the chaser is driven to the closest point on the sphere to the desired final state. The series of control inputs for this action is given by the following figure.

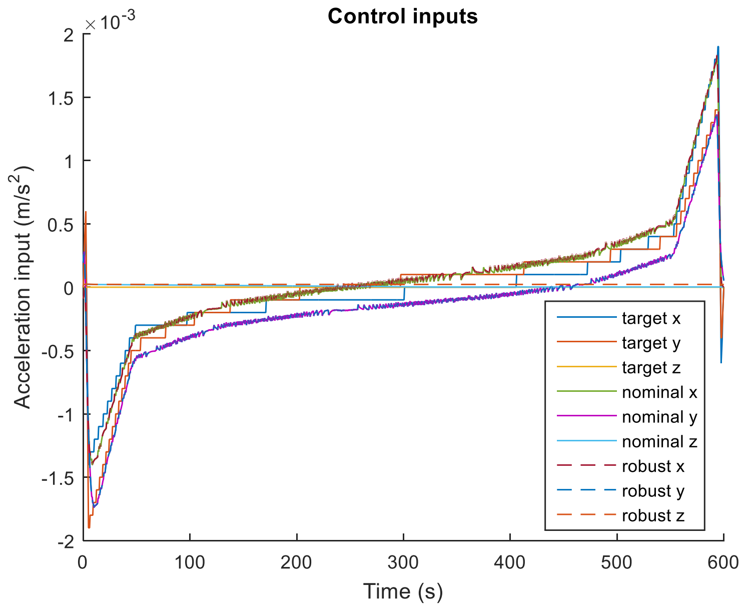


Figure 8-29 Control action with the tube constraint

The chaser therefore maneuvers to the steady state within the tube which is closest to it desired destination. This effect is independent of the control method's direction of trajectories to the nearest point in the admissible steady state set, which is indemonstrable in the confines of this problem. As can be seen in Figure 8-28, the controller will follow the form of the reference trajectory as best as it can. However, when the disturbed trajectory reaches the boundary of the tube, the harder constraint wins, and the trajectory is forced to remain in the tube. Due to the nature of the cost function, the controller still attempts to follow the form of the reference. In many aspects it does this job remarkably well.

In each of these simulations, the rendezvous maneuver expectedly fails. In the first scenario, the chaser is maneuvered into the neighbourhood of the target, but the mouth of the tube includes points which are valid in terms of the system constraints, but not valid set points in terms of the rendezvous goal. It is unlikely that the chaser would be required to be maneuvered to this point, but should the event occur, an additional correction maneuver or the docking maneuver would be required to account for this discrepancy. In the second scenario, the steady state is actually not reachable under the operation conditions – the uncertainty bound has been violated.

The set of admissible steady states are valid for the general controller, but when reference tracking is employed in the manner described here with the permissible allotted disturbance, the nominal trajectory is locked and the only truly admissible states are located in the intersection of the tube and the spherical shell on which the grasping point may lie. Points located on the shell but outside of the intersection will require deviation from the form of the nominal or the permission of disturbances applied to the relative velocity states.

The design of the controller means that each point within the state constraints are valid initial states and nominal set points. The set of desirable steady states, located on the spherical shell described by the uncertain motion of the grasping point, is well within the set of admissible steady states, along with most of points contained by the state constraint. Each of these desired steady states is theoretically reachable under nominal dynamics in the presence of a state disturbance less than or equal to the disturbance bound. The reference trajectory locks a set of initial and nominal terminal conditions for the maneuver and the form of the path and series of control actions. The tube then defines the subset of trajectories which can be driven to the disturbed set points whilst adhering to the nominal tracking requirements.

Finally, some simulations were conducted in which the permissible satellite rendezvous time was much shortened. The same basic simulator was used and a final state of $x(t_f)$ =(0,-4.6662,0,0,0,0) was used. The code for these simulations is also included in part C of the appendix. In these simulations, the rendezvous time was progressively halved to observe the effect on the required actuation. While this situation is not particularly realistic, it is contrived here for the purposes of demonstrating the behaviour of the controlled system. The nominal actuation series and disturbance rejection terms of these simulations are presented below in Figure 8-30 on the following page.

The sequence of images on the left hand side of the figure demonstrate the difference between the reference and nominal actuations as the simulation time is shortened. In the first halving of the simulation time, the nominal actuation approximately doubles in magnitude but the form of the reference series is largely retained. This is largely true for the second halving as well, however the nominal actuation does not reach back to zero at the end of the simulation. At the third halving of the rendezvous time, the actuation constraints are met. The maximum acceleration is attained and the thrusters saturate. Nonetheless, the form of the actuation series still vaguely resembles the reference. Now, in the 37.5 s rendezvous simulation, a complete breakdown in the reference tracking occurs. In all of the preceding rendezvous durations, the chaser is able to track the position reference and meet the target. However, in this last scenario, the chaser will not even reach the vicinity.

While the nominal actuation experiences great change in the truncation of the simulation time, the disturbance rejection term remains largely similar. This is not to be wholly unexpected in this implementation. It can be noted that the reference actuations and the nominal actuations while following the reference trajectory for the rendezvous duration that it was designed for are very small – to the order of 10^{-3} at their maxima, bur 10^{-4} for most of the maneuver, while the nominal constraint is to the order of 10^{-2}. There is clearly just a lot of head room for the nominal control to utilize before the robust control term must take on any meaningful additional effort.

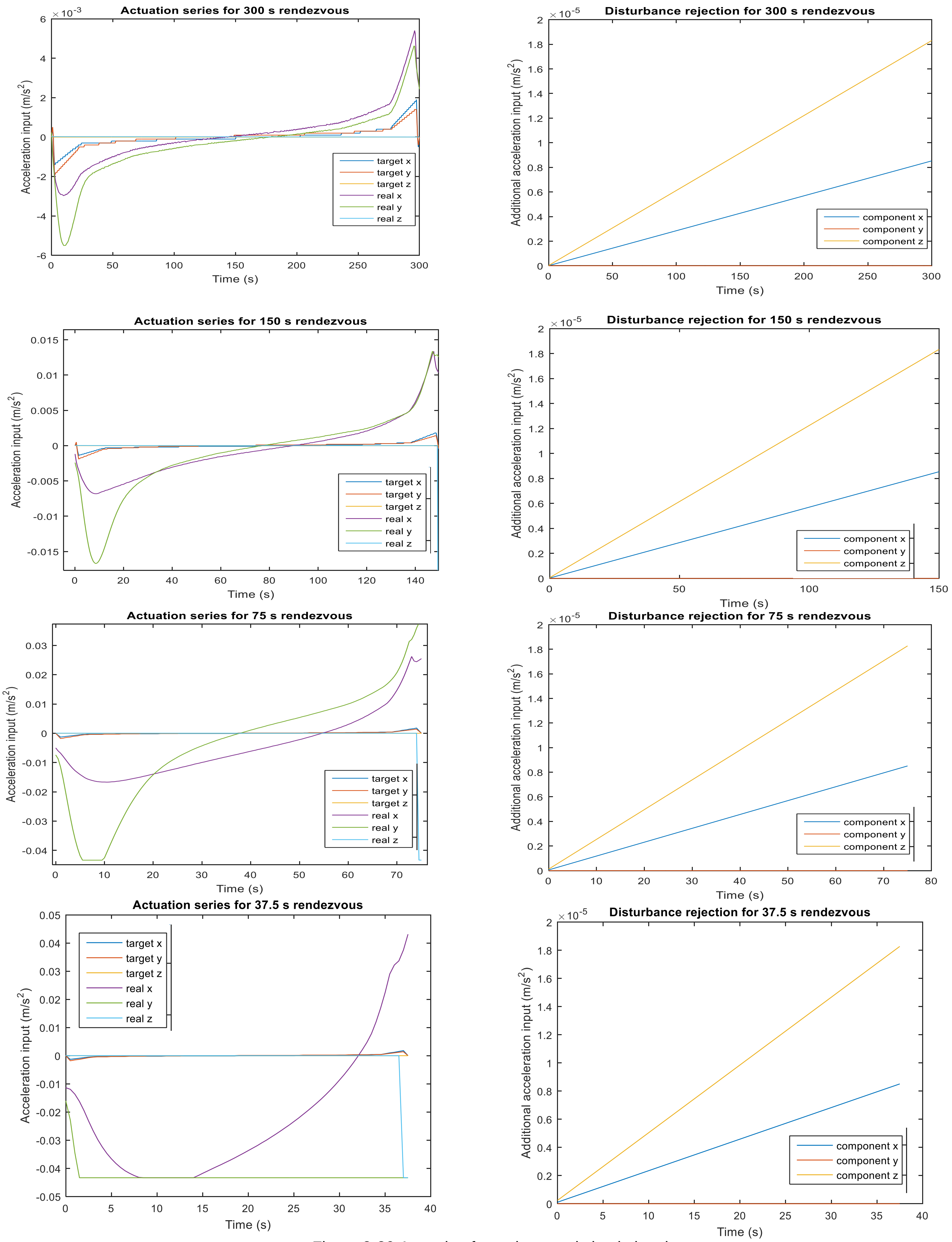


Figure 8-30 Actuation for a shortened simulation time

8.3.4 On characterizing the limits of the controller

A predictable subset of the points contained within the end of the tube will be located on the sphere drawn by the uncertainty of the grasping point. This can simply be determined as follows.

Algorithm 8-1 Set points for successful rendezvous

```
Input:        mRPI mapped to the nominal set point
              Radius $r$ of sphere drawn by the grasping point

Algorithm:    for each point $\boldsymbol{p} = [x, y, z]$ in the mapped mRPI set

                  If $|\boldsymbol{p}| = \boldsymbol{r}$
                  $\boldsymbol{p}$ is a robust set point
                  else
                          $\boldsymbol{p}$ is not a robust set point
                  end_conditional

              end_for

Output:       set of desirable set points which can be robustly reached for the given nominal trajectory
```

For the nominal set point used in these simulations, the intersection is as in the following figure

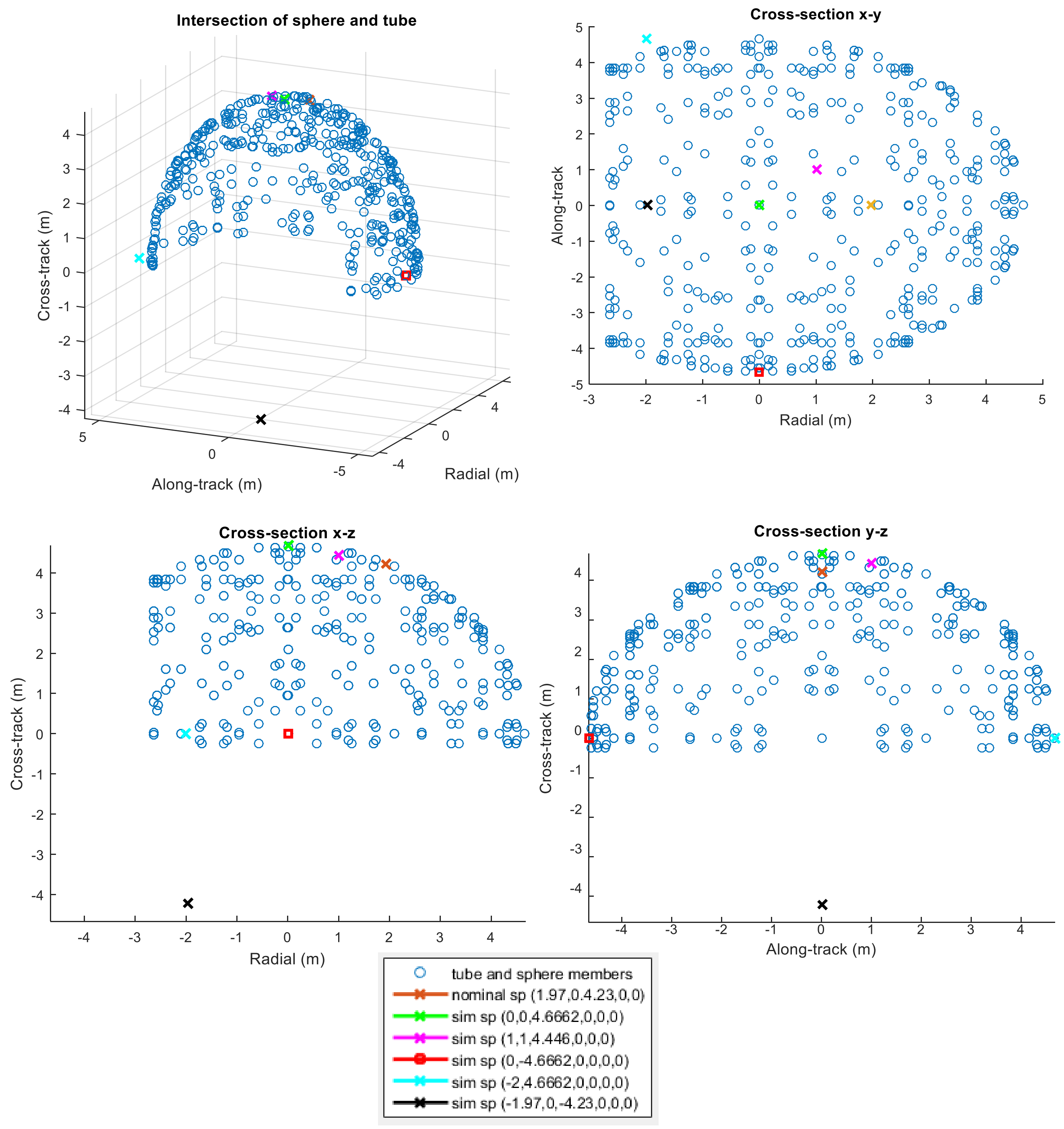


Figure 8-31 Intersection of tube with sphere drawn by the uncertainty of target motion for the nominal set point (1.97,0,4.23,0,0,0)

In the figure, the portion of the sphere contained in the tube is indicated by the scatter of points determined by the algorithm. The points on the surface of the sphere were determined sparsely to allow the simulation set points to be visible. The nominal and simulation set points are indicated in relation to this portion, differentiated from each other as described in the legend. As can be seen, nearly the whole of the upper hemisphere is contained within the tube at this nominal set point. Given the approximately symmetric nature of the mRPI set, it can be reasoned that approximately the same surface area of the sphere will be available at all nominal set points. This suggest that at any nominal set point, the chaser can be robustly maneuvered to nearly half of the possible points at which the grasping point could be while tracking the nominal response.

8.3.5 Validation

In an effort to validate these results, an inverse dynamics test and an analytical solution for the CWH equations test were applied to the obtained simulation data. The results of these tests are presented in this section. The code used to conduct these tests is given in Appendix part D. Validation.

In the inverse dynamics test, the simulated nominal and real trajectories were passed to a re-arranged system to determine the requisite acceleration component magnitudes to acquire these trajectories

$$u = (B^T B)^{-1} B^T (x^+ - Ax) \tag{8-15}$$

In the case of the real trajectory, the disturbance w is incorporated into the states, and so should not be additionally addressed in the inverse system. The series of actuation inputs u can then be compared to the reference and/or the series obtained from the simulator and the requisite disturbance rejection terms can be observed and compared to those obtained in simulation. The actuation series and disturbance rejections yielded by the application of (8-15) are presented in Figure 8-32, for comparison to Figure 8-8, Figure 8-9, Figure 8-14, Figure 8-15, Figure 8-19, Figure 8-20.

It is evident from the input series that the simulator and inverse dynamics that yield the same actuation sequences. By simple comparison to the preceding figures, the disturbance rejection is similarly equivalent. As should be clear, then, that the series of input and disturbance rejection terms obtained in simulation are plausible for the obtained trajectories.

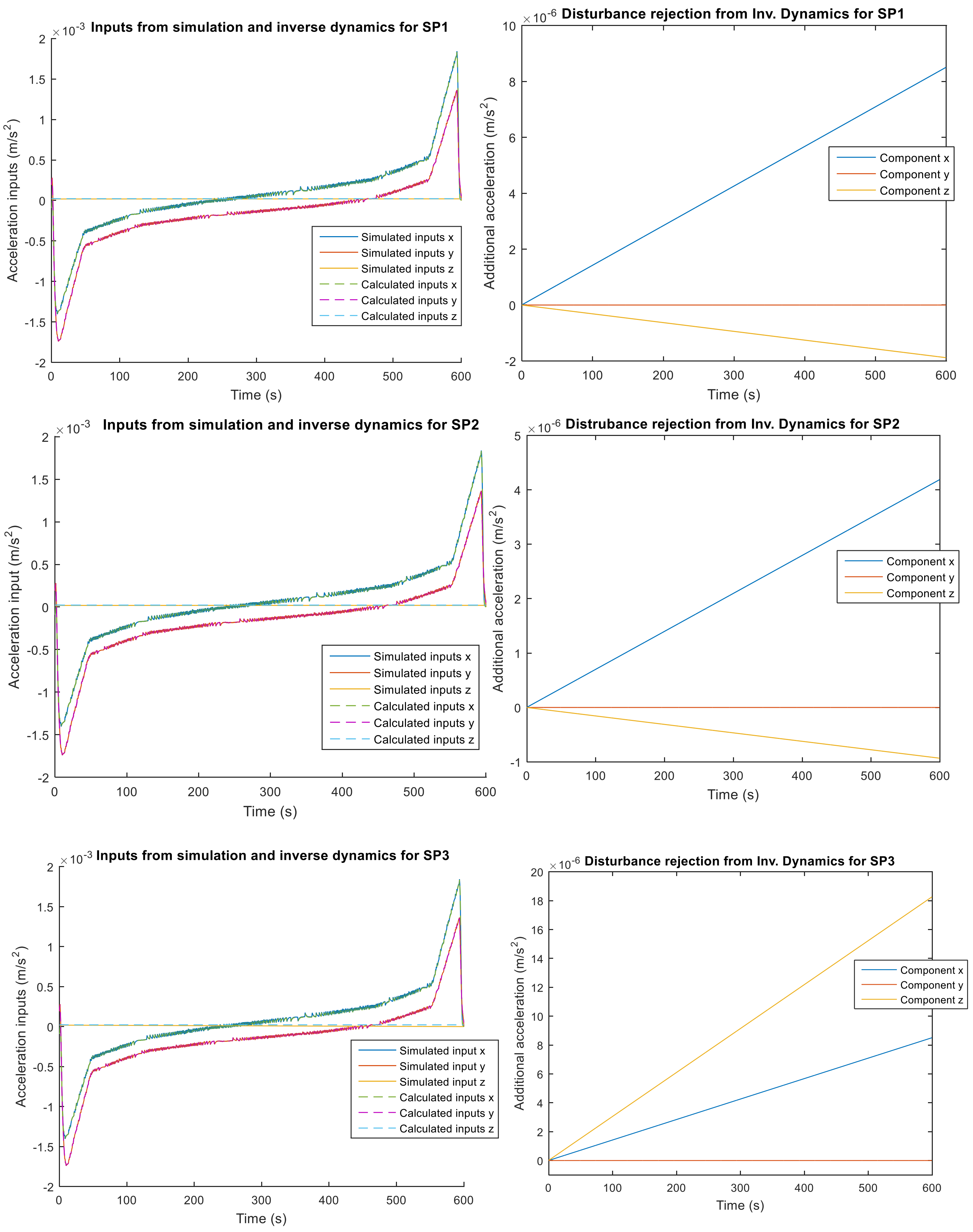


Figure 8-32 Actuation and disturbance rejection series obtained from inverse dynamics

The analytical solution for the CWH equations is included to show that the obtained trajectories are valid for CWH dynamics. The series of positions obtained in simulation are used to determine the initial conditions for the analytical solution, which is then solved. The results are presented here.

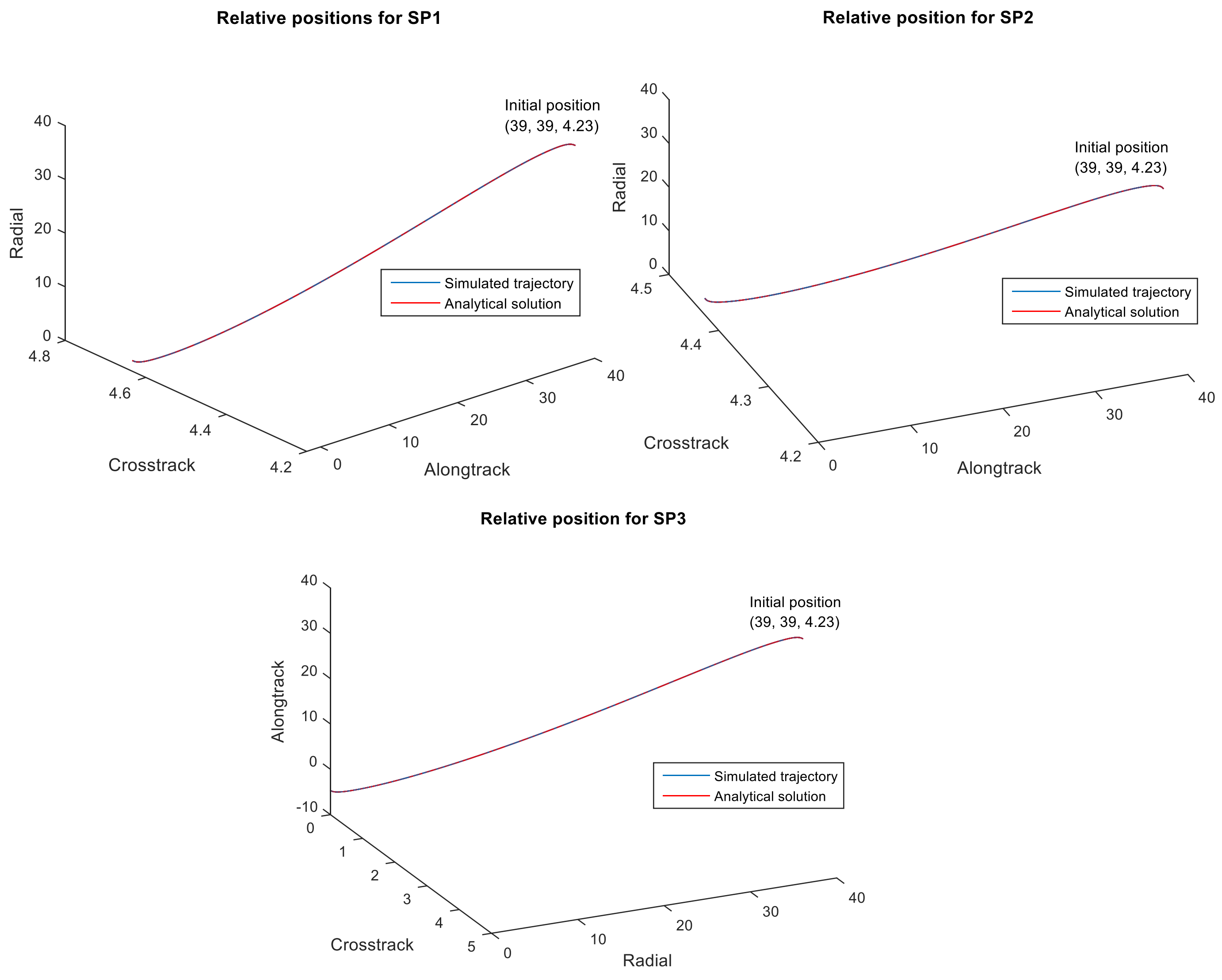


Figure 8-33 Analytical solution trajectories

For each simulation scenario, the trajectory obtained through the analytical solution tracks the simulated trajectory. Again, the test clearly supports the validity of the results obtained by the simulator.

8.3.6 Timing statistics

Tube-based robust model predictive control is computationally intensive in the design of the controller. However, these calculations must be conducted only once. The implementation of the controller is similar in intensity to nominal model predictive control. The timing aspects are therefore also similar, making the controller suitable to online operation. Table 8-1 enumerates the timing statistics for the presented simulations to demonstrate this capability.

Table 8-1 Timing metrics for 600 s reference trajectory, from 100 simulations

Simulator	Max time (s)	Min time (s)	Avg time (s)
Nominal controller, no disturbance rejection term	64.5776	40.5262	46.3674
Tube-based robust nominal controller	63.3588	40.4447	46.2618
Tube-based robust MPC, perturbed systems	59.8543	40.5003	46.3782

The additional simulator used to obtain the timing metric for the nominal controller with no disturbance rejection term is also given Appendix part C. Satellite Rendezvous simulator.

9 Conclusion

The goal of this work was to robustly conduct a rendezvous maneuver while tracking a provided optimal trajectory. This was conducted using Tube-based robust Model Predictive Control. In Part 1, the relevant theory for rendezvous and proximity maneuvers and for nominal and tube-based robust model predictive control was detailed; the design methodology was conducted and implemented in software for a sample problem; and the controller designed for the sample problem was simulated. In Part 2, the participants and maneuver requirements were defined and the tube-based model predictive controller for the task was developed and simulated.

To the best of the author's knowledge, there exist only two publications, [37] and [38], on the use of tube-based robust model predictive control to control rendezvous and proximity operations maneuvers. In these two works, the satellites are taken to be travelling on elliptical orbits, the uncertainty was derived from navigation and thruster timing errors, respectively, and tracking was omitted. In this work, a circular orbit was assumed and the uncertainty was introduced to the dynamics by the target rather than the chaser.

In tube-based robust MPC for tracking, trajectories contained within a tube centered on the nominal response of the system are guaranteed to evolve robustly to a neighbourhood, defined by the mRPI, of an admissible final state. In this implementation of Tube-based robust MPC, the nominal dynamics are constrained to track a provided optimal reference trajectory. Simulations of the controller show that the nominal response is robustly tracked and a set point offset from the nominal final state by a disturbance in the motion of the target will be successfully reached while this real set point remains within the tube. It has been shown that not all points on the sphere of uncertainty described by the possible target motion will be contained within the progression of the tube as a direct result of the requirement that the reference trajectory be tracked. The intersection of the tube and the sphere represents the set of possible set points which can be reached with guaranteed reference trajectory tracking with robust adherence to system requirements.

The controller has additionally been shown to be appropriate for online control applications. One of the motivating qualities of tube-based robust model predictive control is the similar complexity and processing times to nominal model predictive controllers. Model predictive controllers are used in online, real-time applications. As indicated in Table 8-1, the standard MPC and the robust MPC implementations have similar time requirements, supporting the claims of online, real-time applicability of the method.

There are several aspects which should be considered in future work:

For continued operation in the orbital frame, some additional constraints for the avoidance of collisions with obstacles or the target satellite should be implemented. This would require the re-consideration of the terminal set $\mathbb{X}_f$ such that regions of known danger are not permitted.

The trajectory devised by the motion planner is constructed with obstacle avoidance in mind. These obstacles may be mobile relative to the target in the target orbital frame, in which this work was conducted. In the target body frame, the obstacles are static with respect to the target. By tracking the reference trajectory referred to the target body frame, the optimal obstacle avoidance characteristics of the trajectory will remain intact. Retaining these properties while maneuvering to the true position of the docking point is desirable for obvious reasons. Therefore, tracking the reference trajectory referred to the target body frame should be conducted in the future.

In this implementation, the actuation constraints were a non-issue, as it is not particularly easy to force the system to reach the nominal actuation constraint boundary. This is not likely the case when the problem is expressed in the body frame of the target. Therefore, referral of the maneuver and control action to the body frame will also allow more strenuous use of the actuation constraints.

It should be noted that the designed controller not only functions as the theory suggests for the defined disturbance bound and constraints, it is also valid with minor adjustment for rendezvous with other target spacecraft. The success of the controller could therefore be further characterized in future work through the conduction of more simulations with the characteristics of other targets. Future work should also investigate and characterize the causes and effects of relative velocity uncertainty.

Further work could also explore the implications of non-linearity in the state equation, which would require the abandonment of the linear tube-based robust model predictive control theory in favour of the non-linear theory. There are various ways in which non-linearity can be introduced. The most obvious would to consider that in a multi-body system, the equations of motion involve non-linear and non-inertial terms. The Newtonian relation of force and acceleration is therefore not applicable and the linear dependence of the state equation on the input actuation becomes non-linear. Additional non-linear elements exist in the production of the acceleration dictated by the controller as between the determination and output of the optimized acceleration from the controller and a force being applied by the thrusters, a number of steps occur, which may likewise introduce a non-linear relationship to the actuation in the state equation.

10 References

[1] I. Alvarado, "Model predictive control for tracking constrained linear systems," Doctor of Philosophy, Department of Systems Engineering and Automation, University of Sevilla Sevilla, 2007.

[2] M. Balandat, "Constrained robust optimal trajectory tracking: model predictive control approaches," Dipl.-Ing. Diplomarbeit, Fachbereich Maschinenbau Institut für Flugsysteme und Regelungstechnik, Technische Universität Darmstadt, Darmstadt, Germany, 2010.

[3] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking of piece-wise constant references for constrained linear systems," in *16th IFAC World Congress*, Prague, Czech Republic, 2005.

[4] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking piecewise constant references for constrained linear systems," *Automatica,* vol. 44, pp. 2382-2387, 2008.

[5] CVX Research Inc., "CVX: Matlab software for disciplined convex programming, version 2.0," ed, April 2011.

[6] M. Grant and S. Boyd, "Graph implementation for nonsmooth convex programs, Recent advances in Learning and control," in *Lecture Notes in Control and Information Sciences*, V. Blondel, S. Boyd, and H. Kimura, Eds., ed: Springer-Verlag Limited, 2008, pp. 95-110.

[7] M. Herceg, C. N. Kvasnica, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zurich, Switzerland, 17-19 July 2013, pp. 502-510.

[8] J. Löfberg, "YALMIP: A Toolbox for Modeling and Optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[9] A. Bemporad, M. Morari, and N. Ricker. (2015). *Model Preditive Control Toolbox User's Guide (R2015b)*.

[10] The MathWorks Inc. *Model Predictive Control Toolbox Documentation (R2015b)*. Available: https://www.mathworks.com/help/mpc/

[11] Model Predictive Control Toolbox (r2015b), The MathWorks, Inc., Natick, Massachusetts, United States.

[12] J. L. Goodman, "History of space shuttle rendezvous and proximity operations," *Journal of Spacecraft and Rockets,* vol. 43, pp. 944-959, 2006.

[13] E. F. Camacho and C. Bordons, *Model Predictive Control*, 2 ed. London: Springer-Verlag, 2007.

[14] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, WI, USA: Nob Hill 2009.

[15] F. Gavilan, R. Vazques, and E. Camacho, "Robust model predictive control for spacecraft rendezvous with online prediction of disturbance bounds," 2011.

[16] L. Breger and J. P. How, "Safe trajectories for autonomous rendezvous of spacecraft," *Journal of Guidance, Control, and Dynamics,* vol. 31, 2008 2008.

[17] A. Richards, L. Breger, and J. P. How, "Analytical performance prediction for robust constrained model predictive control," *International Journal of Control,* vol. 79, pp. 877-894, 20 Feb 2007 2007.

[18] A. Richards and J. P. How, "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility," in *American Control Conference*, 2003, 2003, pp. 4034-4040.

[19] E. Hartley, "Model predictive control for spacecraft rendezvous," Ph.D., University of Cambridge, 2010.

[20] S. Di Cairano, H. Park, and I. Kolmanovsky, "Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering," *International Journal of Robust and Nonlinear Control,* vol. 22, pp. 1398-1427, 2012.

[21] H. Park, S. Di Cairano, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking with a rotating/tumbling platform and or debris avoidance," in *2011 American Control Conference*, San Francisco, CA, USA, 2011, pp. 1922-1927.

[22] A. Weiss, I. Kolmanovsky, M. Baldwin, and R. S. Erwin, "Model predictive control of three dimensional spacecraft relative motion," presented at the 2012 American Control Conference, Montreal, Canada, 2012.

[23] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking: strategies for handling constraints and case studies," *IEEE Transactions on Control Systems Technology,* vol. 23, pp. 1638-1647, July 2015 2015.

[24] A. Weiss and S. Di Cairano, "Robust dual control MPC with guaranteed constraint satisfaction," in *2014 IEEE 53rd Annual Conference on Decision and Control*, Los Angeles, CA, USA, 2014, pp. 6713-6718.

[25] W. Langson, I. Chryssochoos, S. Raković, and D. Mayne, "Robust model predictive control using tubes," *Automatica,* vol. 40, pp. 125-133, 2004.

[26] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica,* vol. 41, pp. 219-224, 2005.

[27] D. Q. Mayne, S. V. Raković, R. Findeisen, and F. Allgöwer, "Robust output feedback model predictive control of constrained linear systems," *Automatica,* vol. 42, pp. 1217-1222, 2006.

[28] D. Q. Mayne, S. V. Raković, R. Findeisen, and F. Allgöwer, "Robust output feedback model predictive control of constrained linear systems: Time varying case," *Automatica,* vol. 45, pp. 2082-2087, 2009.

[29] D. Q. Mayne, S. V. Raković, R. Findeisen, and F. Allgöwer, "Robust output feedback model predictive control of constrained linear systems," *Automatica,* vol. 42, pp. 1217-1222, 2006.

[30] S. Raković and D. Mayne, "A simple Tube Cotroller for efficient Robust Model Predictive Control of constrained linear discrete time systems subject to bounded disturbances," in *16th IFAC World Congress*, 2005, pp. 241-246.

[31] I. Alvarado, D. Limon, T. Alamo, and E. Camacho, "Output feedback Robust tube based MPC for tracking of piece-wise constant references," presented at the 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 2007.

[32] I. Alvarado, D. Limon, T. Alamo, M. Fiacchini, and E. Camacho, "Robust tube based MPC for tracking of piece-wise constant references," presented at the 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 2007.

[33] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "On the design of Robust tube-based MPC for tracking," in *17th IFAC Triennial Congress*, Seoul, 2008.

[34] D. Mayne, E. Kerrigan, E. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control,* vol. 21, pp. 1341-1353, 17 April 2011 2011.

[35] D. Q. Mayne and E. C. Kerrigan, "Tube-based robust nonlinear model predictive control," in *7th IFAC Symposium on Nonlinear Control Systems*, Pretoria, South Africa, 2007, pp. 36-41.

[36] D. Q. Mayne, E. C. Kerrigan, and P. Falugi, "Robust model predictive control: advantages and disadvantages of tube-based methods," in *Proceedings of the 18th IFAC World Congress*, Milano, Italy, 2011, pp. 191-196.

[37] G. Deaconu, C. Louembet, and A. Théron, "Minimizing the effects of navigation uncertainties on the spacecraft rendezvous precision," *Journal of Guidance, Control, and Dynamics,* vol. 37, pp. 695-700, 2014.

[38] C. Louembet, D. DArzelier, and G. Deaconu, "Robust rendezvous planning under maneuver execution errors," *Journal of Guidance, Control, and Dynamics,* vol. 38, pp. 76-93, Jan 2015 2015.

[39] T. E. Carter, "State transition matrices for terminal rendezvous studies: brief survey and new example," *Journal of Guidance, Control, and Dynamics,* vol. 21, pp. 148-155, Jan-Feb 1998 1998.

[40] G.-R. Duan and G. Xu, "Direct parametric control approach to robust integrated relative position and attitude control for non-cooperative rendezvous," presented at the 2015 34th Chinese Control Conference, Hangzhou, China, 2015.

[41] L. Sun and W. Huo, "Robust adaptive control of spacecraft proximity maneuvers under dynamic coupling and uncertainty," *Advances in Space Research,* vol. 56, pp. 2206-2217, 2015.

[42] J. R. Wertz and R. Bell, "Autonomous rendezvous and docking technologies: status and propects," in *SPIE*, Orlando, FL, 2003, pp. 20-30.

[43] Y. Luo, Z. Jin, and G. Tang, "Survey of orbital dynamics and control of space rendezvous," *Chinese Journal of Aeronautics,* vol. 27, pp. 1-11, 2014.

[44] ESA, "Journey to the ISS Part 2: Soyuz rendezvous and docking explained," ed, 2014.

[45] M. Rott, Spacecraft Technology I: "Ascent Flight," Institute of Astronautics, TUM, 2015.

[46] M. Rott, Spacecraft Technology I: "Orbit Transfers," Institute of Astronautics, TUM, 2015.

[47] M. Rott, Spacecraft Technology I: "Astrodynamics," Institute of Astrodynamics, TUM, 2015.

[48] K.-D. Reiniger, Ground and User Segment: "Orbit," Institute of Astronomical and Physical Geodesy, TUM, 2015.

[49] R. Lampariello, On-Orbit Dynamics and Robotics: "Orbital dynamics and control," Institute of Astrodynamics, 2015.

[50] D. Gorinevsky, "Lecture 14 - Model Predictive Control Part 1: The concept," Stanford University, 2016.

[51] W. Wojsznis, "Model Predictive Control and Optimization," in *Instrument Engineers' Handbook: Process Control and Optimization*. vol. 2, B. Liptak, Ed., 4 ed Boca Raton, FL: CRC Press, 2006, pp. 242-251.

[52] M. Cannon, "C21 Model Predictive Control," Oxford University, 2016.

[53] M. S. Tšoeu, "Lecture Notes: EEE4093F - Model Predictive Control (MPC)," UCT, Ed., ed. Cape Town, South Africa, 2014.

[54] The MathWorks Inc. (2016). *Choosing Sample Time and Horizons*. Available: http://www.mathworks.com/help/mpc/ug/choosing-sample-time-and-horizons.html?s_tid=gn_loc_drop

[55] D. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica,* vol. 36, pp. 789-814, 2000.

[56] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Lecture Notes in Control and Information Sciences*. vol. 245, A. Garulli and A. Tesi, Eds., ed London: Springer London, 1999, pp. 207-226.

[57] F. Blanchini, "Set invariance in control," *Automatica,* vol. 35, pp. 1747-1767, 1991.

[58] S. V. Raković, E. Kerrigan, K. Kouramas, and D. Mayne, "Invariant approximations of the Minimal Robust Positively Invariant Set," *IEEE Transactions on Automatic Control,* vol. 50, pp. 406-420, 2005.

[59] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust Constrained Model Predictive Control using Linear Matrix Inequalities," in *1994 American Control Conference*, Baltimore, MD, USA, 1994, pp. 440-444.

[60] U. Jönsson, "A Lecture on the S-Procedure," Royal Institute of Technology, 2006.

[61] S. Boyd, EE363: Linear Dynamical Systems: "EE363 Review Session 4: Linear Matrix Inequalities," Sandford University, 2008.

[62] M. Grant and S. Boyd. (2016). *The CVX User's Guide, Release 2.1*.

[63] I. Kolmanovsky and E. C. Gilbert, "Theory and Computation of Distrubance Invariant Sets for Discrete-Time Linear Systems," *Mathematical Problems in Engineering,* vol. 4, pp. 317-367, 1998.

[64] E. C. Gilbert and K. T. Tan, "Linear systems with state and control constraints: The theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control,* vol. 36, pp. 1008-1020, 1991.

[65] E. Kerrigan, "Robust constraint satisfaction: Invariant sets and predictive control," Ph.D., Department of Engineering, St. John's College, University of Cambridge, Cambridge, UK, 2000.

[66] M. Kvasnica, P. Grieder, M. Baotic, and F. J. Christophersen. (2004). *Multi-Parametreic Toolbox (MPT)*. Available: http://people.ee.ethz.ch/~mpt/2/docs/

[67] S. Stoneman and R. Lampariello, "A nonlinear optimization method to provide real-time feasible reference trajectories to approach a tumbling target satellite," presented at the International symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS 2016), Beijing, P.R. China, 2016.

[68] eoPortal. *EnviSat*. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/e/envisat

[69] Clean Space, "e.deorbit Implementation Plan," ESA ESTEC 2015.

[70] S. p. A. ELV, ESA e.Deorbit Symposium: "Active debris removal by Vega upper state adaptation," Leeuwenhorst, Netherlands, 2014.

[71] M. Sánchez Nogales, e.Deorbit Symposium: "ELECNOR DEIMOS Participation in e.Deorbit and ADR Activities," Leeuwenhorst, Netherlands, 2014.

[72] A. Seel, "Error characterization of motion prediction of tumbling rigid bodies," Masters Thesis, Lehrstuhl für Astronomische und Physikalische Geodäsie, Ingenieurfakultät Bau Geo Umwelt, Technische Universität München Munich, 2016.

[73] P. J. Campo and M. Morari, "Robust Model Predictive Control," in *1987 American Control Conference*, Minneapolis, MN, USA, 1987, pp. 1021-1026.

Appendix: Software

In this appendix, the software used in the conduction of this work is given. Functions provided by the Model Predictive Control toolbox, Multi-parametric toolbox (MPT3), YALMIP, and CVX toolbox are frequently made use of in the following code.

The flow in the presentation of this code will follow similarly as to the rest of the work for easy reference. The program used to reproduce the double-pendulum controller will be presented first, followed by the controller design and simulation for the satellite rendezvous maneuver. There are some common functions between the two scenarios. These shared functions will be presented at the first instance and referred to at the later function call. The lines of code are numbered for easy in-text reference.

A. Double Pendulum

Recall from *chapter 5,* the double pendulum problem is a well-studied problem frequently used as a sample problem in literature. The problem was tackled in this chapter as a trial implementation of the control method. The software required for this implementation is given in this section.

The main function of this implementation conducts the controller design, controller construction, and controller simulation. Some portions of this design are identical in application in the satellite rendezvous scenario, and are therefore moved into separate function which can be called in both implementations. But first, the main function of the reproduction of the double pendulum problem is give below:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Double pendulum: tube-based robust MPC design and simulation                      %
% --------------------------------------------------------------------------------- %
% This script details the design process and simulation of the tube-based robust    %
% model predictive controller for the double-pendulum problem (refer to: chapter 6) %
% The process begins with the robust constraints. The disturbance rejection gain    %
% and then the mRPI set are determined. The tightened constraints are thereby given %
% Subsequently, the terminal (invariant set for tracking) and admissible sets and   %
% the region of attraction are determined. The controller is set implemented and    %
% simulated.                                                                        %
% --------------------------------------------------------------------------------- %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

1

```
clc
clear all


%% Define system matrices
A=[1,1;0,1];                                                          % state matrix
B=[0,0.5;1,0.5];                                                      % input matrix
C=[1 0];                                                             % output matrix
D=zeros(1,2);                                                   % feedthrough matrix

m=2;n=2;p=1;                                 % input, state, and output dimensions

%% Define robust constraints
Aw=[-1 0;0 -1;1 0;0 1];                                         % disturbance bound
bw=[0.1;0.1;0.1;0.1];
Ax=[-1 0;0 -1;1 0;0 1];                                   % robust state constraint
bx=[5;5;5;5];
Au=[-1 0;0 -1;1 0;0 1];                                  % robust input constraints
bu=[0.3;0.3;0.3;0.3];

scriptW=Polyhedron(Aw,bw);                   % build inequalities into polyhedrons
doubleX=Polyhedron(Ax,bx);
doubleU=Polyhedron(Au,bu);

%% Define weighting matrices for nominal MPC cost optimization
Q=eye(2);                                                          % state weights
R=10*eye(2);                                                       % input weights

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determine disturbance rejection gain Kdr
% refer to section 5.2.1
display(sprintf('\nDetermining K_dr CL dist rejection gain....\n'))

% Initializations for the optimization problem
lamdas=zeros(4);
a=[0;0];                                                            % zeros vector
p=0.48;                        % weight on control constraint (rho, refer to [33])
wa=[-0.1 -0.1];                                   % vertices of uncertainty bound
wb=[-0.1 0.1];
wc=[0.1 -0.1];
wd=[0.1 0.1];
l1=[-3.333333333333334 0];                   % rows of normalized state constraint
l2=[0 -3.333333333333334];
l3=[3.333333333333334 0];
l4=[0 3.333333333333334];
h1=[-0.2 0];                                 % rows of normalized input constraint
h2=[0 -0.2];
h3=[0.2 0];
h4=[0 0.2];


cvx_begin sdp quiet                                         % Optimization problem
    variable Y(2,2)                                             % maximizing Gamma
    variable W(2,2) symmetric                                      % to obtain Kdr
    variable Gamma(1,1) banded(0,1)

    lamda=0.49999;

    minimize (Gamma)
    subject to
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wa;(A*W+B*Y),wa',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wb;(A*W+B*Y),wb',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wc;(A*W+B*Y),wc',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wd;(A*W+B*Y),wd',W]>=0
        [p^2,(Y'*l1')';(Y'*l1'),W]>=0
        [p^2,(Y'*l2')';(Y'*l2'),W]>=0
        [p^2,(Y'*l3')';(Y'*l3'),W]>=0
        [p^2,(Y'*l4')';(Y'*l4'),W]>=0
        [Gamma,(W*h1')';(W*h1'),W]>=0
        [Gamma,(W*h2')';(W*h2'),W]>=0
        [Gamma,(W*h3')';(W*h3'),W]>=0
        [Gamma,(W*h4')';(W*h4'),W]>=0
cvx_end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine mRPI following from determination of Kdr
% refer to section 5.2.2, Algorithms 5-1 and 5-2, [33, 58]
display(sprintf('Determining mRPI....\n'))

Ak=A+B*Y*inv(W);

H=[-0.1 0;0 0.1];                                     % Non-singular matrix to define
                                                      % disturbance bound as zonotope
eps=0.01;                                             % percentage difference

s=1;                                                  % initialize s

alpha_goal=norm(inv(H)*(Ak^(s))*H,Inf);         % find initial alpha from support function

Mplus=zeros(n,1);Mminus=zeros(n,1);         % initialize sum and difference of error bound
As=A^(s-1);                                           % Determine initial error bound

for count=1:n
   Mplus(count)=Mplus(count)+supportfunc(Aw,bw,As(count,:)');      % From support function
                                                                   % as in [58]
   Mminus(count)=Mminus(count)+supportfunc(Aw,bw,-As(count,:)');   % see below for
                                                                   % supportfunc()
end

Msum=max(max(Mplus),max(Mminus));

display('Current ratio alpha/(eps/(eps+Ms)) is:')

while alpha_goal >= ((eps)/(eps+Msum))      % While alpha is greater than the error bound

    fprintf('\t%d \t %d \t %d\n',(eps/(eps+Msum)), alpha_goal, Msum)

    s=s+1;                                              % Increment s

    alpha_goal=norm(inv(H)*(Ak^(s))*H,Inf);   % Calculate the new alpha from support func
    As=A^(s-1);                                         % Raise Ak to current power of s-1

   for count1=1:n                                       % Determine new error bound
      Mplus(count1)=Mplus(count1)+supportfunc(Aw,bw,As(count1,:)');
      Mminus(count1)=Mminus(count1)+supportfunc(Aw,bw,-As(count1,:)');
   end

   Msum=max(max(Mplus),max(Mminus));

end

%% Refer to Algorithm 5-2
Bb=Polyhedron([1 0;0 1;-1 0;0 -1],[1;1;1;1]);  % Unit norm ball for definition of scriptW
Bi=Bb;                                          % as a zonotope
for i=1:1:s-1
    Bi=Bi*Bb;
end
Hz=[];                                     % As in [33]
for i=1:1:s                                % Using obtained s,
    Hz=[Hz Ak^(s-i)*H];
end
Z=plus(((1-alpha_goal)^-1)*Hz*Bi,(eye(2)-Ak)^-1*[0;0]);      % Obtain mRPI


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Pdr=inv(W);
Kdr=Y*inv(W);                                   % Disturbance rejection gain found above

Utight=minus(doubleU,mtimes(Kdr,Z));            % Determine tightened constraints
KZ=mtimes(Kdr,Z);                               % equations (5-31)-(5-32)
Xtight=minus(doubleX,Z);

display(sprintf('Pdr, Kdr, mRPI, and tightened constraints Xtight and Utight  ...
obtained....\n'))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Now determine invariant set for tracking
% Refer to section 5.4.2, Algorithm 5-4, [2, 63]
display(sprintf('Determining MRPI - invariant set for tracking....\n'))% repeating system
A=[1,1;0,1];B=[0,0.5;1,0.5];C=[1 0];D=zeros(1,2);m=2;n=2;p=1;
sys=ss(A,B,C,[],-1);

% Split the polyhedrons into A matrix and b vector of inequalities
doubleZ=mtimes(Xtight,Utight);
Az=doubleZ.H(:,1:4);
bz=doubleZ.H(:,5);

Autight=Utight.H(:,1:2);
butight=Utight.H(:,3);
Axtight=Xtight.H(:,1:2);
bxtight=Xtight.H(:,3);

[Klqr Plqr e]=dlqr(A,B,Q,R);                  % LQR parameters, section 5.4.1
Klqr=-Klqr;

% Defining CL system
Mtheta=[1 0 0 0;0 1 1 -2]';                    % Parameterization matrices for set points
Ntheta=[1 0];                                   % and targets

L=[-Klqr eye(2)]*Mtheta;                        % Refer to (4-54)
Ae=[A+B*Klqr B*L;zeros(2,2) eye(2)];

% Determine MRPI
lambda=0.99;
Oinf=MRPIsetforTracking(Ae,Klqr,Mtheta,Xtight,Utight,lambda,n,m);      % function defined
                                                                        % below
Ox=Oinf.projection(1:2);


%% Admissible states, inputs, outputs
Theta=Polyhedron(Az*Mtheta,bz);
Xa=affineMap(Theta,...
    [eye(size(Axtight,2)),zeros(size(Axtight,2),size(Autight,2))]*Mtheta);
Ua=affineMap(Theta,...
    [zeros(size(Autight,2),size(Axtight,2)) eye(size(Autight,2))]*Mtheta);
Ya=affineMap(Theta,Ntheta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Build controller

display(sprintf('Building controller....\n'))

display(sprintf('\tInit....\n'))                    % structure of parameters
params.Q=Q;                                         % state weights
params.R=R;                                         % input weights
params.P=Plqr;                                      % terminal weights
params.Mtheta=Mtheta;                               % parameterization matrix
params.Wscript=scriptW;                             % disturbance bound
params.Z=Z;                                         % mRPI
params.K=Kdr;                                       % disturbance rejection gain
params.Xtight=Xtight;                               % tightened state constraint
params.Utight=Utight;                               % tightened input constraint
params.Oinf=Oinf;                                          % invariant set for tracking
params.T=1000*Plqr;                                        % offset weight
params.N=10;                                               % prediction horizon

display(sprintf('\tsending to controller....\n'))
ctrl=TBRMPCcontroller(sys,params);                  % Make controller, function defined below

%% Region of attraction
Xrat=regAttraction(sys,Ox,Xtight,Utight,params.N);           % nominal
Xra=Xrat+Z;                                                       % robust

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Simulating controller

display(sprintf('Simulating....\n'))
display(sprintf('\tPart 1:\n'))
display(sprintf('\tsetting initial conditons, sim horizon, set point,
disturbance....\n'))

x0=[-3;1.5];                                                       % Initial state
Nsim1=20;                                                          % Simulation length
theta1=[-4;0];                                                    % First set point

w1=-0.1*ones(length(x0),Nsim1)+0.2*rand(length(x0),Nsim1); % Define disturbance


display(sprintf('\tsending to simulator....\n'))
[x1,z1,u1,y1]=TBRMPCsimulator(sys,ctrl,Nsim1,x0,theta1,w1); % Simulate,
                                                             % function defined below


display(sprintf('\n\tPart 2:\n'))
display(sprintf('\tsetting new sim horizon, set point, disturbances....\n'))

Nsim2=Nsim1;                                              % Simulation length remains the same
x01=[-3;3]                                                 % New initial state
theta2=[4;-0.5];                                          % New set point
w2=-0.1*ones(length(x0),Nsim2)+0.2*rand(length(x0),Nsim2); % Define disturbance

display(sprintf('\tsending to simulator...\n'))
[x2,z2,u2,y2]=TBRMPCsimulator(sys,ctrl,Nsim2,x1(:,end),theta2,w2);           % Simulate

display(sprintf('\n\tPart 3:\n'))
display(sprintf('\tconcatonating results of Part 1 and Part 2...\n'))

x=[x1,x2];                                   % Combine first and second simulations into
z=[z1,z2];                                   % one long simulation
u=[u1,u2];
y=[y1,y2];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PLOTTING FIGURES
% Extending mRPI to tube, refer to Algorithm 6-3

display(sprintf('Extending mRPI into tube....'))

tube=[];
for j=1:Nsim1+Nsim2
    tube=[tube, affmap(Z,eye(2),z(:,j))];
end

% ytube=[];
% for j=1:Nsim1+Nsim2
%    ytube=[ytube, affmap(Z,C,x(:,j))];
% end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Create visualizations
display(sprintf('Making pretty pictures....\n'))
display(sprintf('\tmRPI\n'))
figure;plot(Z);title('Z')
display(sprintf('\tcontrol constriants\n'))
figure;hold
on;doubleU.plot('color','red');Utight.plot('color','blue');KZ.plot('color','yellow');titl
e('control constraint')
display(sprintf('\tstate constraints\n'))
figure;hold
on;doubleX.plot('color','red');Xtight.plot('color','blue');Z.plot('color','yellow');title
('state constraint')
display(sprintf('\tinvariant set for tracking\n'))
figure;plot(Ox);title('invariant set for tracking')

display(sprintf('\tcombo state plot\n'))
figure;hold on;
display(sprintf('\t\tinvariant set for tracking\n'))
plot(Ox,'color','b');
display(sprintf('\t\tadmissible steady states\n'))
plot(Xa);

display(sprintf('\t\ttube\n'))
for k=1:Nsim1+Nsim2
   tube(k).plot('color','y');
end
display(sprintf('\t\tnominal and actual states\n'))
plot(z(1,:),z(2,:),'--xk');
plot(x(1,:),x(2,:),'--*k');

display(sprintf('\t\tnominal region of attraction\n'))
plot(Xra,'color','k','wire',1);
display(sprintf('\t\tdisturbed system region of attraction\n'))
plot(Xrat,'color','k','wire',1,'wirestyle','--');


axis([-5.5 5.5 -2.5 2.5])
title('x-space')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
display(sprintf('\tcombo actuation plot\n'))

figure;hold on;
display(sprintf('\t\tadmissible inputs\n'))
line([-0.1334 0.1334],[0.1334 -0.1334],'color','g','linewidth',2);
display(sprintf('\t\tboundaries of sets guaranteed to contain control input\n'))
display(sprintf('\t\t\t generated for uncertain system at steady state\n'))
display(sprintf('\t\t\tfor any admissible disturbance sequence\n'))
plot(affmap_robust(Z,Kdr),'color','k','wire',1,'wirestyle','--')
plot([-0.1334; 0.1334]+affmap_robust(Z,Kdr),'color','k','wire',1,'wirestyle','--')
display(sprintf('\t\t"trajectory" of inputs\n'))
plot(u(1,:),u(2,:),'d--','MarkerFaceColor',[0 0 1]);

axis([-0.35 0.35 -0.35 0.35])
title('u-space')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
display(sprintf('\toutput dynamics\n'))

target=[theta1(1,:)*ones(1,Nsim1),theta2(1,:)*ones(1,Nsim2)];

figure;hold on;
% plot(x(1,:))
% plot(y(1,:))
plot(z(1,:))
plot(target,'-')

title('target evolution')
xlabel('Time steps')
ylabel('Output')
```

```
legend('Output evolution','Output target')%'Artificial reference',

figure;hold on;plot(u)


% % figure;hold on;
% % for k=1:Nsim1+Nsim2
% %    ytube(k).plot('color','y');
% % end
%
% % figure;hold on;plot(plus(x(1,Nsim1+Nsim2),mtimes(C,Z)))
%
% figure;hold on;
% % plot(doubleX,'color','r')
% % plot(Xtight,'color','b')
% plot(Ya,'wire',1,'wirestyle','--')
% plot(x(1,:),x(2,:),'k*--')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
display(sprintf('...Fin...'))
```

The functions called by this main program are expressed in sequence. First, `supportfunc`, which returns the solution of the support function:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Support function                                                                   %
% ---------------------------------------------------------------------------------- %
% Simple optimization to solve support function via YALMIP.                          %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                     %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sup=supportfunc(H,k,a)
%% Optimization variables
w=sdpvar(size(H,2),1);
y=sdpvar(1,1);

%% Constraints
constraint=[a'*w>=y; H*w<=k];

%% Solve
solvesdp(constraint,-y,sdpsettings('verbose',0));
sup=double(y);
end
```

Second, the function for determining the nominal region of attraction, `regAttraction`, is given.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Region of attraction                                                              %
% --------------------------------------------------------------------------------- %
% Determine region of attraction for the controller.                                %
% --------------------------------------------------------------------------------- %
% function RegionAttractionTight=regAttraction(sys,Xf,Xtight,Utight,N)              %
%                                                                                   %
% Parameters:                                                                       %
%       sys     :  state space system                                               %
%       Xf      :  MRPI projected onto state dimensions                             %
%       Xtight  :  tightened state constraints                                      %
%       Utight  :  tightened input constraints                                      %
%       N       :  prediction horizon                                               %
% Returns:                                                                          %
%       RegionAttractionTight  :  region of attraction for the nominal controller   %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RegionAttractionTight=regAttraction(sys,Xf,Xtight,Utight,N)

%% unpacking system and polytopes
[A, B, C, D]=ssdata(sys);                              % System matrices
[n m]=size(B);                                         % State and input dimensions

f=size(Xf.H,2);                                % Terminal, state, input set dimensions
x=size(Xtight.H,2);
u=size(Utight.H,2);
Hf=Xf.H(:,1:f-1);                                      % Unpack terminal
kf=Xf.H(:,f);
Hx=Xtight.H(:,1:x-1);                                  % state
kx=Xtight.H(:,x);
Hu=Utight.H(:,1:u-1);                                  % input constraint sets
ku=Utight.H(:,u);

%% Constraint polytope in augmented state space
Ha=[Hf*A Hf*B; Hx zeros(size(Hx,1),m); zeros(size(Hu,1),n) Hu];
ka=[kf;kx;ku];
Aug=Polyhedron(Ha,ka);
pre=Aug.projection(1:n);

%% For prediction horizon
for i=1:N-1

    % Unpack augmented state space polytope
    p=size(pre.H,2);
    Hp=pre.H(:,1:p-1);
    kp=pre.H(:,p);

    % Determine pre-set
    Ha=[Hp*A Hp*B; Hx zeros(size(Hx,1),m); zeros(size(Hu,1),n) Hu];
    ka=[kp;kx;ku];
    Aug=Polyhedron(Ha,ka);
    pre=Aug.projection(1:n);

end
RegionAttractionTight=pre;
End
```

Next, `MRPIsetforTracking`, which determines the maximal RPI set to be used as the terminal constraint set for the prediction step of the nominal controller.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MRPI as the invariant set for tracking                                             %
% ---------------------------------------------------------------------------------- %
% Approximates MRPI from tightened state and input constraints.                      %
% ---------------------------------------------------------------------------------- %
% function Oinf = MRPIsetforTracking(A,K,M,X,U,lambda,n,m)                           %
%                                                                                    %
%   Parameters:                                                                      %
%        A      :  CL state matrix                                                   %
%        K      :  LQR gain                                                          %
%        M      :  parameterization matrix Mtheta                                    %
%        X      :  tightened state constraints                                       %
%        U      :  tightened input constraints                                       %
%        lambda :  arbitrary scale matrix to ensure feasibility                      %
%        n      :  dimension of states                                               %
%        m      :  dimension of inputs                                               %
%                                                                                    %
%   Returns: Oinf   :  invariant set for tracking                                    %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                     %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Oinf = MRPIsetforTracking(A,K,M,X,U,lambda,n,m)

%% Unpacking polytopes
    Hx=X.H(:,1:n);                                          % State constraint
    kx=X.H(:,n+1);
    Hu=U.H(:,1:m);                                          % Input constraint
    ku=U.H(:,m+1);

%% Build constraint set combining state + actuation spaces
    Hz=blkdiag(Hx,Hu);
    kz = [kx; ku];

%% Incorporate parameterized θ-space
    Ho=blkdiag(Hx,Hz*M);
    ko = [kx; lambda*kz];
    O = Polyhedron(Ho,ko);

%%  Compute presets
    O1 = preset(O,A,K,M,X,U,n,m);
    O2 = preset(O1,A,K,M,X,U,n,m);

%% Iteratively compare presets until Maximal set is found
    while O2~=O1

        O1 = O2;
        O2 = preset(O1,A,K,M,X,U,n,m);

    end
    Oinf = O2;
end


%% Function for computing pre-sets
function PreSet = preset(O,A,K,M,X,U,n,m)

% Unpack polytopes
    Hx=X.H(:,1:n);                                          % State constraint
    kx=X.H(:,n+1);
    Hu=U.H(:,1:m);                                      % Input constraint
    ku=U.H(:,m+1);
    Ho=O.A;                                             % Set incorporating θ-space
    ko=O.b;

    L = [-K eye(m)]*M;


%  Build preset inequalities
    Hp=[Ho*A; [Hx zeros(size(Hx,1),size(L,2))]; Hu*[K L]];
    kp=[ko;kx;ku];

% convert to polyhedron
    PreSet=Polyhedron(Hp,kp);

end

```

Now, the function which builds the MPT controller `TBRMPCcontroller` is given.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Double Pendulum: Tube-based robust model predictive controller                      %
% ----------------------------------------------------------------------------------- %
% This function constructs a controller object with the additional tube-based robust%
% MPC constraints and offset cost.                                                    %
% ----------------------------------------------------------------------------------- %
% function ctrl=TBRMPCcontroller(sys,params)                                          %
%                                                                                     %
%   Parameters:                                                                       %
%       sys     :  MATLAB discrete state space object                                 %
%       params  :  parameter structure containing the fields:                         %
%                 .Xtight    :  tightened state constraint set                        %
%                 .Utight    :  tightened ctrl constraint set                         %
%                 .Oinf      :  terminal constraint set / invariant set for tracking%
%                 .W         :  disturbance bounds                                    %
%                 .Z         :  approximated mRPI set                                 %
%                 .Mtheta    :  matrix describing parametrization                     %
%                 .Q         :  state weighting matrix                                %
%                 .R         :  control input weighting matrix                        %
%                 .P         :  terminal weighting matrix                             %
%                 .T         :  offset weighting matrix                               %
%                 .Kdr       :  disturbance rejection controller                      %
%                 .N         :  prediction horizon                                    %
%                                                                                     %
%   Returns:      ctrl       :  the controller - YALMIP optimization object           %
%                                                                                     %
% ----------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                      %
%                                                                                     %
% Copyright: Caroline Buckner                                                         %
% Last edited: 22 Dec 2016                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function ctrl=TBRMPCcontroller(sys,params)

%% Unpack system
display(sprintf('\twelcome to controller...\n'))
display(sprintf('\t\tunpacking system....\n'))
[A B C D]=ssdata(sys);                                      % System matrices
[n m]=size(B);                                          % State and input dimensions
nu=1;                                                   % Control horizon = 1

%% Unpack sets
display(sprintf('\t\tunpacking sets....\n'))
Ho=params.Oinf.H(:,1:4);                                 % Invariant set for tracking
ko=params.Oinf.H(:,5);

Hz=params.Z.H(:,1:2);                                           % mRPI
```

```
kz=params.Z.H(:,3);

Axtight=params.Xtight.H(:,1:2);                            % Tightened state constraints
bxtight=params.Xtight.H(:,3);

Autight=params.Utight.H(:,1:2);                           % Tightened input constraints
butight=params.Utight.H(:,3);

display(sprintf('\t\tbuild optimization problem...\n'))

%% Optimization variables
v=sdpvar(repmat(m,1,params.N),repmat(1,1,params.N));      % Nominal actuation
z=sdpvar(repmat(n,1,params.N+1),repmat(1,1,params.N+1));        % Nominal state
x=sdpvar(n,1);                                                  % Real state
theta_bar=sdpvar(2,1);                                    % Artificial steady state
theta=sdpvar(2,1);                                        % Steady state


%% Helper matrices - breakdown space into state and actuation
Mx=[eye(n) zeros(n,m)]*params.Mtheta;
Mu=[zeros(m,n) eye(m)]*params.Mtheta;


display(sprintf('\t\tbuild objective function and constraints....\n'))
%% Objective functions and constraints
objfunc=0;
constraint=[];

% k=1:N step cost
for k=1:params.N
   objfunc=objfunc+0.5*((z{k}-Mx*theta_bar)'*params.Q*(z{k}-Mx*theta_bar)+...
       (v{k}-Mu*theta_bar)'*params.R*(v{k}-Mu*theta_bar));
   constraint=[constraint, z{k+1}==A*z{k}+B*v{k}];
   constraint=[constraint, Axtight*z{k}<=bxtight, Autight*v{k}<=butight];
end
% Terminal cost k=N+1
objfunc=objfunc+0.5*(z{params.N+1}-Mx*theta_bar)'*params.P*(z{params.N+1}-Mx*theta_bar);
constraint=[constraint, Ho*[z{params.N+1};theta_bar]<=ko];
% Offset cost
objfunc=objfunc+0.5*((theta_bar-theta)'*params.T*(theta_bar-theta));
% Terminal set constraint
constraint=[constraint, Hz*(x-z{1})<=kz];

% Nominal output
nomout=[v{1}+params.K*(x-z{1});z{1}];

% Controller object
display(sprintf('\t\tsending to optimizer....\n'))
options=sdpsettings('verbose',0);
ctrl=optimizer(constraint,objfunc,options,[x,theta],nomout);

display(sprintf('\t\tleaving controller....\n'))

end
```

And finally, the function used for the simulation of the controller, `TBRMPCsimulator`:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Double Pendulum: Tube-based robust model predictive controller simulation            %
% ---------------------------------------------------------------------------------- %
%This function simulates the control behaviour for a given set point and init state.%
% ---------------------------------------------------------------------------------- %
% function [x,z,u,y] = TBRMPCsimulator(sys,ctrl,Nsim,x0,theta,w)                       %
%                                                                                      %
%   Parameters:                                                                        %
%       sys     :  MATLAB discrete state space object                                  %
%       params  :  parameter structure containing the fields:                          %
%                 .Xtight     :  tightened state constraint set                        %
%                 .Utight     :  tightened ctrl constraint set                         %
%                 .Oinf       :  terminal constraint set / invariant set for tracking%
%                 .W          :  disturbance bounds                                    %
%                 .Z          :  approximated mRPI set                                 %
%                 .Mtheta     :  matrix describing parametrization                     %
%                 .Q          :  state weighting matrix                                %
%                 .R          :  control input weighting matrix                        %
%                 .P          :  terminal weighting matrix                             %
%                 .T          :  offset weighting matrix                               %
%                 .Kdr        :  disturbance rejection controller                      %
%                 .N          :  prediction horizon                                    %
%                                                                                      %
%   Returns:      ctrl        :  the controller - YALMIP optimization object           %
%                                                                                      %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%           of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,        %
%           Technische Universität München, 2016.                                      %
%                                                                                      %
% Copyright: Caroline Buckner                                                          %
% Last edited: 22 Dec 2016                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,z,u,y] = TBRMPCsimulator(sys,ctrl,Nsim,x0,theta,w)

%% Unpack system
display(sprintf('\twelcome to simulator....\n'))
display(sprintf('\t\tunpacking system....\n'))
[A B C D]=ssdata(sys);                                       % System matrices
[n,m]=size(B);                                               % state and input dimensions

%% Initialize solution vectors
display(sprintf('\t\tbuilding sim vectors....\n'))
x=[x0,zeros(n,Nsim)];                                        % Real state
z=zeros(n,Nsim);                                              % Nominal state
u=zeros(m,Nsim);                                              % Robust actuation
y=zeros(1,Nsim);                                               % Output

%% Control action at each simulation step
display(sprintf('\t\tctrl at ea time step....\n'))
for k=1:Nsim

    op=ctrl{[x(:,k) theta]};                          % Call controller

    if max(isnan(op))==1                             % Exception catching for infeasibility
        error(['Problem has become infeasible'])
    end

    u(:,k)=op(1:m);                                   % Robust actuation from controller
    z(:,k)=op(m+1:end);                               % Nominal state from controller
    x(:,k+1)=A*x(:,k)+B*u(:,k)+w(:,k);       % Determine real successive state from system
    y(:,k)=C*x(:,k);                                  % System output
end
end
```

B. Satellite Rendezvous controller design

The controller for the satellite rendezvous problem was designed in *chapter 7.* The designed parameters and sets were determined using the following script. Any function calls made within this program will be referenced to a function defined in the preceding section or given following this main function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Satellite Rendezvous: tube-based robust MPC design script                          %
% ---------------------------------------------------------------------------------- %
% This script details the design process of the tube-based robust model predictive  %
% controller for the satellite rendezvous problem (refer to: Part 2, particularly    %
% chapter 8). The process begins with the robust constraints. The disturbance        %
% rejection gain and then the mRPI set are determined. The tightened                 %
% constraints are thereby given. Subsequently, the terminal (invariant set           %
% for tracking) and admissible sets and the region of attraction are                 %
% determined. The nominal controller is set as an mpc object in                      %
% 'SatRend_controller_MPC_toolbox_initscript.m'                                      %
% ---------------------------------------------------------------------------------- %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                     %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear all

%% Define discrete model
nOrbit=0.0012;                                                          % orbital rate

Ac=[         0 0          0         1        0 0                     % Continuous state matrix
             0 0          0         0        1 0
             0 0          0         0        0 1
    3*nOrbit^2 0          0         0 2*nOrbit 0
             0 0          0 -2*nOrbit        0 0
             0 0 3*nOrbit^2         0        0 0];

Bc=[0 0 0                                                          % Continuous input matrix
    0 0 0
    0 0 0
    1 0 0
    0 1 0
    0 0 1];
Cc=eye(6);                                                         % Continuous output matrix

sys=ss(Ac,Bc,Cc,[],0.5);                                  % Discretized state space, Ts=0.5

[A,B,C,D]=ssdata(sys);
n=6;                                                                 % Model matrix dimensions
m=3;                                                                 % (refer to chapter 3)
p=6;

%% Define polytopic constraints
% State constraint - relative position and velocity
Ax=[-1 0 0 0 0 0
    1 0 0 0 0 0
    0 -1 0 0 0 0
    0 1 0 0 0 0
    0 0 -1 0 0 0
    0 0 1 0 0 0
    0 0 0 -1 0 0
    0 0 0 1 0 0
    0  0  0  0  -1  0
    0  0  0  0  1  0
    0  0  0  0  0  -1
    0  0  0  0  0  1];
bx=[100
    100
    100
    100
    100
    100
    1
    1
    1
    1
    1
    1];

% Actuation constraint
%|F|max=65N -> 4.3e-2 m/s^2
Au=[-1  0  0
    1  0  0
    0  -1  0
    0  1  0
    0  0  -1
    0  0  1];
bu=0.0433333*ones(6,1);



% Disturbance bound
r=4.6662;                                    % radius of sphere (refer to chapter 7)
Aw=[-1  0  0  0  0  0
    1  0  0  0  0  0
    0  -1  0  0  0  0
    0  1  0  0  0  0
    0  0  -1  0  0  0
    0  0  1  0  0  0
    0  0  0  1  0  0
    0  0  0  -1  0  0
    0  0  0  0  1  0
    0  0  0  0  -1  0
    0  0  0  0  0  1
    0  0  0  0  0  -1];

bw=[r*ones(6,1); zeros(6,1)];

% Create state, input, disturbance polyhedrons
doubleX=Polyhedron(Ax,bx);
doubleU=Polyhedron(Au,bu);
scriptW=Polyhedron(Aw,bw);

%% LQR parameters
% Values initially chosen. Tuned to:
Q=10e5*eye(6);                                                      % State weights
R=100*eye(3);                                                       % Actuation weights
[Klqr Plqr e]=dlqr(A,B,Q,R);            % determine LQR gain and sol. to Riccatti equation
Klqr=-Klqr;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of the disturbance rejection gain
% Refer to section 5.2.1
display(sprintf('\nDetermining K_dr CL dist rejection gain....\n'))

% Set constants
a=zeros(6,1);                                % zeros vector
p=0.4;                            % weight on control constraint (rho, refer to [25])
wVert=scriptW.V;                             % vertices of disturbance bound
l=(1/(65/1500))*Au;                          % normalized input
```

```
h=[(1/100)*Ax(1:6,:); Ax(7:12,:)];               % and state constraint inequalities

cvx_begin sdp quiet                               % optimization problem:
    variable Y(3,6)
    variable W(6,6) symmetric
    variable Gamma(1,1) banded(0,1)

    lamda=1e-5;

    minimize (Gamma)                              % minimize optimization parameter Gamma
    subject to
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(1,:);(A*W+B*Y),wVert(1,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(2,:);(A*W+B*Y),wVert(2,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(3,:);(A*W+B*Y),wVert(3,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(4,:);(A*W+B*Y),wVert(4,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(5,:);(A*W+B*Y),wVert(5,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(6,:);(A*W+B*Y),wVert(6,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(7,:);(A*W+B*Y),wVert(7,:)',W]>=0
        [lamda*W,a,(A*W+B*Y)';a',1-lamda,wVert(8,:);(A*W+B*Y),wVert(8,:)',W]>=0
        [p^2,(Y'*l(1,:)')';(Y'*l(1,:)'),W]>=0
        [p^2,(Y'*l(2,:)')';(Y'*l(2,:)'),W]>=0
        [p^2,(Y'*l(3,:)')';(Y'*l(3,:)'),W]>=0
        [p^2,(Y'*l(4,:)')';(Y'*l(4,:)'),W]>=0
        [p^2,(Y'*l(5,:)')';(Y'*l(5,:)'),W]>=0
        [p^2,(Y'*l(6,:)')';(Y'*l(6,:)'),W]>=0
        [Gamma,(W*h(1,:)')';(W*h(1,:)'),W]>=0
        [Gamma,(W*h(2,:)')';(W*h(2,:)'),W]>=0
        [Gamma,(W*h(3,:)')';(W*h(3,:)'),W]>=0
        [Gamma,(W*h(4,:)')';(W*h(4,:)'),W]>=0
        [Gamma,(W*h(5,:)')';(W*h(5,:)'),W]>=0
        [Gamma,(W*h(6,:)')';(W*h(6,:)'),W]>=0
cvx_end                                              % To obtain a W and Y which give the
Kdr=Y*inv(W);                                        % disturbance rejection gain
Pdr=inv(W);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now use Kdr to determine mRPI set
% refer to section 5.2.2, Algorithms 5-1 and 5-2, [33, 63]

display(sprintf('Determining OPT mRPI....\n'))

Akc=A+B*Kdr;

H=[r 0 0 0 0 0                                           % Non-singular matrix to define
    0 r 0 0 0 0                                          % disturbance bound as zonotope
    0 0 r 0 0 0
    0 0 0 1 0 0
    0 0 0 0 1 0
    0 0 0 0 0 1];

eps=0.01;                                                % Percentage difference

s=1;

alpha_goal=norm(inv(H)*(Akc^(s-1))*H,Inf);      % Find initial alpha from support function

Mplus=zeros(n,1);Mminus=zeros(n,1);         % Initialize sum and difference of error bound
As=Akc^(s-1);                               % Determine initial error bound

for count=1:n
   Mplus(count)=Mplus(count)+supportfunc(Aw,bw,As(count,:)');    % refer to definition of
   Mminus(count)=Mminus(count)+supportfunc(Aw,bw,-As(count,:)');   %  supportfunc() above
end

Msum=max(max(Mplus),max(Mminus));

display('Current (eps/(eps+Msum)), alpha_goal, Msum are:')

while alpha_goal >= ((eps)/(eps+Msum));      % While alpha is greater than the error bound
```

```

     fprintf('\t%d \t %d \t %d\n',(eps/(eps+Msum)), alpha_goal, Msum)

    s=s+1;                                            % Increment s

    alpha_goal=norm(inv(H)*(Akc^(s-1))*H,Inf);        % Calculate the new alpha from
                                                       % support function
    As=Akc^(s-1);                                     % Raise Akc to current power of s

   for count1=1:n
       Mplus(count1)=Mplus(count1)+supportfunc(Aw,bw,As(count1,:)');     % Determine new
                                                                          % error bound
       Mminus(count1)=Mminus(count1)+supportfunc(Aw,bw,-As(count1,:)');
   end

   Msum=max(max(Mplus),max(Mminus));

End
Ba=[1  0  0  0  0  0                               % Unit norm ball for definition of scriptW
    0  1  0  0  0  0                               % as a zonotope
    0  0  1  0  0  0
    0  0  0  1  0  0
    0  0  0  0  1  0
    0  0  0  0  0  1
    -1  0  0  0  0  0
    0  -1  0  0  0  0
    0  0  -1  0  0  0
    0  0  0  -1  0  0
    0  0  0  0  -1  0
    0  0  0  0  0  -1];
Bb=Polyhedron(Ba,[ones(3,1);zeros(3,1);ones(3,1);zeros(3,1)]);
Bi=Bb;
for i=1:1:s-1
    Bi=Bi*Bb;
end
Hz=[];
for i=1:1:s                                                             % Using obtained s,
    Hz=[Hz Akc^(s-i)*H];
end
Z=plus(((1-alpha_goal)^-1)*Hz*Bi,(eye(n)-Akc)^-1*a);                          % Obtain mRPI

%% Tightened constraints
Xtight=minus(doubleX,Z);                                             % equations (7-14)-(7-17)
KZ=mtimes(Kdr,Z);
Utight=minus(doubleU,KZ);
display('tight constraints obtained')

% Split the polyhedrons into A matrix and b vector of inequalities
doubleZ=mtimes(Xtight,Utight);
Az=doubleZ.H(:,1:9);
bz=doubleZ.H(:,10);

Autight=Utight.H(:,1:3);
butight=Utight.H(:,4);
Axtight=Xtight.H(:,1:6);
bxtight=Xtight.H(:,7);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PARAMETERIZATION of targets and set points
display('parameterizing')
[Mtheta Ntheta]=ssCharacterization(sys);                      % Refer to following function
                                                              % definition
L=[-Klqr eye(m)]*Mtheta;                                      % Form CL matrices, refer to
(4-54)
Ae=[A+B*Klqr B*L;zeros(n,n) eye(n)];                          % CL system

%% Determine invariant set for tracking
display('set for tracking')
lambda=0.99;
```

```
Oinf=MRPIsetforTracking(Ae,Klqr,Mtheta,Xtight,Utight,lambda,n,m);      % Refer to function
                                                                       % definition above
display('projecting')
Ox=projection(Oinf,1:length(Ac));                             % Project into state space
display('Ox Obtained')

%% Admissible states, inputs, outputs
Theta=Polyhedron(Az*Mtheta,bz);
Xa=affineMap(Theta,...
    [eye(size(Ax,2)),zeros(size(Ax,2),size(Au,2))]*Mtheta);
Ua=affineMap(Theta,...
    [zeros(size(Au,2),size(Ax,2)) eye(size(Au,2))]*Mtheta);
Ya=affineMap(Theta,Ntheta);




%% Region of attraction
display(sprintf('reg attraction'))
Xrat=regAttraction(sys,Ox,Xtight,Utight,40);                       % Refer to function
Xra=Xrat+Z;                                                        % definition above
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Control parameters
display(sprintf('Building controller object....\n'))

display(sprintf('\tInit....\n'))
params.Q=Q;                                                             % State weights
params.R=R;                                                             % Input weights
params.P=Plqr;                                                     % LQR sol to Riccatti eq
params.W=scriptW;                                                       % uncertainty bound
params.Z=Z;                                                             % mRPI
params.K=Kdr;                                                  % disturbance rejection gain
params.Xtight=Xtight;                                          % tightened state constraint
params.Utight=Utight;                                          % tightened input constraint
params.Oinf=Oinf;                                              % invariant set for tracking
params.T=1000*Plqr;                                                     % offset weight
params.N=40;                                                            % prediction horizon
params.Nsim=1201;                           % simulation horizon in [timesteps], = 600 s
z_s1=[1.97;0;4.23;0;0;0;0;0;0];                                         % nominal set point
params.theta=Mtheta'*z_s1;                                    % the previous in theta space
params.Mtheta=Mtheta;                                         % parameterization matrices
params.Ntheta=Ntheta;
refdata=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05se
c_1.dat');                                                            % reference data

%% Initialize controller
SatRend_controller_MPC_toolbox_initscript             % Refer to the following definition
```

This main function calls two functions additional to those described in part A. First, the function `ssCharacterization` determines the parameterization matrices M_θ and N_θ.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameterization method for relating between targets and set points             %
% -------------------------------------------------------------------------------- %
% Parameterizing the relationship of targets and set points using the method       %
% given in [1]                                                                     %
% -------------------------------------------------------------------------------- %
% function [Mtheta,Ntheta]=ssCharacterization(sys)                                 %
%                                                                                  %
%     Parameter: sys: state space system                                           %
%                                                                                  %
%     Returned:                                                                    %
%           Mtheta : matrix relating states+actuation to parameterization          %
%                         varbiale Theta                                           %
%           Ntheta : matrix relating output to parameterization                    %
%                         varbiale Theta                                           %
% -------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver  %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,     %
%          Technishe Universität München, 2016.                                    %
%                                                                                  %
% Copyright: Caroline Buckner                                                      %
% Last edited: 22 Dec 2016                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Mtheta,Ntheta]=ssCharacterization(sys)
% unpack system
[A,B,C,D]=ssdata(sys);
n=size(A,2);
m=size(B,2);
p=size(C,1);

% Build E and F (refer to section 4.4.2
E=[A-eye(n) B;C zeros(p,m)];
F=[zeros(n,p); eye(p)];
r=rank(E);

% SVD on matrix E
[U,S,V]=svd(E);
Up = U(:,r+1:end);
U = U(:,1:r);
Vp = V(:,r+1:end);
V = V(:,1:r);
S = S(1:r,1:r);

% Build Mtheta, Ntheta
if r==(n+p)
   G=eye(p);
else
    G=[F'*Up null((F'*Up)')];
end

if r==(n+m)
    Mtheta=V/S*U'*F*G;
    Ntheta=G;
else
    Mtheta=[V/S*U'*F*G Vp];
    Ntheta=[G zeros(p,n+m-r)];
end
end
```

Second, the initialization script for the mpc controller object to be used in simulation, `SatRend_controller_MPC_toolbox_initscript`.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mpc Controller Object initiation script                                          %
% -------------------------------------------------------------------------------- %
% This script initializes the Model Predictive Control toolbox controller object    %
% to be used with the accompanying simulation. Sample time = 0.5 s.                 %
% -------------------------------------------------------------------------------- %
% -------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initialize constants

n=0.0012;                                              % target orbital rate [rad/s]
% Nsim=600;                                            % simulation horizon [s]
Ts=0.5;                                                % sample time [s]
                                                       % initial state
dx=39;
dy=39;
dz=4.23;
ddx=0;
ddy=0;
ddz=0;

%% SS model of the system
% Continuous time model
A=[      0  0       0    1    0  0                         % State matrix
         0  0       0    0    1  0
         0  0       0    0    0  1
     3*n^2  0       0    0  2*n  0
         0  0       0 -2*n    0  0
         0  0 -3*n^2    0    0  0];
 B=[0  0  0                                                % Input matrix
    0  0  0
    0  0  0
    1  0  0
    0  1  0
    0  0  1];
C=eye(6);                                                  % Output matrix
D=zeros(6,3);                                              % Feedthrough matrix

sys=ss(A,B,C,[]);                                        % Combine into a state space
model/object
[A,B,C,D]=ssdata(sys);

% Discretized SS model
sysd=c2d(sys,0.5)
[Ad,Bd,Cd,Dd]=ssdata(sysd);

% Extended SS for MPC, incorporating offset cost
A=[         1  0         0      0.5  0.0003    0  0  0  0  0  0  0;
   -2.16e-10  1         0  -0.0003     0.5    0  0  0  0  0  0  0;
            0  0         1        0       0  0.5  0  0  0  0  0  0;
     2.16e-6  0         0        1  0.0012    0  0  0  0  0  0  0;
   -1.296e-9  0         0  -0.0012       1    0  0  0  0  0  0  0;
            0  0  -2.16e-6        0       0    1  0  0  0  0  0  0
            0  0         0        0       0    0  0  0  0  0  0  0
            0  0         0        0       0    0  0  0  0  0  0  0
            0  0         0        0       0    0  0  0  0  0  0  0
            0  0         0        0       0    0  0  0  0  0  0  0
            0  0         0        0       0    0  0  0  0  0  0  0
            0  0         0        0       0    0  0  0  0  0  0  0];
```

```
B=[   0.125   5e-5     0;
      -5e-5  0.125     0;
          0      0 0.125;
        0.5 0.0003     0;
    -0.0003    0.5     0;
          0      0   0.5
          0      0     0
          0      0     0
          0      0     0
          0      0     0
          0      0     0
          0      0     0];
C=[eye(12)];
D=zeros(12,3);

%% SS model for control
sysx=ss(A,B,C,D,0.5);
sysy=minreal(sysx);

%% mpc Object
controller_tumbling3d=mpc(sysx,0.5,40,1)

%% Controller constraints
% Horizons
controller_tumbling3d.PredictionHorizon=40;
controller_tumbling3d.ControlHorizon=1;



% Nominal actuation min/max constraints
controller_tumbling3d.MV(1).Min=-0.043313455523267;
controller_tumbling3d.MV(1).Max=0.043313455525738;
controller_tumbling3d.MV(2).Min=-0.043333299864778;
controller_tumbling3d.MV(2).Max=0.043333300135222;
controller_tumbling3d.MV(3).Min=-0.043313449862363;
controller_tumbling3d.MV(3).Max=0.043313449859726;

% Nominal state constraints
controller_tumbling3d.OV(1).Min=-95.333305708733178;
controller_tumbling3d.OV(1).Max=95.333306340018140;
controller_tumbling3d.OV(2).Min=-95.333306023326998;
controller_tumbling3d.OV(2).Max=95.333306025424321;
controller_tumbling3d.OV(3).Min=-95.333305714396644;
controller_tumbling3d.OV(3).Max=95.333306334354674;
controller_tumbling3d.OV(4).Min=-1.000000000000000;
controller_tumbling3d.OV(4).Max=0.999999684357523;
controller_tumbling3d.OV(5).Min=-1.000000000892764;
controller_tumbling3d.OV(5).Max=0.999999999107236;
controller_tumbling3d.OV(6).Min=-1.000000000000000;
controller_tumbling3d.OV(6).Max=0.999999690020980;

% State and actuation weights
controller_tumbling3d.W.ManipulatedVariables=100*eye(3);
controller_tumbling3d.W.OutputVariables=[10e5 0 0 0 0 0 0 0 0 0 0 0
                                          0 10e5 0 0 0 0 0 0 0 0 0 0
                                          0 0 10e5 0 0 0 0 0 0 0 0 0
                                          0 0 0 10e5 0 0 0 0 0 0 0 0
                                          0 0 0 0 10e5 0 0 0 0 0 0 0
                                          0 0 0 0 0 10e5 0 0 0 0 0 0
                                          0 0 0 0 0 0 0 0 0 0 0 0
                                          0 0 0 0 0 0 0 0 0 0 0 0
                                          0 0 0 0 0 0 0 0 0 0 0 0
                                          0 0 0 0 0 0 0 0 0 0 0 0
                                          0 0 0 0 0 0 0 0 0 0 0 0
                                          0 0 0 0 0 0 0 0 0 0 0 0];

% Terminal state and offset weights and (invariant set for tracking)
% terminal constraint
Y=struct('Weight',[1.000000000031104,1,1.000000000003456,2.000009600000385,...
    2.000009600038402,2.000000000003456,1000*1.000000000031104,1000,...

1000*1.000000000003456,1000*2.000009600000385,1000*2.000009600038402,1000*2.0000000000034
56],...
    'Min',[-95.333305708733192,-95.333306023327012,-95.333305714396658,-1,-1,-1,...
            -0.276806311745190,-46.445315091220948,-2.757660444863814e+02,...
            -1.333304627419510e+02,-1.343679944982861e+02,-1.465918324789734e+02],...
    'Max',[95.333306340029267,95.333306025435476,95.333306334365830,...
            0.999999684357523,0.999999999107237,0.999999690020981,...
            6.086883384630116e+02,6.125513619825520e+02,4.495870645805429e+02,...
            7.589576390831120,7.416002199688840,0.066603242987696]);
U=struct('Weight',[10,10,10]);

setterminal(controller_tumbling3d,Y,U)

% Set inital conditions of controller
state=mpcstate(controller_tumbling3d,[39,39,4.23,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,
0],[],[0,0,0],[]);
```

The following scripts was used in the design the of parameter λ for the determination of K_{dr}.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Satellite Rendezvous Kdr design: determination of lambda                            %
% ----------------------------------------------------------------------------------- %
% This script produces an l-curve for the Kdr determining optimization problem.       %
% ----------------------------------------------------------------------------------- %
% ----------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver     %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,        %
%          Technishe Universität München, 2016.                                       %
%                                                                                     %
% Copyright: Caroline Buckner                                                         %
% Last edited: 22 Dec 2016                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Define discrete model
nOrbit=0.0012;                                                        % orbital rate

Ac=[         0  0           0         1        0  0                % Continuous state matrix
             0  0           0         0        1  0
             0  0           0         0        0  1
    3*nOrbit^2  0           0         0  2*nOrbit 0
             0  0           0  -2*nOrbit       0  0
             0  0  3*nOrbit^2         0        0  0];
Bc=[0  0  0                                                        % Continuous input matrix
    0  0  0
    0  0  0
    1  0  0
    0  1  0
    0  0  1];
Cc=eye(6);                                                     % Continuous output matrix

sys=ss(Ac,Bc,Cc,[],0.5);                                       % Discretized state space, Ts=0.5

[A,B,C,D]=ssdata(sys);

n=6;                                                               % Model matrix dimensions
m=3;                                                               % (refer to chapter 3)
p=6;

%% Define polytopic constraints
% State constraint - relative position and velocity
Ax=[-1  0  0  0  0  0
    1  0  0  0  0  0
    0  -1  0  0  0  0
    0  1  0  0  0  0
    0  0  -1  0  0  0
    0  0  1  0  0  0
    0  0  0  -1  0  0
    0  0  0  1  0  0
    0  0  0  0  -1  0
    0  0  0  0  1  0
    0  0  0  0  0  -1
    0  0  0  0  0  1];
bx=[100
    100
    100
    100
    100
    100
    1
    1
    1
    1
    1
    1];

% Actuation constraint
%|F|max=65N -> 4.3e-2 m/s^2
Au=[-1  0  0
    1  0  0
    0  -1  0
    0  1  0
    0  0  -1
    0  0  1];
bu=0.0433333*ones(6,1);


% Disturbance bound
r=4.6662;                                            % radius of sphere
Aw=[-1  0  0  0  0  0
    1  0  0  0  0  0
    0  -1  0  0  0  0
    0  1  0  0  0  0
    0  0  -1  0  0  0
    0  0  1  0  0  0
    0  0  0  1  0  0
    0  0  0  -1  0  0
    0  0  0  0  1  0
    0  0  0  0  -1  0
    0  0  0  0  0  1
    0  0  0  0  0  -1];

bw=[r*ones(6,1); zeros(6,1)];

% Create polyhedrons
doubleX=Polyhedron(Ax,bx);
doubleU=Polyhedron(Au,bu);
scriptW=Polyhedron(Aw,bw);




%% LQR parameters
% Values initially chosen. Tuned to:
Q=10e5*eye(6);                                                          % State weights
R=100*eye(3);                                                           % Actuation weights
[Klqr Plqr e]=dlqr(A,B,Q,R);          % determine LQR gain and sol. to Riccati equation
Klqr=-Klqr;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of the disturbance rejection gain
display(sprintf('\nDetermining K_dr CL dist rejection gain....\n'))

% Set constants
a=zeros(6,1);                                                                   % zeros vector
```

```
p=0.4;                               % weight on control constraint (rho, refer to [25])
wVert=scriptW.V;                              % vertices of disturbance bound
l=(1/(65/1500))*Au;                           % normalized input
h=[(1/100)*Ax(1:6,:); Ax(7:12,:)];            % and state constraint inequalities

    lamda=1e-5;
    k=0;
for lamda=1e-10:1e-9:1e-5                     % iteratively conduct optimization
    lamda                                     % store determined gamma for each
lambda
    k=k+1;                                    % to determine correct lambda to
minimize
cvx_begin sdp quiet                           % gamma
    variable Y(3,6)
    variable W(6,6) symmetric
    variable Gamma(1,1) banded(0,1)



    minimize (Gamma)
    subject to
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(1,:);(Ac*W+Bc*Y),wVert(1,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(2,:);(Ac*W+Bc*Y),wVert(2,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(3,:);(Ac*W+Bc*Y),wVert(3,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(4,:);(Ac*W+Bc*Y),wVert(4,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(5,:);(Ac*W+Bc*Y),wVert(5,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(6,:);(Ac*W+Bc*Y),wVert(6,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(7,:);(Ac*W+Bc*Y),wVert(7,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(8,:);(Ac*W+Bc*Y),wVert(8,:)',W]>=0
        [p^2,(Y'*l(1,:)')';(Y'*l(1,:)'),W]>=0
        [p^2,(Y'*l(2,:)')';(Y'*l(2,:)'),W]>=0
        [p^2,(Y'*l(3,:)')';(Y'*l(3,:)'),W]>=0
        [p^2,(Y'*l(4,:)')';(Y'*l(4,:)'),W]>=0
        [p^2,(Y'*l(5,:)')';(Y'*l(5,:)'),W]>=0
        [p^2,(Y'*l(6,:)')';(Y'*l(6,:)'),W]>=0
        [Gamma,(W*h(1,:)')';(W*h(1,:)'),W]>=0
        [Gamma,(W*h(2,:)')';(W*h(2,:)'),W]>=0
        [Gamma,(W*h(3,:)')';(W*h(3,:)'),W]>=0
        [Gamma,(W*h(4,:)')';(W*h(4,:)'),W]>=0
        [Gamma,(W*h(5,:)')';(W*h(5,:)'),W]>=0
        [Gamma,(W*h(6,:)')';(W*h(6,:)'),W]>=0
cvx_end

gamma1(k,:)=[lamda Gamma];

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Satellite Rendezvous Kdr design: determination of rho                                %
% ---------------------------------------------------------------------------------- %
% This script produces an l-curve for the Kdr determining optimization problem.      %
% ---------------------------------------------------------------------------------- %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technishe Universität München, 2016.                                      %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Define discrete model
nOrbit=0.0012;                                              % orbital rate

Ac=[         0 0          0         1        0 0          % Continuous state matrix
             0 0          0         0        1 0
             0 0          0         0        0 1
    3*nOrbit^2 0          0         0 2*nOrbit 0
```

```
             0 0          0 -2*nOrbit        0 0
             0 0 3*nOrbit^2         0        0 0];
Bc=[0 0 0                                                    % Continuous input matrix
    0 0 0
    0 0 0
    1 0 0
    0 1 0
    0 0 1];
Cc=eye(6);                                                 % Continuous output matrix

sys=ss(Ac,Bc,Cc,[],0.5);                                % Discretized state space, Ts=0.5

[A,B,C,D]=ssdata(sys);

n=6;                                                    % Model matrix dimensions
m=3;                                                    % (refer to chapter 3)
p=6;

%% Define polytopic constraints
% State constraint - relative position and velocity
Ax=[-1 0 0 0 0 0
    1 0 0 0 0 0
    0 -1 0 0 0 0
    0 1 0 0 0 0
    0 0 -1 0 0 0
    0 0 1 0 0 0
    0 0 0 -1 0 0
    0 0 0 1 0 0
    0 0 0 0 -1 0
    0 0 0 0 1 0
    0 0 0 0 0 -1
    0 0 0 0 0 1];
bx=[100
    100
    100
    100
    100
    100
    1
    1
    1
    1
    1
    1];

% Actuation constraint
%|F|max=65N -> 4.3e-2 m/s^2
Au=[-1 0 0
    1 0 0
    0 -1 0
    0 1 0
    0 0 -1
    0 0 1];
bu=0.0433333*ones(6,1);


% Disturbance bound
r=4.6662;                                         % radius of sphere (refer to chapter 7)
Aw=[-1 0 0 0 0 0
    1 0 0 0 0 0
    0 -1 0 0 0 0
    0 1 0 0 0 0
    0 0 -1 0 0 0
    0 0 1 0 0 0
    0 0 0 1 0 0
    0 0 0 -1 0 0
    0 0 0 0 1 0
    0 0 0 0 -1 0
    0 0 0 0 0 1
    0 0 0 0 0 -1];

bw=[r*ones(6,1); zeros(6,1)];

% Create polyhedrons
doubleX=Polyhedron(Ax,bx);
doubleU=Polyhedron(Au,bu);
scriptW=Polyhedron(Aw,bw);




%% LQR parameters
% Values initially chosen. Tuned to:
Q=10e5*eye(6);                                                      % State weights
R=100*eye(3);                                                       % Actuation weights
[Klqr Plqr e]=dlqr(A,B,Q,R);                  % determine LQR gain and sol. to Riccati
equation
Klqr=-Klqr;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of the disturbance rejection gain
display(sprintf('\nDetermining K_dr CL dist rejection gain....\n'))

% Set constants
a=zeros(6,1);                                          % zeros vector
wVert=scriptW.V;                                       % vertices of disturbance bound
l=(1/(65/1500))*Au;                                    % normalized input
h=[(1/100)*Ax(1:6,:); Ax(7:12,:)];                     % and state constraint inequalities

    lamda=1e-5;
    k=0;
for p=0.05:0.01:0.99                                % iteratively conduct optimization
    p
    k=k+1;
cvx_begin sdp quiet
    variable Y(3,6)
    variable W(6,6) symmetric
    variable Gamma(1,1) banded(0,1)



    minimize (Gamma)
    subject to
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(1,:);(Ac*W+Bc*Y),wVert(1,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(2,:);(Ac*W+Bc*Y),wVert(2,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(3,:);(Ac*W+Bc*Y),wVert(3,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(4,:);(Ac*W+Bc*Y),wVert(4,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(5,:);(Ac*W+Bc*Y),wVert(5,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(6,:);(Ac*W+Bc*Y),wVert(6,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(7,:);(Ac*W+Bc*Y),wVert(7,:)',W]>=0
        [lamda*W,a,(Ac*W+Bc*Y)';a',1-lamda,wVert(8,:);(Ac*W+Bc*Y),wVert(8,:)',W]>=0
        [p^2,(Y'*l(1,:)')';(Y'*l(1,:)'),W]>=0
        [p^2,(Y'*l(2,:)')';(Y'*l(2,:)'),W]>=0
        [p^2,(Y'*l(3,:)')';(Y'*l(3,:)'),W]>=0
        [p^2,(Y'*l(4,:)')';(Y'*l(4,:)'),W]>=0
        [p^2,(Y'*l(5,:)')';(Y'*l(5,:)'),W]>=0
        [p^2,(Y'*l(6,:)')';(Y'*l(6,:)'),W]>=0
        [Gamma,(W*h(1,:)')';(W*h(1,:)'),W]>=0
        [Gamma,(W*h(2,:)')';(W*h(2,:)'),W]>=0
        [Gamma,(W*h(3,:)')';(W*h(3,:)'),W]>=0
        [Gamma,(W*h(4,:)')';(W*h(4,:)'),W]>=0
        [Gamma,(W*h(5,:)')';(W*h(5,:)'),W]>=0
        [Gamma,(W*h(6,:)')';(W*h(6,:)'),W]>=0
cvx_end

gamma1(k,:)=[p];
Kdr=Y*inv(W);
```

```
Kdr
end
```

C. Satellite Rendezvous simulator

The controller for the satellite rendezvous maneuver was implemented in Simulink using the MPC controller block from the Model Predictive Control toolbox and several custom function blocks. Recall from *sections 8.3.5* and *8.3.6* that several simulations were constructed for testing the limitations and characteristics of the controller. However, there are only a couple of basis controllers. The simulators will be sorted accordingly. First, the simulator used to obtain the results in *sections 8.3.1* to *8.3.3* as presented in *section 8.1.*

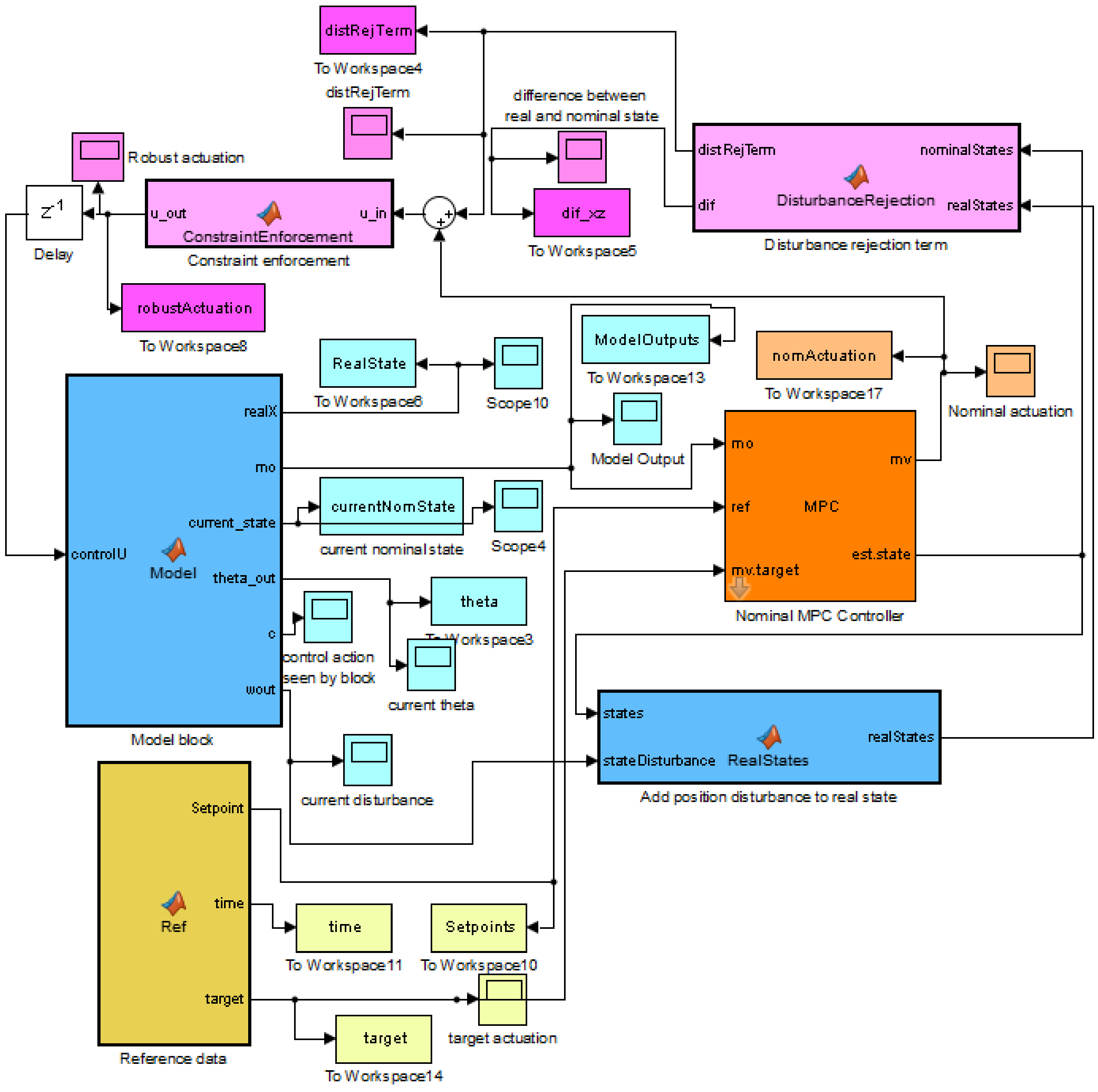


Figure Appendix - 1 Simulink simulator for Tube-base robust MPC of satellite rendezvous

The optional settings activated in the Model Predictive Control toolbox block labelled Nominal MPC controller (orange) are the estimated states output and actuation target inputs. The controller and initial controller state set in `SatRend_controller_MPC_toolbox_initscript` above are selected for use here. The code used to operate each block is given in the following. First, the Model block is presented.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL BLOCK: Simulation state and output estimation                              %
% -------------------------------------------------------------------------------- %
% This function handles state initialization and updates for the simulation.       %
% -------------------------------------------------------------------------------- %
% [realX,mo,current_state,theta_out,c,wout]  = fcn(controlU)                       %
%                                                                                  %
% Block input:                                                                     %
%       controlU : robust control action, input u to the state space model         %
%                                                                                  %
% Block outputs:                                                                   %
%       realX :  current real state of the system                                  %
%       mo    :  current nominal state of the system                               %
%       wout  :  current position disturbance                                      %
%                                                                                  %
% -------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver  %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,     %
%          Technische Universität München, 2016.                                   %
%                                                                                  %
% Copyright: Caroline Buckner                                                      %
% Last edited: 22 Dec 2016                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [realX,mo,current_state,theta_out,c,wout]  = Model(controlU)
%#codegen

%% Set system variables initial conditions for the states and system model

persistent time                                         % Time-step counter (=0.5[s])
if isempty(time)                                        % If not set,1
    time=1;                                             % then set time to first time step.
end

persistent dx                                      % Relative position: radial [m]
if isempty(dx)                                     % If not set,
    dx=39;                                         % set to initial radial position state.
end
persistent dy                                      % Relative position: along-track [m]
if isempty(dy)                                     % If not set,
    dy=39;                                         % set to initial along-track
end                                                % position state.
persistent dz                                      % Relative position: cross-track [m]
if isempty(dz)                                     % If not set,
    dz=4.23;                                       % set to initial cross-track
end                                                % position state.
persistent ddx                                     % Relative velocity: radial [m/s]
if isempty(ddx)                                    % If not set,
    ddx=0;                                         % set to initial radial velocity state.
end
persistent ddy                                     % Relative velocity: along-track [m/s]
if isempty(ddy)                                    % If not set,
    ddy=0;                                         % set to initial along-track
end                                                % position state.
persistent ddz                                     % Relative velocity: cross-track [m/s]
if isempty(ddz)                                    % If not set,
    ddz=0;                                         % set to initial cross-track
end                                                % position state.
persistent w                                       % Initialize disturbance
```

```
if isempty(w)
   w=zeros(12,1);
end
persistent theta                                % Initialize theta state
if isempty(theta)
    theta=zeros(6,1);
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];
    z_s1=[1.97;0;4.23;0;0;0;0;0;0];
    theta=Mtheta'*z_s1;
end
persistent next_state                               % Initialize persistent variable to
if isempty(next_state)                              % carry over state to next time step
    next_state=zeros(1,12);
    next_state=[dx dy dz ddx ddy ddz theta'];
end

persistent data                                 % Initialize variable to hold reference
if isempty(data)                                % Only load reference data if it has not

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');                                         % yet been loaded.
end

if isempty(controlU)                            % Set initial input value
    controlU=[data(1,8),data(1,9),data(1,10)]';
end


%% Actuation constraint assurance
if controlU(1)>0.043333335388468
    controlU(1)=0.043333335388468;
elseif controlU(1)<-0.043333335388552
    controlU(1)=-0.043333335388552;
else
    controlU(1)=controlU(1);
end

if controlU(2)>0.043333333452083
    controlU(2)=0.043333333452083;
elseif controlU(2)<-0.043333333449083
    controlU(2)=-0.043333333449083;
else
    controlU(2)=controlU(2);
end

if controlU(3)>0.043333333333334
    controlU(3)=0.043333333333334;
elseif controlU(3)<-0.043333333333334
    controlU(3)=-0.043333333333334;
else
    controlU(3)=controlU(3);
end

```

```
%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];

%% Set current states
current_state=next_state;
realX=current_state(1:6)'+w(1:6);

%% State Update

% % CTS model
% % norbit=0.0012;
% A=[0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3*norbit^2 0 0 0 2*norbit 0;
       0 0 0 -2*norbit 0 0; 0 0 -norbit^2 0 0 0];
% B=[0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
% C=eye(6);
% D=zeros(6,3);
% % sys=ss(A,B,C,D,0.5);
% % [A,B,C,D]=ssdata(sys);

% %  DIS Model
% %  Sample time  0.5 s
% State matrix
% A=[ 1 0 0 0.5 0.0003 0; -2.16e-10 1       0 -0.0003    0.5   0; 0 0 1 0 0 0.5;
%      2.16e-6 0 0 1 0.0012 0; -1.296e-9 0 0 -0.0012 1 0; 0 0 -7.2e-7 0 0 1];
% Input matrix
% B=[ 0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5];
% Output matrix
% C=eye(6);
% Feedthrough matrix
% D=zeros(6,3);

% Extend to incorporate terminal theta cost, extra states only processed in
% cost function in final the offset cost term.
% State matrix
A=[1         0 0       0.5     0.0003 0   0 0 0 0 0 0;
   -2.16e-10 1 0       -0.0003 0.5    0   0 0 0 0 0 0;
   0         0 1       0       0      0.5 0 0 0 0 0 0;
   2.16e-6   0 0       1       0.0012 0   0 0 0 0 0 0;
   -1.296e-9 0 0       -0.0012 1      0   0 0 0 0 0 0;
   0         0 -2.6e-6 0       0      1   0 0 0 0 0 0
   0         0 0       0       0      0   1 0 0 0 0 0
   0         0 0       0       0      0   0 1 0 0 0 0
   0         0 0       0       0      0   0 0 1 0 0 0
   0         0 0       0       0      0   0 0 0 1 0 0
   0         0 0       0       0      0   0 0 0 0 1 0
   0         0 0       0       0      0   0 0 0 0 0 1];
```

```
% Input matrix
B=[0.125   5e-5   0;
   -5e-5   0.125  0;
    0      0      0.125;
    0.5    0.0003 0;
   -0.0003 0.5    0;
    0      0      0.5
    0  0  0
    0  0  0
    0  0  0
    0  0  0
    0  0  0
    0  0  0];
% Output matrix
C=[eye(12)];
% Feedthrough matrix
D=zeros(12,3);


% Calculate next sate and output
Xplus=A*current_state'+B*controlU;
Y=C*current_state'+D*controlU;
dx=Xplus(1);
dy=Xplus(2);
dz=Xplus(3);
ddx=Xplus(4);
ddy=Xplus(5);
ddz=Xplus(6);
theta=Mtheta'*([Xplus(1);Xplus(2);Xplus(3);Xplus(4);Xplus(5);Xplus(6);controlU(1);control
U(2);controlU(3)]'+w(1:9)')';
theta_out=theta;

next_state=[Xplus(1:6)' theta'];
% Set nominal state output
mo=next_state';


% Apply tube constraints and calculate next position disturbance
% dist=sqrt(w(1)^2+w(2)^2+w(3)^2);
if sqrt((realX(1)-current_state(1))^2)<=4.6662&&sqrt((realX(2)-
current_state(2))^2)<=4.6662&&sqrt((realX(3)-current_state(3))^2)<=4.6662

% % % Final state (0,0,4.6662,0,0,0)
%    w(1)=(-1.97/(600*2+1))*time;
%    w(2)=(0-0/(600*2+1))*time;
%    w(3)=((4.6662-4.23)/(600*2+1))*time;

% % % Final state (-4.6662,0,0,0,0,0)
%     w(1)=((-4.6662-1.97)/(600*2+1))*time;
%     w(2)=((0-0)/(600*2+1))*time;
%     w(3)=((0-4.23)/(600*2+1))*time;

% % % Final state (1,1,4.446,0,0,0)
%    w(1)=((1-1.97)/(600*2+1))*time;
%    w(2)=((1-0)/(600*2+1))*time;
%    w(3)=((4.446-4.23)/(600*2+1))*time;

% % % Final state (1,2,4.095536893,0,0,0)
%    w(1)=((1-1.97)/(600*2+1))*time;
%    w(2)=((2-0)/(600*2+1))*time;
%    w(3)=((4.095536893-4.23)/(600*2+1))*time;

% % % Final state (-3,-2,-2.961996361,0,0,0)
%    w(1)=((-3-1.97)/(600*2+1))*time;
%    w(2)=((-3-0)/(600*2+1))*time;
%    w(3)=((-2.961996361-4.23)/(600*2+1))*time;

% % % Final state (0,-4.6662,0,0,0,0)
```

```
   w(1)=((0-1.97)/(600*2+1))*time;
   w(2)=((-4.6662-0)/(600*2+1))*time;
   w(3)=((0-4.23)/(600*2+1))*time;

% % % Final state (-3,-2,-2.961996361,0,0,0)
%    w(1)=((-1.97-1.97)/(600*2+1))*time;
%    w(2)=((0)/(600*2+1))*time;
%    w(3)=((-4.23-4.23)/(600*2+1))*time;

% % % Final state (5,5,5,0,0,0)
%    w(1)=((-10-1.97)/(600*2+1))*time;
%    w(2)=((-10-0)/(600*2+1))*time;
%    w(3)=((-10-4.23)/(600*2+1))*time;

% % % Final state (-2, 4.6662, 0, 0, 0, 0)
%    w(1)=((-2-1.97)/(600*2+1))*time;
%    w(2)=((4.6662-0)/(600*2+1))*time;
%    w(3)=((0-4.23)/(600*2+1))*time;

% % % Final state (-1.97, 0, -4.23, 0, 0, 0)
%    w(1)=((-1.97-1.97)/(600*2+1))*time;
%    w(2)=((0-0)/(600*2+1))*time;
%    w(3)=((-4.23-4.23)/(600*2+1))*time;
% w(1:3)=[0;0;0];
else
    w=w;
end

% Apply position disturbance to real state for visualization
realX=next_state(1:6)'+w(1:6);

% Set outputs
c=controlU;
wout=w(1:6);
% Update time
time=time+1;
end

```

The state disturbance is set by uncommenting or adding the applicable values. Next the function sending the reference data to the controller is given.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REFERENCE BLOCK: Simulation reference vectors                                        %
% ------------------------------------------------------------------------------------ %
% This function parses the reference data to send the appropriately sized series of    %
% vectors to the reference and target ports of the controller block.                   %
% ------------------------------------------------------------------------------------ %
% [Setpoint, time, target]  = Ref                                                      %
%                                                                                      %
% Block input:                                                                         %
%       ------                                                                         %
%                                                                                      %
% Block outputs:                                                                       %
%       Setpoint :  set of vectors containing reference state values for each          %
%                          prediction step                                             %
%       time     :  current time step, currently used for error checking only          %
%       target   :  reference for current actuation                                    %
%                                                                                      %
% ------------------------------------------------------------------------------------ %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver      %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,         %
%          Technische Universität München, 2016.                                        %
%                                                                                      %
% Copyright: Caroline Buckner                                                          %
% Last edited: 22 Dec 2016                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Setpoint, time, target] = Ref
%#codegen

```

```

%% Set persistent and data
persistent time_step;                                    % Current time step [0.5 s]
if isempty(time_step)
    time_step=1;
end
persistent data                                 % Load reference data,
if isempty(data)                                % but only if it hasn't been loaded yet

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
              0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
              0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
             -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
              0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
              0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
             -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
              0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
              0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
              -0.038964497175050, 0.705290934311349];


%% Initialize vectors
Setpoint=zeros(40,12);
Setpoints_3D=data(:,2:7);


%% Set the series of reference vectors. Prediction horizon = 40 vectors are set from the
reference data for each time step

if time_step+39<=1340
    thets=Mtheta'*data(time_step:time_step+39,2:10)';
    Setpoint=[Setpoints_3D(time_step:time_step+39,:),thets'];
elseif time_step==1341
    thets=Mtheta'*data(1341,2:10)';
    Setpoint=[repmat(Setpoints_3D(1341,:),40,1),repmat(thets',40,1)];
else
end

%% Set the actuation target for the current prediction from reference data

Ux=data(time_step,8);
Uy=data(time_step,9);
Uz=data(time_step,10);
target=[Ux;Uy;Uz];

% Increment time
time=time_step;
time_step=time_step+1;

```

Next the function applying the position disturbance to the real system is given.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POSITION DISTURBANCE BLOCK                                                         %
% ---------------------------------------------------------------------------------- %
% Adds the position disturbance to obtain the real state of the system.             %
% ---------------------------------------------------------------------------------- %
% realStates = RealStates(states,stateDisturbance)                                   %
%                                                                                    %
% Block input:                                                                       %
%       states              : real state before position disturbance added           %
%       stateDisturbance    : current state disturbance                              %
%                                                                                    %
% Block outputs:                                                                     %
%       realStates :  state of the disturbed system                                  %
%                                                                                    %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                     %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function realStates = RealStates(states,stateDisturbance)
realStates=states(1:6)+stateDisturbance;
end
```

The disturbance rejection term is determined using the following function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DISTURBANCE REJECTION BLOCK                                                        %
% ---------------------------------------------------------------------------------- %
% Determine the disturbance rejection term from the disturbance rejection gain and  %
% the difference between the real state and the nominal state.                       %
% ---------------------------------------------------------------------------------- %
% function [distRejTerm, dif] = DisturbanceRejection(nominalStates,realStates)      %
%                                                                                    %
% Block input:                                                                       %
%       nominalStates    : nominal state                                             %
%       realStates       : real state                                                %
%                                                                                    %
% Block outputs:                                                                     %
%       distRejTerm :  additional actuation                                          %
%       dif         :  difference between the real and nominal state                 %
%                                                                                    %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                      %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [distRejTerm, dif] = DisturbanceRejection(nominalStates,realStates)
%% Set internal variables
z=nominalStates(1:6);
x=realStates(1:6);
 %% Set disturbance rejection gain (refer to: section 7.2)
Kdr=[-0.00000432 0 0 0 -0.002399927735347 0;
       0 0 0 0.002399927789568 0 0;
       0 0 -0.00000432 0 0 0];
%% Calculated outputs
distRejTerm=Kdr*(x-z);
dif=(x-z);
end
```

The final block is a redundancy to ensure the robust input constraint is enforced.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constraint enforcement                                                            %
% --------------------------------------------------------------------------------- %
% Error catching redundancy.                                                        %
% --------------------------------------------------------------------------------- %
% function u_out = ConstraintEnforcement(u_in)                                      %
%                                                                                   %
% Block input:                                                                      %
%       u_in   : calculated actuation                                               %
%                                                                                   %
% Block outputs:                                                                    %
%       u_out :  constraint enforced actuation                                      %
%                                                                                   %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%         of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%         Technische Universität München, 2016.                                     %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function u_out = ConstraintEnforcement(u_in)

u_out=u_in;

if u_in(1)>0.043333335388468
    u_out(1)=0.043333335388468;
elseif u_in(1)<-0.043333335388552
    u_out(1)=-0.043333335388552;
else
    u_out(1)=u_in(1);
end

if u_in(2)>0.043333333452083
    u_out(2)=0.043333333452083;
elseif u_in(2)<-0.043333333449083
    u_out(2)=-0.043333333449083;
else
    u_out(2)=u_in(2);
end

if u_in(3)>0.043333333333334
    u_out(3)=0.043333333333334;
elseif u_in(3)<-0.043333333333334
    u_out(3)=-0.043333333333334;
else
    u_out(3)=u_in(3);
end
end
```

The Delay block steps the algebraic loop forward in time. The remainder of the blocks are either scopes or variable outputs to the MATLAB workspace for further processing. In the simulation used to stress the actuation constraints, only the Model block and Reference block are very slightly modified to increase the step through the reference data and the output variable names are modified to make them unique in the MATLAB workspace. As these changes are so minor, only the code for the new Reference and Model blocks are given here, grouped by step-size change.

For 300 s rendezvous time:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL BLOCK: Simulation state and output estimation for 300 s rendezvous time     %
% --------------------------------------------------------------------------------- %
% This function handles state initialization and updates for the simulation.       %
% --------------------------------------------------------------------------------- %
% [realX,mo,current_state,theta_out,c,wout]  = fcn(controlU)                         %
%                                                                                    %
% Block input:                                                                       %
%       controlU : robust control action, input u to the state space model          %
%                                                                                    %
% Block outputs:                                                                     %
%       realX :  current real state of the system                                   %
%       mo    :  current nominal state of the system                                %
%       wout  :  current position disturbance                                       %
%                                                                                    %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [realX,mo,current_state,theta_out,c,wout]  = Model(controlU)
%#codegen

%% Set system variables initial conditions for the states and system model
persistent time                                      % Time-step counter (=0.5[s])
if isempty(time)                                     % If not set,
    time=1;                                          % then set time to first time step.
end

persistent dx                                   % Relative position: radial [m]
if isempty(dx)                                  % If not set,
    dx=39;                                      % set to initial radial position state.
end
persistent dy                                   % Relative position: along-track [m]
if isempty(dy)                                  % If not set,
    dy=39;                                      % set to initial along-track
end                                             % position state.
persistent dz                                   % Relative position: cross-track [m]
if isempty(dz)                                  % If not set,
    dz=4.23;                                    % set to initial cross-track
end                                             % position state.
persistent ddx                                  % Relative velocity: radial [m/s]
if isempty(ddx)                                 % If not set,
    ddx=0;                                      % set to initial radial velocity state.
end
persistent ddy                                  % Relative velocity: along-track [m/s]
if isempty(ddy)                                 % If not set,
    ddy=0;                                       % set to initial along-track
end                                                % position state.
persistent ddz                                     % Relative velocity: cross-track [m/s]
if isempty(ddz)                                    % If not set,
    ddz=0;                                         % set to initial cross-track
end                                                % position state.
persistent w                                       % Initialize disturbance
if isempty(w)
   w=zeros(12,1);
end
persistent theta                                   % Initialize theta state
if isempty(theta)
    theta=zeros(6,1);
    Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];
    z_s1=[1.97;0;4.23;0;0;0;0;0;0];
    theta=Mtheta'*z_s1;
end
persistent next_state                               % Initialize persistent variable to
if isempty(next_state)                              % carry over state to next time step
    next_state=zeros(1,12);
    next_state=[dx dy dz ddx ddy ddz theta'];
end

persistent data1
persistent data                                  % Initialize variable to hold reference
if isempty(data)                                 % Only load reference data if it has not

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');                                          % yet been loaded.
end
if isempty(data1)                                % Scale reference data to 300 s
     data1=data(1:2:1201,:);
end
%% Actuation constraint assurance
if controlU(1)>0.043333335388468
    controlU(1)=0.043333335388468;
elseif controlU(1)<-0.043333335388552
    controlU(1)=-0.043333335388552;
else
    controlU(1)=controlU(1);
end

if controlU(2)>0.043333333452083
    controlU(2)=0.043333333452083;
elseif controlU(2)<-0.043333333449083
    controlU(2)=-0.043333333449083;
else
    controlU(2)=controlU(2);
end

if controlU(3)>0.043333333333334
    controlU(3)=0.043333333333334;
elseif controlU(3)<-0.043333333333334
```

```
    controlU(3)=-0.043333333333334;
else
    controlU(3)=controlU(3);
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];


%% Set current states
current_state=next_state;
realX=current_state(1:6)'+w(1:6);


%% State Update
% % CTS model
% % norbit=0.0012;
% A=[0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3*norbit^2 0 0 0 2*norbit 0;
%     0 0 0 -2*norbit 0 0; 0 0 -norbit^2 0 0 0];
% B=[0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
% C=eye(6);
% D=zeros(6,3);
%
% % sys=ss(A,B,C,D,0.5);
% % [A,B,C,D]=ssdata(sys);

% %  DIS Model
% %  Sample time  0.5 s
% State matrix
% A=[1 0 0 0.5 0.0003 0; -2.16e-10 1 0 -0.0003 0.5 0; 0 0 1 0 0 0.5;
%     2.16e-6 0 0 1 0.0012 0; -1.296e-9 0 0 -0.0012 1 0; 0 0 -7.2e-7 0 0 1];
% Input matrix
% B=[ 0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5];
% Output matrix
% C=eye(6);
% Feedthrough matrix
% D=zeros(6,3);

% Extend to incorporate terminal theta cost, extra states only processed in
% cost function in final the offset cost term.
% State matrix
A=[1 0 0 0.5 0.0003 0 0 0 0 0 0 0; -2.16e-10 1 0 -0.0003 0.5 0 0 0 0 0 0 0;
   0 0 1 0 0 0.5 0 0 0 0 0 0; 2.16e-6 0 0 1 0.0012 0 0 0 0 0 0 0;
   -1.296e-9 0 0 -0.0012 1 0 0 0 0 0 0 0; 0 0 -2.6e-6 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0; 0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 0 0 0 0 1];
% Input matrix
B=[0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5; 0 0 0;
    0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];
% Output matrix
C=[eye(12)];
% Feedthrough matrix
```

```
D=zeros(12,3);


% Calculate next sate and output
Xplus=A*current_state'+B*controlU;
Y=C*current_state'+D*controlU;
dx=Xplus(1);
dy=Xplus(2);
dz=Xplus(3);
ddx=Xplus(4);
ddy=Xplus(5);
ddz=Xplus(6);
theta=Mtheta'*([Xplus(1);Xplus(2);Xplus(3);Xplus(4);Xplus(5);Xplus(6);controlU(1);control
U(2);controlU(3)]'+w(1:9)')';
theta_out=theta;

next_state=[Xplus(1:6)' theta'];
% Set nominal state output
mo=next_state';


%% Apply tube constraints and calculate next position disturbance
if sqrt((realX(1)-current_state(1))^2)<=4.6662&&sqrt((realX(2)-
current_state(2))^2)<=4.6662&&sqrt((realX(3)-current_state(3))^2)<=4.6662
% % % Final state (0,0,4.6662,0,0,0)
%    w(1)=(-1.97/(600))*time;
%    w(2)=(0-0/(600))*time;
%    w(3)=((4.6662-4.23)/(600))*time;
% % % %

% % % Final state (1,1,4.446,0,0,0)
%    w(1)=((1-1.97)/600)*time;
%    w(2)=((1-0)/600)*time;
%    w(3)=((4.446-4.23)/600)*time;
% % % %
% % % Final state (0,-4.6662,0,0,0,0)
   w(1)=((0-1.97)/600)*time;
   w(2)=((-4.6662-0)/600)*time;
   w(3)=((0-4.23)/600)*time;
% % % %
%    w(1)=((-1.97-1.97)/(601))*time;
%    w(2)=((0-0)/(601))*time;
%    w(3)=((-4.23-4.23)/(601))*time;

% % % % w(1:3)=[0;0;0];
else
    w=w;
end

% Apply position disturbance to real state for visulization
realX=next_state(1:6)'+w(1:6);

%% Set outputs
c=controlU;
wout=w(1:6);
% Update time
time=time+1;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REFERENCE BLOCK: Simulation reference vectors for 300 s rendezvous time            %
% ---------------------------------------------------------------------------------- %
% This function parses the reference data to send the appropriately sized series of  %
% vectors to the reference and target ports of the controller block.                 %
% ---------------------------------------------------------------------------------- %
% [Setpoint, time, target]  = Ref                                                    %
%                                                                                    %
% Block input:                                                                       %
%       ------                                                                       %
%                                                                                    %
% Block outputs:                                                                     %
%       Setpoint :  set of vectors containing reference state values for each        %
%                            prediction step                                         %
%       time     :  current time step, currently used for error checking only        %
%       target   :  reference for current actuation                                  %
%                                                                                    %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%           of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%           Technishe Universität München, 2016.                                     %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Setpoint, time, target] = Ref
%#codegen

%% Set persistent and data
persistent time_step;                                      % Current time step [0.5 s]
if isempty(time_step)
    time_step=1;
end
persistent data
persistent refdata                              % Load reference data,
if isempty(data)                                % but only if it hasn't been loaded yet

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');
    refdata=zeros(700,10);
    k=1;                                                    % Scale reference to 300 s
    for j=1:2:1201
        refdata(k,:)=data(j,1:10);
        k=k+1;
    end
    for k=601:1:700
        refdata(k,:)=data(1201,1:10);
    end
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];
```

```
%% Initialize vectors
Setpoint=zeros(40,12);


%% Set the series of reference vectors. Prediction horizon = 40 vectors are set from the
reference data for each time step

if time_step+39<=650
    thets=Mtheta'*refdata(time_step:time_step+39,2:10)';
    Setpoint=[refdata(time_step:time_step+39,2:7),thets'];

elseif time_step>=602
    thets=Mtheta'*refdata(602,2:10)';
    Setpoint=[repmat(refdata(602,2:7),40,1),repmat(thets',40,1)];
else
end

%% Set the actuation target for the current prediction from reference data
Ux=refdata(time_step,8);
Uy=refdata(time_step,9);
Uz=refdata(time_step,10);
target=[Ux;Uy;Uz];

%% Increment time
time=time_step;
time_step=time_step+1;
```

For a rendezvous time of 150 s:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL BLOCK: Simulation state and output estimation for 150 s rendezvous          %
% --------------------------------------------------------------------------------- %
% This function handles state initialization and updates for the simulation.        %
% --------------------------------------------------------------------------------- %
% [realX,mo,current_state,theta_out,c,wout]  = fcn(controlU)                        %
%                                                                                   %
% Block input:                                                                      %
%       controlU : robust control action, input u to the state space model          %
%                                                                                   %
% Block outputs:                                                                    %
%       realX :  current real state of the system                                   %
%       mo    :  current nominal state of the system                                %
%       wout  :  current position disturbance                                       %
%                                                                                   %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 function [realX,mo,current_state,theta_out,c,wout]  = Model(controlU)
%#codegen

%% Set system variables initial conditions for the states and system model

persistent time                                         % Time-step counter (=0.5[s])
if isempty(time)                                        % If not set,1
    time=1;                                             % then set time to first time step.
end

persistent dx                                       % Relative position: radial [m]
if isempty(dx)                                      % If not set,
    dx=39;                                          % set to initial radial position state.
end
persistent dy                                       % Relative position: along-track [m]
```

```
if isempty(dy)                                    % If not set,
    dy=39;                                        % set to initial along-track
end                                               % position state.
persistent dz                                     % Relative position: cross-track [m]
if isempty(dz)                                    % If not set,
    dz=4.23;                                      % set to initial cross-track
end                                               % position state.
persistent ddx                                    % Relative velocity: radial [m/s]
if isempty(ddx)                                   % If not set,
    ddx=0;                                        % set to initial radial velocity state.
end
persistent ddy                                    % Relative velocity: along-track [m/s]
if isempty(ddy)                                   % If not set,
    ddy=0;                                         % set to initial along-track
end                                                % position state.
persistent ddz                                     % Relative velocity: cross-track [m/s]
if isempty(ddz)                                    % If not set,
    ddz=0;                                         % set to initial cross-track
end                                                % position state.
persistent w                                       % Initialize disturbance
if isempty(w)
   w=zeros(12,1);
end
persistent theta                                   % Initialize theta state
if isempty(theta)
    theta=zeros(6,1);
    Mtheta=[-0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];
    z_s1=[1.97;0;4.23;0;0;0;0;0;0];
    theta=Mtheta'*z_s1;
end
persistent next_state                            % Initialize persistent variable to
if isempty(next_state)                           % carry over state to next time step
    next_state=zeros(1,12);
    next_state=[dx dy dz ddx ddy ddz theta'];
end

persistent data1
persistent data                                  % Initialize variable to hold reference
if isempty(data)                                 % Only load reference data if it has not

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');                                          % yet been loaded.
end
if isempty(data1)                                % Scale reference data to 300 s
     data1=data(1:4:1201,:);
end

%% Actuation constraint assurance
if controlU(1)>0.043333335388468
    controlU(1)=0.043333335388468;
elseif controlU(1)<-0.043333335388552
    controlU(1)=-0.043333335388552;
else
```

```
    controlU(1)=controlU(1);
end

if controlU(2)>0.043333333452083
    controlU(2)=0.043333333452083;
elseif controlU(2)<-0.043333333449083
    controlU(2)=-0.043333333449083;
else
    controlU(2)=controlU(2);
end

if controlU(3)>0.043333333333334
    controlU(3)=0.043333333333334;
elseif controlU(3)<-0.043333333333334
    controlU(3)=-0.043333333333334;
else
    controlU(3)=controlU(3);
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
 Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];

%% Set current states
current_state=next_state;
realX=current_state(1:6)'+w(1:6);


%% State Update
% % CTS model
% % norbit=0.0012;
% A=[0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3*norbit^2 0 0 0 2*norbit 0;
%     0 0 0 -2*norbit 0 0; 0 0 -norbit^2 0 0 0];
% B=[0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
% C=eye(6);
% D=zeros(6,3);
%
% % sys=ss(A,B,C,D,0.5);
% % [A,B,C,D]=ssdata(sys);

% %  DIS Model
% %  Sample time  0.5 s
% State matrix
% A=[1 0 0 0.5 0.0003 0; -2.16e-10 1 0 -0.0003 0.5 0; 0 0 1 0 0 0.5;
%     2.16e-6 0 0 1 0.0012 0; -1.296e-9 0 0 -0.0012 1 0; 0 0 -7.2e-7 0 0 1];
% Input matrix
% B=[ 0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5];
% Output matrix
% C=eye(6);
% Feedthrough matrix
% D=zeros(6,3);

% Extend to incorporate terminal theta cost, extra states only processed in
% cost function in final the offset cost term.
% State matrix
A=[1 0 0 0.5 0.0003 0 0 0 0 0 0 0; -2.16e-10 1 0 -0.0003 0.5 0 0 0 0 0 0 0;
   0 0 1 0 0 0.5 0 0 0 0 0 0; 2.16e-6 0 0 1 0.0012 0 0 0 0 0 0 0;
   -1.296e-9 0 0 -0.0012 1 0 0 0 0 0 0 0; 0 0 -2.6e-6 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0; 0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 0 0 0 0 1];
% Input matrix
B=[0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5; 0 0 0;
    0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];
% Output matrix
C=[eye(12)];
% Feedthrough matrix
D=zeros(12,3);
% Calculate next sate and output
Xplus=A*current_state'+B*controlU;
Y=C*current_state'+D*controlU;
dx=Xplus(1);
dy=Xplus(2);
dz=Xplus(3);
ddx=Xplus(4);
ddy=Xplus(5);
ddz=Xplus(6);
theta=Mtheta'*([Xplus(1);Xplus(2);Xplus(3);Xplus(4);Xplus(5);Xplus(6);controlU(1);control
U(2);controlU(3)]'+w(1:9)')';
theta_out=theta;

next_state=[Xplus(1:6)' theta'];
% Set nominal state output
mo=next_state';


%% Apply tube constraints and calculate next position disturbance
if sqrt((realX(1)-current_state(1))^2)<=4.6662&&sqrt((realX(2)-
current_state(2))^2)<=4.6662&&sqrt((realX(3)-current_state(3))^2)<=4.6662%dist<=4.6662
% % % % % % Final state (0,0,4.6662,0,0,0)
% % % % % % PLOTTED
%    w(1)=(-1.97/(300))*time;
%    w(2)=(0-0/(300))*time;
%    w(3)=((4.6662-4.23)/(300))*time;
% % %
% % % Final state (1,1,4.446,0,0,0)
% % % %    w(1)=((1-1.97)/(600*2+1))*time;
% % % %    w(2)=((1-0)/(600*2+1))*time;
% % % %    w(3)=((4.446-4.23)/(600*2+1))*time;
% % %
% % % Final state (0,-4.6662,0,0,0,0)
   w(1)=((0-1.97)/(300))*time;
   w(2)=((-4.6662-0)/(300))*time;
   w(3)=((0-4.23)/(300))*time;
% % % %
%    w(1)=((-1.97-1.97)/(300))*time;
%    w(2)=((0-0)/(300))*time;
%    w(3)=((-4.23-4.23)/(300))*time;

% % % % w(1:3)=[0;0;0];
else
    w=w;
end

% Apply position disturbance to real state for visualization
realX=next_state(1:6)'+w(1:6);

%% Set outputs
c=controlU;
wout=w(1:6);
% Update time
time=time+1;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REFERENCE BLOCK: Simulation reference vectors for 150 s rendezvous time           %
% --------------------------------------------------------------------------------- %
% This function parses the reference data to send the appropriately sized series of %
% vectors to the reference and target ports of the controller block.               %
% --------------------------------------------------------------------------------- %
% [Setpoint, time, target]  = Ref                                                   %
%                                                                                   %
% Block input:                                                                      %
%       ------                                                                      %
%                                                                                   %
% Block outputs:                                                                    %
%       Setpoint :  set of vectors containing reference state values for each       %
%                         prediction step                                           %
%       time     :  current time step, currently used for error checking only       %
%       target   :  reference for current actuation                                 %
%                                                                                   %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Setpoint, time, target] = Ref
%#codegen

%% Set persistent and data
persistent time_step;                                           % Current time step [0.5 s]
if isempty(time_step)
    time_step=1;
end
persistent data
persistent refdata                                  % Load reference data,
if isempty(data)                                    % but only if it hasn't been loaded yet

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');
    refdata=zeros(700,10);

    k=1;                                            % Scale reference to 150 s
    for j=1:4:1201
        refdata(k,:)=data(j,1:10);
        k=k+1;
    end
    for k=350:1:700
        refdata(k,:)=data(1201,1:10);
    end
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
            -0.038964497175050, 0.705290934311349];

%% Initialize vectors
Setpoint=zeros(40,12);

%% Set the series of reference vectors. Prediction horizon = 40 vectors are set from the
reference data for each time step

if time_step+39<=360
    thets=Mtheta'*refdata(time_step:time_step+39,2:10)';
    Setpoint=[refdata(time_step:time_step+39,2:7),thets'];

elseif time_step>=361
    thets=Mtheta'*refdata(361,2:10)';
    Setpoint=[repmat(refdata(361,2:7),40,1),repmat(thets',40,1)];
else
end

%% Set the actuation target for the current prediction from reference data

Ux=refdata(time_step,8);
Uy=refdata(time_step,9);
Uz=refdata(time_step,10);
target=[Ux;Uy;Uz];

%% Increment time
time=time_step;
time_step=time_step+1;
```

For a 75 s rendezvous:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL BLOCK: Simulation state and output estimation for 75 s rendezvous time     %
% -------------------------------------------------------------------------------- %
% This function handles state initialization and updates for the simulation.      %
% -------------------------------------------------------------------------------- %
% [realX,mo,current_state,theta_out,c,wout]  = fcn(controlU)                       %
%                                                                                  %
% Block input:                                                                     %
%       controlU : robust control action, input u to the state space model         %
%                                                                                  %
% Block outputs:                                                                   %
%       realX :  current real state of the system                                  %
%       mo    :  current nominal state of the system                               %
%       wout  :  current position disturbance                                      %
%                                                                                  %
% -------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver  %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,     %
%          Technische Universität München, 2016.                                   %
%                                                                                  %
% Copyright: Caroline Buckner                                                      %
% Last edited: 22 Dec 2016                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [realX,mo,current_state,theta_out,c,wout]  = Model(controlU)
%#codegen

%% Set system variables initial conditions for the states and system model

persistent time                                      % Time-step counter (=0.5[s])
if isempty(time)                                     % If not set,1
    time=1;                                          % then set time to first time step.
end

persistent dx                                    % Relative position: radial [m]
if isempty(dx)                                   % If not set,
    dx=39;                                        % set to initial radial position state.
end
persistent dy                                     % Relative position: along-track [m]
if isempty(dy)                                    % If not set,
    dy=39;                                        % set to initial along-track
end                                               % position state.
persistent dz                                     % Relative position: cross-track [m]
if isempty(dz)                                    % If not set,
    dz=4.23;                                      % set to initial cross-track
end                                               % position state.
persistent ddx                                    % Relative velocity: radial [m/s]
if isempty(ddx)                                   % If not set,
    ddx=0;                                        % set to initial radial velocity state.
end
persistent ddy                                   % Relative velocity: along-track [m/s]
if isempty(ddy)                                  % If not set,
    ddy=0;                                        % set to initial along-track
end                                               % position state.
persistent ddz                                    % Relative velocity: cross-track [m/s]
if isempty(ddz)                                   % If not set,
    ddz=0;                                        % set to initial cross-track
end                                               % position state.
persistent w                                      % Initialize disturbance
if isempty(w)
   w=zeros(12,1);
end
persistent theta                                  % Initialize theta state
if isempty(theta)
    theta=zeros(6,1);
    Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];

    z_s1=[1.97;0;4.23;0;0;0;0;0;0];
    theta=Mtheta'*z_s1;
end
persistent next_state                               % Initialize persistent variable to
if isempty(next_state)                              % carry over state to next time step
    next_state=zeros(1,12);
    next_state=[dx dy dz ddx ddy ddz theta'];
end

persistent data1
persistent data                                  % Initialize variable to hold reference
if isempty(data)                                 % Only load reference data if it has not

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');                                          % yet been loaded.
end
if isempty(data1)                                % Scale reference data to 300 s
     data1=data(1:8:1201,:);
end

%% Actuation constraint assurance
if controlU(1)>0.043333335388468
```

```
    controlU(1)=0.043333335388468;
elseif controlU(1)<-0.043333335388552
    controlU(1)=-0.043333335388552;
else
    controlU(1)=controlU(1);
end

if controlU(2)>0.043333333452083
    controlU(2)=0.043333333452083;
elseif controlU(2)<-0.043333333449083
    controlU(2)=-0.043333333449083;
else
    controlU(2)=controlU(2);
end

if controlU(3)>0.043333333333334
    controlU(3)=0.043333333333334;
elseif controlU(3)<-0.043333333333334
    controlU(3)=-0.043333333333334;
else
    controlU(3)=controlU(3);
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
 Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];

%% Set current states
current_state=next_state;
realX=current_state(1:6)'+w(1:6);


%% State Update
% % CTS model
% % norbit=0.0012;
% A=[0  0  0  1  0  0;  0  0  0  0  1  0;  0  0  0  0  0  1;  3*norbit^2  0  0  0  2*norbit  0;
%      0  0  0  -2*norbit  0  0;  0  0  -norbit^2  0  0  0];
% B=[0  0  0;  0  0  0;  0  0  0;  1  0  0;  0  1  0;  0  0  1];
% C=eye(6);
% D=zeros(6,3);
%
% % sys=ss(A,B,C,D,0.5);
% % [A,B,C,D]=ssdata(sys);

% %  DIS Model
% %  Sample time  0.5 s
% State matrix
% A=[1  0  0  0.5  0.0003  0;  -2.16e-10  1  0  -0.0003  0.5  0;  0  0  1  0  0  0.5;
%      2.16e-6  0  0  1  0.0012  0;  -1.296e-9  0  0  -0.0012  1  0;  0  0  -7.2e-7  0  0  1];
% Input matrix
% B=[ 0.125  5e-5  0;  -5e-5  0.125  0;  0  0  0.125;  0.5  0.0003  0;  -0.0003  0.5  0;  0  0  0.5];
% Output matrix
% C=eye(6);
% Feedthrough matrix
% D=zeros(6,3);

% Extend to incorporate terminal theta cost, extra states only processed in
% cost function in final the offset cost term.
% State matrix
A=[1 0 0 0.5 0.0003 0 0 0 0 0 0 0; -2.16e-10 1 0 -0.0003 0.5 0 0 0 0 0 0 0;
   0 0 1 0 0 0.5 0 0 0 0 0 0; 2.16e-6 0 0 1 0.0012 0 0 0 0 0 0 0;
   -1.296e-9 0 0 -0.0012 1 0 0 0 0 0 0 0; 0 0 -2.6e-6 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0; 0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 0 0 0 0 1];
% Input matrix
B=[0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5; 0 0 0;
    0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];
% Output matrix
C=[eye(12)];
% Feedthrough matrix
D=zeros(12,3);

% Calculate next sate and output
Xplus=A*current_state'+B*controlU;
Y=C*current_state'+D*controlU;
dx=Xplus(1);
dy=Xplus(2);
dz=Xplus(3);
ddx=Xplus(4);
ddy=Xplus(5);
ddz=Xplus(6);
theta=Mtheta'*([Xplus(1);Xplus(2);Xplus(3);Xplus(4);Xplus(5);Xplus(6);controlU(1);control
U(2);controlU(3)]'+w(1:9)')';
theta_out=theta;

next_state=[Xplus(1:6)' theta'];
% Set nominal state output
mo=next_state';


%% Apply tube constraints and calculate next position disturbance
if sqrt((realX(1)-current_state(1))^2)<=4.6662&&sqrt((realX(2)-
current_state(2))^2)<=4.6662&&sqrt((realX(3)-current_state(3))^2)<=4.6662
% % % % % % Final state (0,0,4.6662,0,0,0)
% % % % % % PLOTTED
%    w(1)=(-1.97/(151))*time;
%    w(2)=(0-0/(151))*time;
%    w(3)=((4.6662-4.23)/(151))*time;
% % %
% % % Final state (1,1,4.446,0,0,0)
% % % %    w(1)=((1-1.97)/(151))*time;
% % % %    w(2)=((1-0)/(151))*time;
% % % %    w(3)=((4.446-4.23)/(151))*time;
% % %
% % % Final state (0,-4.6662,0,0,0,0)
   w(1)=((0-1.97)/(151))*time;
   w(2)=((-4.6662-0)/(151))*time;
   w(3)=((0-4.23)/(151))*time;
% % % %
%    w(1)=((-1.97-1.97)/(151))*time;
%    w(2)=((0-0)/(151))*time;
%    w(3)=((-4.23-4.23)/(151))*time;

% % % % w(1:3)=[0;0;0];
else
    w=w;
end

% Apply position disturbance to real state for visualization
realX=next_state(1:6)'+w(1:6);

%% Set outputs
```

```
c=controlU;
wout=w(1:6);
% Update time
time=time+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REFERENCE BLOCK: Simulation reference vectors for 75 s rendezvous time             %
% ---------------------------------------------------------------------------------- %
% This function parses the reference data to send the appropriately sized series of %
% vectors to the reference and target ports of the controller block.                %
% ---------------------------------------------------------------------------------- %
% [Setpoint, time, target]  = Ref                                                    %
%                                                                                    %
% Block input:                                                                       %
%       ------                                                                       %
%                                                                                    %
% Block outputs:                                                                     %
%       Setpoint :  set of vectors containing reference state values for each        %
%                          prediction step                                           %
%       time     :  current time step, currently used for error checking only        %
%       target   :  reference for current actuation                                  %
%                                                                                    %
% ---------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver    %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,       %
%          Technische Universität München, 2016.                                      %
%                                                                                    %
% Copyright: Caroline Buckner                                                        %
% Last edited: 22 Dec 2016                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Setpoint, time, target] = Ref
%#codegen

%% Set persistent and data
persistent time_step;                                        % Current time step [0.5 s]
if isempty(time_step)
    time_step=1;
end
persistent data
persistent refdata                                  % Load reference data,
if isempty(data)                                    % but only if it hasn't been loaded yet

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');
    refdata=zeros(700,10);

    k=1;                                                      % Scale reference to 75 s
    for j=1:8:1201
        refdata(k,:)=data(j,1:10);
        k=k+1;
    end
    for k=350:1:700
        refdata(k,:)=data(1201,1:10);
    end
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];

%% Initialize vectors
Setpoint=zeros(40,12);

%% Set the series of reference vectors. Prediction horizon = 40 vectors are set from the
reference data for each time step

if time_step+39<=360
    thets=Mtheta'*refdata(time_step:time_step+39,2:10)';
    Setpoint=[refdata(time_step:time_step+39,2:7),thets'];

elseif time_step>=361
    thets=Mtheta'*refdata(361,2:10)';
    Setpoint=[repmat(refdata(361,2:7),40,1),repmat(thets',40,1)];
else
end

%% Set the actuation target for the current prediction from reference data
Ux=refdata(time_step,8);
Uy=refdata(time_step,9);
Uz=refdata(time_step,10);
target=[Ux;Uy;Uz];

%% Increment time
time=time_step;
time_step=time_step+1;
```

And finally for a rendezvous time of 37.5 s:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL BLOCK: Simulation state and output estimation for a rendezvous time of 37.5s%
% -------------------------------------------------------------------------------- %
% This function handles state initialization and updates for the simulation.       %
% -------------------------------------------------------------------------------- %
% [realX,mo,current_state,theta_out,c,wout]  = fcn(controlU)                        %
%                                                                                   %
% Block input:                                                                      %
%       controlU : robust control action, input u to the state space model          %
%                                                                                   %
% Block outputs:                                                                    %
%       realX :  current real state of the system                                   %
%       mo    :  current nominal state of the system                                %
%       wout  :  current position disturbance                                       %
%                                                                                   %
% -------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [realX,mo,current_state,theta_out,c,wout]  = Model(controlU)
%#codegen

%% Set system variables initial conditions for the states and system model

persistent time                                        % Time-step counter (=0.5[s])
if isempty(time)                                       % If not set,
    time=1;                                            % then set time to first time step.
end
```

```

persistent dx                                     % Relative position: radial [m]
if isempty(dx)                                    % If not set,
    dx=39;                                        % set to initial radial position state.
end
persistent dy                                     % Relative position: along-track [m]
if isempty(dy)                                    % If not set,
    dy=39;                                        % set to initial along-track
end                                               % position state.
persistent dz                                     % Relative position: cross-track [m]
if isempty(dz)                                    % If not set,
    dz=4.23;                                      % set to initial cross-track
end                                               % position state.
persistent ddx                                    % Relative velocity: radial [m/s]
if isempty(ddx)                                   % If not set,
    ddx=0;                                        % set to initial radial velocity state.
end
persistent ddy                                    % Relative velocity: along-track [m/s]
if isempty(ddy)                                   % If not set,
    ddy=0;                                         % set to initial along-track
end                                                % position state.
persistent ddz                                     % Relative velocity: cross-track [m/s]
if isempty(ddz)                                    % If not set,
    ddz=0;                                         % set to initial cross-track
end                                                % position state.
persistent w                                       % Initialize disturbance
if isempty(w)
   w=zeros(12,1);
end
persistent theta                                   % Initialize theta state
if isempty(theta)
    theta=zeros(6,1);
   Mtheta=[-0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];
    z_s1=[1.97;0;4.23;0;0;0;0;0;0];
    theta=Mtheta'*z_s1;
end
persistent next_state                             % Initialize persistent variable to
if isempty(next_state)                            % carry over state to next time step
    next_state=zeros(1,12);
    next_state=[dx dy dz ddx ddy ddz theta'];
end

persistent data1
persistent data                                  % Initialize variable to hold reference
if isempty(data)                                 % Only load reference data if it has not

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');                                          % yet been loaded.
end
if isempty(data1)                                % Scale reference data to 300 s
     data1=data(1:16:1201,:);
end

```

```
%% Actuation constraint assurance
if controlU(1)>0.043333335388468
    controlU(1)=0.043333335388468;
elseif controlU(1)<-0.043333335388552
    controlU(1)=-0.043333335388552;
else
    controlU(1)=controlU(1);
end

if controlU(2)>0.043333333452083
    controlU(2)=0.043333333452083;
elseif controlU(2)<-0.043333333449083
    controlU(2)=-0.043333333449083;
else
    controlU(2)=controlU(2);
end

if controlU(3)>0.043333333333334
    controlU(3)=0.043333333333334;
elseif controlU(3)<-0.043333333333334
    controlU(3)=-0.043333333333334;
else
    controlU(3)=controlU(3);
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
             0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
             0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
             -0.038964497175050, 0.705290934311349];


%% Set current states
current_state=next_state;
realX=current_state(1:6)'+w(1:6);


%% State Update
% % CTS model
% % norbit=0.0012;
% A=[0  0  0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3*norbit^2 0 0 0 2*norbit 0;
%      0  0  0 -2*norbit 0 0; 0 0 -norbit^2 0 0 0];
% B=[0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
% C=eye(6);
% D=zeros(6,3);
%
% % sys=ss(A,B,C,D,0.5);
% % [A,B,C,D]=ssdata(sys);

% %  DIS Model
% %  Sample time  0.5 s
% State matrix
% A=[1 0 0 0.5 0.0003 0; -2.16e-10 1 0 -0.0003 0.5 0; 0 0 1 0 0 0.5;
%      2.16e-6 0 0 1 0.0012 0; -1.296e-9 0 0 -0.0012 1 0; 0 0 -7.2e-7 0 0 1];
% Input matrix
% B=[ 0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5];
% Output matrix
% C=eye(6);
% Feedthrough matrix
% D=zeros(6,3);

% Extend to incorporate terminal theta cost, extra states only processed in
% cost function in final the offset cost term.
% State matrix
A=[1 0 0 0.5 0.0003 0 0 0 0 0 0 0; -2.16e-10 1 0 -0.0003 0.5 0 0 0 0 0 0 0;
   0 0 1 0 0 0.5 0 0 0 0 0 0; 2.16e-6 0 0 1 0.0012 0 0 0 0 0 0 0;
   -1.296e-9 0 0 -0.0012 1 0 0 0 0 0 0 0; 0 0 -2.6e-6 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0; 0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 0 0 0 0 1];
% Input matrix
B=[0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5; 0 0 0;
    0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];
% Output matrix
C=[eye(12)];
% Feedthrough matrix
D=zeros(12,3);

% Calculate next sate and output
Xplus=A*current_state'+B*controlU;
Y=C*current_state'+D*controlU;
dx=Xplus(1);
dy=Xplus(2);
dz=Xplus(3);
ddx=Xplus(4);
ddy=Xplus(5);
ddz=Xplus(6);
theta=Mtheta'*([Xplus(1);Xplus(2);Xplus(3);Xplus(4);Xplus(5);Xplus(6);controlU(1);control
U(2);controlU(3)]'+w(1:9)')';
theta_out=theta;

next_state=[Xplus(1:6)' theta'];
% Set nominal state output
mo=next_state';


%% Apply tube constraints and calculate next position disturbance
if sqrt((realX(1)-current_state(1))^2)<=4.6662&&sqrt((realX(2)-
current_state(2))^2)<=4.6662&&sqrt((realX(3)-current_state(3))^2)<=4.6662
% % % % % % Final state (0,0,4.6662,0,0,0)
% % % % % % PLOTTED
%    w(1)=(-1.97/(76))*time;
%    w(2)=(0-0/(76))*time;
%    w(3)=((4.6662-4.23)/(76))*time;
% % %
% % % Final state (1,1,4.446,0,0,0)
%    w(1)=((1-1.97)/(76))*time;
%    w(2)=((1-0)/(76))*time;
%    w(3)=((4.446-4.23)/(76))*time;
% % %
% % % Final state (0,-4.6662,0,0,0,0)
   w(1)=((0-1.97)/(76))*time;
   w(2)=((-4.6662-0)/(76))*time;
   w(3)=((0-4.23)/(76))*time;
% % % %
% %    w(1)=((-1.97-1.97)/(76))*time;
% %    w(2)=((0-0)/(76))*time;
% %    w(3)=((-4.23-4.23)/(76))*time;

% % % % w(1:3)=[0;0;0];
else
    w=w;
end

% Apply position disturbance to real state for visualization
```

```
realX=next_state(1:6)'+w(1:6);

%% Set outputs
c=controlU;
wout=w(1:6);
% Update time
time=time+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REFERENCE BLOCK: Simulation reference vectors for 37.5 s rendezvous time          %
% --------------------------------------------------------------------------------- %
% This function parses the reference data to send the appropriately sized series of %
% vectors to the reference and target ports of the controller block.                %
% --------------------------------------------------------------------------------- %
% [Setpoint, time, target]  = Ref                                                   %
%                                                                                   %
% Block input:                                                                      %
%       ------                                                                      %
%                                                                                   %
% Block outputs:                                                                    %
%       Setpoint :  set of vectors containing reference state values for each       %
%                             prediction step                                       %
%       time     :  current time step, currently used for error checking only       %
%       target   :  reference for current actuation                                 %
%                                                                                   %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Setpoint, time, target] = Ref
%#codegen

%% Set persistent and data
persistent time_step;                                        % Current time step [0.5 s]
if isempty(time_step)
    time_step=1;
end
persistent data
persistent refdata                                 % Load reference data,
if isempty(data)                                   % but only if it hasn't been loaded yet

data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');
    refdata=zeros(700,10);

    k=1;                                                       % Scale reference to 75 s
    for j=1:16:1201
        refdata(k,:)=data(j,1:10);
        k=k+1;
    end
    for k=350:1:700
        refdata(k,:)=data(1201,1:10);
    end
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
             0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
             0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
            -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
             0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
              0.706008414024590, 0.038656066811025;
    0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
             -0.038964665502406, 0.705293981181349;
    0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
              0.004125706617421, 0.032558403012700;
    0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
              0.706022382389280, 0.038734429975730;
    0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
              -0.038964497175050, 0.705290934311349];


%% Initialize vectors
Setpoint=zeros(40,12);


%% Set the series of reference vectors. Prediction horizon = 40 vectors are set from the
reference data for each time step

if time_step+39<=360
    thets=Mtheta'*refdata(time_step:time_step+39,2:10)';
    Setpoint=[refdata(time_step:time_step+39,2:7),thets'];

elseif time_step>=361
    thets=Mtheta'*refdata(361,2:10)';
    Setpoint=[repmat(refdata(361,2:7),40,1),repmat(thets',40,1)];
else
end

%% Set the actuation target for the current prediction from reference data
Ux=refdata(time_step,8);
Uy=refdata(time_step,9);
Uz=refdata(time_step,10);
target=[Ux;Uy;Uz];

%% Increment time
time=time_step;
time_step=time_step+1;
```

One additional simulator was constructed which does not make use of any robust terms to provide time data for the nominal MPC case. This simulator is given in the following.

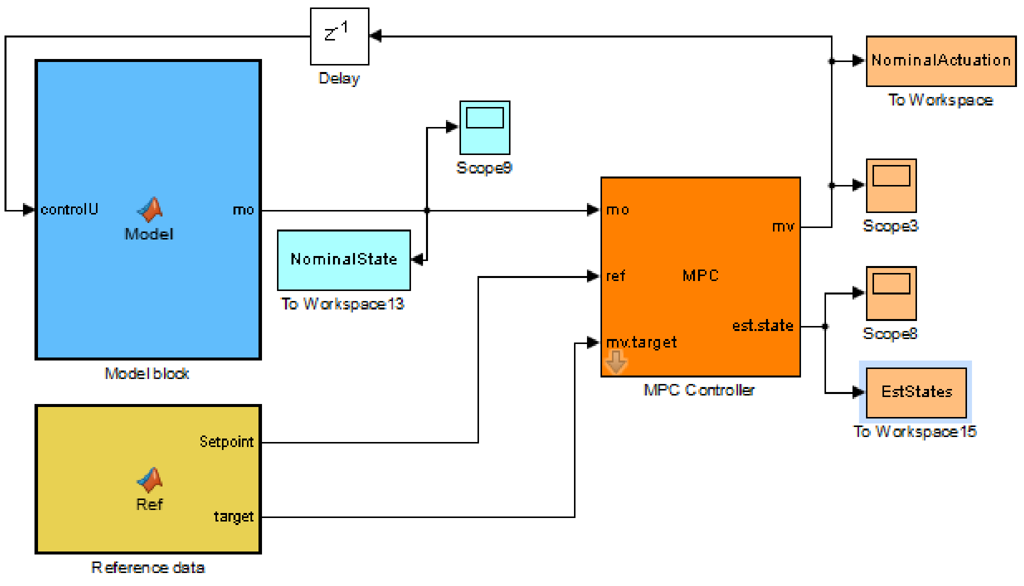


Figure Appendix - 2 Simulation of nominal control only

The MPC controller block is the same as in the previous simulator. The basis of the Model and Reference blocks is similar to the above:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODEL BLOCK: Simulation state and output estimation for Nominal sim w/out tube.   %
% --------------------------------------------------------------------------------- %
% This function handles state initialization and updates for the simulation.        %
% --------------------------------------------------------------------------------- %
% mo  = fcn(controlU)                                                               %
%                                                                                   %
% Block input:                                                                      %
%       controlU : robust control action, input u to the state space model          %
%                                                                                   %
% Block outputs:                                                                    %
%       mo    :  current nominal state of the system                                %
%                                                                                   %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function mo  = Model(controlU)
%#codegen

%% Set system variables initial conditions for the states and system model

persistent time                                    % Time-step counter (=0.5[s])
if isempty(time)                                   % If not set,
    time=1;                                        % then set time to first time step.
end

persistent dx                                   % Relative position: radial [m]
if isempty(dx)                                  % If not set,
    dx=39;                                      % set to initial radial position state.
end
persistent dy                                   % Relative position: along-track [m]
if isempty(dy)                                  % If not set,
    dy=39;                                      % set to initial along-track
end                                             % position state.
persistent dz                                   % Relative position: cross-track [m]
if isempty(dz)                                  % If not set,
    dz=4.23;                                    % set to initial cross-track
end                                             % position state.
persistent ddx                                  % Relative velocity: radial [m/s]
if isempty(ddx)                                 % If not set,
    ddx=0;                                      % set to initial radial velocity state.
end
persistent ddy                                  % Relative velocity: along-track [m/s]
if isempty(ddy)                                 % If not set,
    ddy=0;                                      % set to initial along-track
end                                             % position state.
persistent ddz                                  % Relative velocity: cross-track [m/s]
if isempty(ddz)                                 % If not set,
    ddz=0;                                      % set to initial cross-track
end
persistent next_state                           % Initialize persistent variable to
if isempty(next_state)                          % carry over state to next time step
    next_state=zeros(1,12);
    next_state=[dx dy dz ddx ddy ddz theta'];
end
persistent data                                 % Initialize variable to hold reference
if isempty(data)                                % Only load reference data if it has not
data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');                                         % yet been loaded.
end

if isempty(controlU)
    controlU=[data(1,8),data(1,9),data(1,10)]';
```

```
end

%% Set current states
current_state=next_state;
realX=current_state(1:6)'+w(1:6);


%% State Update
% % CTS model
% % norbit=0.0012;
% A=[0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3*norbit^2 0 0 0 2*norbit 0;
%      0 0 0 -2*norbit 0 0; 0 0 -norbit^2 0 0 0];
% B=[0 0 0; 0 0 0; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
% C=eye(6);
% D=zeros(6,3);
%
% % sys=ss(A,B,C,D,0.5);
% % [A,B,C,D]=ssdata(sys);

% %  DIS Model
% %  Sample time  0.5 s
% State matrix
% A=[1 0 0 0.5 0.0003 0; -2.16e-10 1 0 -0.0003 0.5 0; 0 0 1 0 0 0.5;
%      2.16e-6 0 0 1 0.0012 0; -1.296e-9 0 0 -0.0012 1 0; 0 0 -7.2e-7 0 0 1];
% Input matrix
% B=[ 0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5];
% Output matrix
% C=eye(6);
% Feedthrough matrix
% D=zeros(6,3);

% Extend to incorporate terminal theta cost, extra states only processed in
% cost function in final the offset cost term.
% State matrix
A=[1 0 0 0.5 0.0003 0 0 0 0 0 0 0; -2.16e-10 1 0 -0.0003 0.5 0 0 0 0 0 0 0;
   0 0 1 0 0 0.5 0 0 0 0 0 0; 2.16e-6 0 0 1 0.0012 0 0 0 0 0 0 0;
   -1.296e-9 0 0 -0.0012 1 0 0 0 0 0 0 0; 0 0 -2.6e-6 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0; 0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 0 0 0 0 1];
% Input matrix
B=[0.125 5e-5 0; -5e-5 0.125 0; 0 0 0.125; 0.5 0.0003 0; -0.0003 0.5 0; 0 0 0.5; 0 0 0;
    0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];
% Output matrix
C=[eye(12)];
% Feedthrough matrix
D=zeros(12,3);

% Calculate next sate and output
Xplus=A*current_state'+B*controlU;
Y=C*current_state'+D*controlU;
dx=Xplus(1);
dy=Xplus(2);
dz=Xplus(3);
ddx=Xplus(4);
ddy=Xplus(5);
ddz=Xplus(6);
theta=Mtheta'*([Xplus(1);Xplus(2);Xplus(3);Xplus(4);Xplus(5);Xplus(6);controlU(1);control
U(2);controlU(3)]'+w(1:9)')';
theta_out=theta;

next_state=[Xplus(1:6)' theta'];
% Set nominal state output
mo=current_state';
next_state=Xdot';
time=time+1;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REFERENCE BLOCK: Simulation reference vectors for Nominal sim w/out tube.          %
% ------------------------------------------------------------------------------- %
% This function parses the reference data to send the appropriately sized series of %
% vectors to the reference and target ports of the controller block.               %
% ------------------------------------------------------------------------------- %
% [Setpoint, time, target]  = Ref                                                  %
%                                                                                   %
% Block input:                                                                      %
%       ------                                                                      %
%                                                                                   %
% Block outputs:                                                                    %
%       Setpoint :  set of vectors containing reference state values for each       %
%                         prediction step                                           %
%       time     :  current time step, currently used for error checking only       %
%       target   :  reference for current actuation                                 %
%                                                                                   %
% ------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technische Universität München, 2016.                                    %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Setpoint, target] = Ref
%#codegen

%% Set persistent and data
persistent time_step;                                   % Current time step [0.5 s]
if isempty(time_step)
    time_step=1;
end
persistent data                                       % Load reference data,
if isempty(data)                                      % but only if it hasn't been loaded yet
data=load('C:\Users\Carly\Desktop\DLR\Data\tumble\tumble_wy_3_5_orbital_plane_step05sec_1
.dat');
end

%% Set parameterization matrix M_theta (refer to: sections 4.4.2 and 7.4)
Mtheta=[ -0.185551526369166, 0.018117071393572, 0.047743479379178, 0.706328547648448,
              0.005820151954137, 0.032651318626743;
  -0.002300266364531, -0.182713737993363, 0.060394004339685, -0.007604460874062,
              0.706008414024590, 0.038656066811025;
  -0.051013496976408, -0.057658466070907, -0.176380728691161, -0.032282424810556,
             -0.038964665502406, 0.705293981181349;
   0.185551526369166, -0.018117071393571, -0.047743479379178, 0.706328547648448,
              0.005820151954136, 0.032651318626743;
   0.002300266364531, 0.182713737993363, -0.060394004339685, -0.007604460874062,
              0.706008414024590, 0.038656066811025;
   0.051013496976408, 0.057658466070907, 0.176380728691161, -0.032282424810556,
             -0.038964665502406, 0.705293981181349;
   0.185546807312485, -0.018555662630504, -0.047598740020594, 0.706343747015220,
              0.004125706617421, 0.032558403012700;
   0.002745590027817, 0.182670257022019, -0.060508588690195, -0.005909272359706,
              0.706022382389280, 0.038734429975730;
   0.051013717354715, 0.057658715155481, 0.176381490655910, -0.032282285350481,
              -0.038964497175050, 0.705290934311349];


%% Initialize vectors
Setpoint=zeros(40,12);
Setpoints_3D=data(:,2:7);


%% Set the series of reference vectors. Prediction horizon = 40 vectors are set from the
reference data for each time step
if time_step+39<=1340
    thets=Mtheta'*data(time_step:time_step+39,2:10)';
    Setpoint=[Setpoints_3D(time_step:time_step+39,:),thets'];
```

```
elseif time_step==1341
    thets=Mtheta'*data(1341,2:10)';
    Setpoint=[repmat(Setpoints_3D(1341,:),40,1),repmat(thets',40,1)];
else
end

%% Set the actuation target for the current prediction from reference data

Ux=data(time_step,8);
Uy=data(time_step,9);
Uz=data(time_step,10);
target=[Ux;Uy;Uz];

% Increment time
time=time_step;
time_step=time_step+1;
```

D. Validation

The final pair of functions were used for result validation. The first of the two functions below implements the inverse dynamics and the second conducts the analytical CWH equation solution.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Validation: Inverse dynamics                                                     %
% -------------------------------------------------------------------------------- %
% This script implements the inverse dynamics problem on the simulated results     %
% to evaluate the appropriateness of the simulated actuation.                      %
% -------------------------------------------------------------------------------- %
% -------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver  %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,     %
%          Technishe Universität München, 2016.                                    %
%                                                                                  %
% Copyright: Caroline Buckner                                                      %
% Last edited: 22 Dec 2016                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initialize constants and vectors
n=0.0012;                                                                 % Orbital rate

A=[        1 0        0      0.5 0.0003   0;                              % State matrix
   -2.16e-10 1        0  -0.0003    0.5   0;
           0 0        1        0      0 0.5;
     2.16e-6 0        0        1 0.0012   0;
   -1.296e-9 0        0  -0.0012      1   0;
           0 0 -2.16e-6        0      0   1];
B=[        0      0   0;                                                  % Input matrix
           0      0   0;
           0      0   0;
         0.5 0.0003   0;
     -0.0003    0.5   0;
           0      0 0.5];
acc=zeros(3,1201);                                                      % result vector

%% Import simulation data
inputData=simout8.data;                                               % robust actuation
nomIP=nomActuation;                                                   % nominal actuation
state=RealState;                                                      % real states
nomstate=ModelOutputs;                                                % nominal states

%% Inverse problem
% for simulation duration
for k=1:1:1200
```

```
    acc(:,k)=inv(B'*B)*B'*(state(k+1,:)'-(A*state(k,:)')); % Determine the required
                                        % robust actuation to obtain the suggested
                                        % series of states
    DRterm(:,k)=acc(:,k)-nomIP(k,:)';   % Required disturbance rejection term
    nomacc(:,k)=inv(B'*B)*B'*(nomstate(k+1,1:6)'-(A*nomstate(k,1:6)'));% Determine
                                        % required nominal actuation to obtain
                                        % suggested nomial states

    dif1(k,:)=acc(:,k)'-inputData(k,:); % Difference between determined
                                        % and simulated robust actuation
    dif2(k,:)=acc(:,k)'-nomacc(:,k)';   % Difference between determined
                                        % robust and nominal actuation
end

%% Create figures
     figure;
    hold on
    plot(0:0.5:600,acc)                        % plot inverse determined r. actuation
    plot(0:0.5:600,inputData)                  % plot simulated robust actuation
    title('Direct (simulated) and inverse determined actuation')
    xlabel('Time (s)')
    ylabel('Acceleration input (m/s^2)')
    figure;
    plot(0:0.5:599.5,dif2)                    % plot difference between robust and
                                              % nominal actuation
    title('diff inv calc nom and calculated inputs')
    xlabel('Time (s)')
    ylabel('Acceleration (m/s^2)')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Validation: Analytical solution of the CWH equations                              %
% --------------------------------------------------------------------------------- %
% This script implements an analytical solution of the CWH equations on the         %
% simulated positions of the chaser.                                                %
% --------------------------------------------------------------------------------- %
% --------------------------------------------------------------------------------- %
% FROM: C. Buckner. Tube-based Model Predictive Control for the Approach Maneuver   %
%          of a Spacecraft to a free-tumbling Target Satellite. Master Thesis,      %
%          Technishe Universität München, 2016.                                     %
%                                                                                   %
% Copyright: Caroline Buckner                                                       %
% Last edited: 22 Dec 2016                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Set constants and initialize vectors/matrices
n=0.0012;                                                  % orbital rate
time=0:0.5:600;                                            % simulation duration
num=length(time);                             % length of the simulation duration used
                                              % to initialize the following matrices
Axy=zeros(2*num,4);                 % Jacobian for the radial and along-track dynamics
Az=zeros(num,2);                    % Jacobian for the cross-track dynamics

pos = zeros(1201,3);                % Initialize vector of results

% Import results from simulation
differencePosR=simout6.data(:,1);
differencePosAlongTrack=simout6.data(:,2);
differencePosCrossTrack=simout6.data(:,3);

%% Analytical solution
% For each timestep
t=0;

for k=1:1:1201

%Determine current Jacobian matrices
Axy=[1 (4/n)*sin(n*t)-3*t 6*(sin(n*t)-n*t) 2*(cos(n*t)-1)/n;
    0 2*(1-cos(n*t))/n 4-3*cos(n*t) sin(n*t)/n];
```

```

Az=[cos(n*t) sin(n*t)/n];

% Determine initial conditions through inverse problem
xyinit(1:4,k)=pinv(Axy)*[differencePosR(k) differencePosAlongTrack(k)]';
zinit(:,k)=pinv(Az)*differencePosCrossTrack(k);

initRelPos(k,:) = [xyinit(1,k),zinit(1,k),xyinit(3,k)];
initRelVel(k,:) = [xyinit(2,k),zinit(2,k),xyinit(4,k)];

% Determine analytical positions through direct problem
pos(k,1) =
(2/n)*initRelVel(k,3)*cos(n*t)+((4/n)*initRelVel(k,1)+6*initRelPos(k,3))*sin(n*t)-...
    (3*initRelVel(k,1)+6*n*initRelPos(k,3))*t+initRelPos(k,1)-(2/n)*initRelVel(k,3);
pos(k,3) = ((-2/n)*initRelVel(k,1)-
3*initRelPos(k,3))*cos(n*t)+(initRelVel(k,3)/n)*sin(n*t)+...
    2*initRelVel(k,1)/n+4*initRelPos(k,3);
pos(k,2) = initRelPos(k,2)*cos(n*t)+initRelVel(k,2)*sin(n*t)/n;

% Increment time
t=t+0.5;
end

%% Create figure
figure;
hold on
plot3(pos(:,1),pos(:,2),pos(:,3))                              % plot analytical solution
plot3(differencePosR,differencePosCrossTrack,differencePosAlongTrack,'r')
                                                               % plot simulated solution
xlabel('Alongtrack')
ylabel('Crosstrack')
zlabel('Radial')
title('Relative position')


```