A Nonlinear Model Predictive Control for payload transportation and formation flying by a free-flyer robot

Rui Filipe Freitas Correia
ruifcorreia@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

October 2020

Abstract

The use of free-flyer robots in space applications has been increasing in recent years. Several space agencies, such as NASA, DLR and JAXA, have at this moment, or intend to have, projects involving this type of vehicles. The increasing interest in these robots can be associated with the desire to expand the limits of present space exploration. Inherent to this interest is the necessity to develop control systems capable of autonomous navigation and task execution while considering a minimum energy expenditure. This work presents a performance study of nonlinear model predictive control applied to the free-flyer Space CoBot, developed by ISR-Lisboa, within a real-time operation context, subjected to different mission conditions which aim to explore the limits of its operational stability and reliability. For performing guidance tasks, an offline trajectory generation method was integrated, capable of handling not only highly confined spaces, composed by walls and obstacles but also additional constraints such as control actuation limitations and system dynamics. These trajectories are provided to the controller as references. All tests conducted in this work are obtained exclusively through an accurate simulation environment based in ROS and Gazebo/RotorS.
Keywords: Free flyer robots, Fast NMPC, Offline trajectory optimization, Control, Guidance

Introduction

In recent decades, the interest in Unmanned Aerial Vehicles (UAV) has experienced considerable increases. This is mainly due to the ability to execute actions without an onboard pilot, whether it is controlled remotely or by an autopilot system, commonly composed by several onboard elements that originate Guidance, Navigation and Control (GNC) systems [1]. This allows a vast range of applications that reduce operating costs, operations with no risk to human life or even tasks that would be otherwise impossible to manned vehicles. An even more recent trend is the use of small robots in the International Space Station (ISS). National Aeronautics and Space Administration (NASA) free-flyer robots are one example and they have been onboard the ISS for over a decade. Initially they were a platform used for many experiments from flight directors and researchers on the ground that aimed at improving formation flying and docking algorithms [2]. This research has since allowed the development of new free-flyer robots that include more features and can perform a variety of Intravehicular Activity (IVA) work in the ISS [3]. The Institute For Systems and Robotics (ISR-Lisboa) has also developed a highly maneuverable holonomic free-flyer robot designated Space CoBot [4, 5], which provides a similar platform for conducting research in this type of robots.

Currently, robotic arms are already used to assist humans in maintenance tasks in the ISS, namely the *Canadarm2* [6]. However, in a remote space station there could be significant limitations to these systems. Free-flyer robots will allow more complex operations without human presence. In order to achieve this level of autonomy and unsupervised operation, several aspects have to be extensively studied, such as the vehicle interaction with the station, objects, tools, and astronauts, in the case of a manned station. Its navigation system must guarantee high awareness and risk-free maneuvers. Lastly, but definitely not less important, the control system must use the least amount of energy possible as access to energy might be severely restricted.

State-of-Art

NASA has been using free flyers as research platforms on board the ISS as early as 2006 through Synchronized Position Hold Engage Reorient Experimental Satellites (SPHERES) [2]. SPHERES is a formation flight testing facility consisting of three satellites operating onboard the ISS. It aims to use the microgravity environment to further develop formation flight and docking algorithms. Astrobee

[3] is a similar platform that replaced SPHERES as a testbed onboard the ISS. Astrobee is expected to increase and improve SPHERES features, specially when it comes to navigation and possible interactions with its surroundings, such as cargo transfers and maintenance tasks.

Japan Aerospace Exploration Agency (JAXA) has developed Int-Ball [7], which is able to move autonomously inside the ISS and is used for photographing and filming the crew.

German Aerospace Center (Deutsches Zentrum fr Luft- und Raumfahrt) (DLR) has also developed a free flyer designated Crew Interactive MObile companioN (CIMON) [8] which is currently on the second iteration, CIMON-2, already onboard the ISS as of 2020. The project aims to develop autonomous astronaut assistance through artificial intelligence.

Nonlinear Model Predictive Control (NMPC) is an extension of optimal control methods, which consists in solving an Optimal Control Problem (OCP) repeatedly at each time step. The feedback control law is obtained by applying the first element of the solution at each time step to the system and discarding the rest. NMPC cannot rely on analytical control laws. Alternatively, it relies on discretization techniques that transform the infinite OCP into a standard Nonlinear Programming (NLP) problem that can be solved through general NLP solvers [9]. There are two main families of NLP solvers, Interior Point (IP) and Sequential Quadratic Programming (SQP) solvers.

Currently, there is a wide variety of algorithms available based in both solver families. Recent developments are briefly presented and compared in [10], where an effort to close the gap to the real-time frame is visible. Methods that are worth mention for their extensive presence in literature as benchmark methods are the SQP based SNOPT [11] and the IP based IPOPT [12].

NMPC is very versatile as it can take several roles within the control system.

Guidance tasks such as trajectory planning as in [13], where a NMPC is proposed to adjust a trajectory in the presence of obstacles by incorporating new constraints during operation.

Interestingly, NMPC are being considered to perform low level control tasks, which requires fast and stable rates of operation.

The hierarchical controller implemented in [14] uses a NMPC controller for the attitude of a Multirotor Aerial Vehicle(MRAV) and registered execution averages of $1\,ms$ while operating at a $5\,ms$ sampling time. Although it only controls the attitude component of the system, it represents a very reasonable computation delay.

In [15], a single non-hierarchical real-time NMPC controller implemented in a fixed wing aircraft is used to control both position and attitude for tracking a given reference trajectory while avoiding obstacles and respecting actuator bounds. The execution time required by the NMPC controller averages at $30\,ms$ while operating at a sampling time of $500\,ms$. This can be an indication of the impact of system complexity on execution time.

Also implemented in a MRAV, in [16], a hierarchical controller is used to track a feasible trajectory generated offline. A position NMPC controller computes attitude references feeding a second attitude NMPC controller, operating with sampling times of $200\,ms$ and $2\,ms$, respectively.

The application of NMPC in a MRAV with Multi-Directional Thrust (MDT) capability is studied in [17]. A NMPC controller is used to compute force inputs for each rotor to track both position and attitude references.

Background

Optimization methods allow the formulation of an OCP that can be solved by numerical and optimization solvers. Optimality is defined as the minimization or maximization of an objective function, that can be defined arbitrarily to describe the optimal solution [18].

The term optimization variable is used to describe a variable that the optimization solver adjusts to minimize the objective function. These can be the initial and final time, t_0 and t_f, as well as state and control variables, $\mathbf{x}(t)$ and $\mathbf{u}(t)$, respectively [19]. The set of optimization variables will be henceforth designated by $\mathbf{z} := (t_o, t_f, \mathbf{x}(t), \mathbf{u}(t))$.

The optimization is subject to a variety of limits and constraints, namely dynamic constraints (2), path constraints (3), and boundary constraints (4).

This concludes the formulation of a general continuous-time OCP

$$\min_{t_0,t_f,\mathbf{x},\mathbf{u}} J(t_0,t_f,\mathbf{x}(t_0),\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \ell(\mathbf{x}(t),\mathbf{u}(t),t)dt \quad (1)$$

$$\text{subject to} \quad \dot{\mathbf{x}} = f(\mathbf{x}(t),\mathbf{u}(t),t), \quad (2)$$

$$h(\mathbf{x}(t),\mathbf{u}(t),t) \leq 0, \quad (3)$$

$$g(t_0,t_f,\mathbf{x}(t_0),\mathbf{x}(t_f)) \leq 0 \quad (4)$$

There are several ways to approach an OCP. The two main approaches are indirect and direct methods.

Indirect Methods use the necessary conditions of optimality of the continuous problem to derive a Boundary Value Problem (BVP) in Ordinary Differential Equations (ODE) [20]. This BVP must be numerically solved, and the approach is often viewed as "first optimize, then discretize". The class of indirect methods encompasses the well known calculus of variations and the Pontryagin Maximum Principle. These methods often treat only control inputs as optimization variables while implicitly enforcing dynamic constraints

through the system's simulation.[21]. This propagation is highly dependent on the initial state value. Common indirect methods are Differential Dynamic Programming (DDP) [22] and iterative Linear Quadratic Regulator (iLQR) [23]. It is also worth mentioning the Augmented Lagrangian TRajectory Optimizer (ALTRO) algorithm [21], which combines both indirect and direct methods to achieve advantages of both approaches.

Direct methods transform the original infinite OCP into a finite NLP. This NLP is then solved by variants of numerical optimization methods [18]. It is often viewed as "first discretize, then optimize", which is an intuitive way of distinguishing direct and indirect methods. All direct methods are based on a finite dimensional parameterization of the control trajectory, but differ in the way the state trajectory is handled [20]. Nowadays, direct methods such as Direct Transcription (DIRTRAN) [24], Direct Collocation (DIRCOL) [25], and Multiple shooting [9] are the most widespread and successfully used techniques.

Approach

Throughout this work, three system configurations are considered. A single Space CoBot system, a Space CoBot carrying a payload system and a formation system comprising two Space CoBots and a shared payload.

Space CoBot model

The Space CoBot is an hexarotor UAV with a special rotor design configuration, that can be visualized in Figure 1.

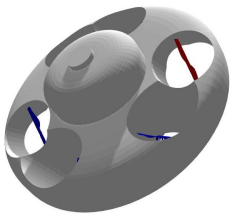

Figure 1: Space CoBot 3D model.

It is modelled through the Newton-Euler equations of motion following [5]. In this work, quaternions are used to describe rotation instead of Euler angles. The resulting model is described by

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v} \\ \dot{\mathbf{v}} = (1/m)\,\mathbf{R}(\mathbf{q})\,\mathbf{F} \\ \dot{\mathbf{q}} = (1/2)\,\mathbf{Q}(\mathbf{q})\,\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\mathbf{M} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}) \end{cases} \tag{5}$$

where $\mathbf{p} = [x\,, y\,, z]$ is the position, $\mathbf{v} = [u\,, v\,, w]$ corresponds to linear velocity, and $\mathbf{q} = [q_w\,, q_x\,, q_y\,, q_z]$ is the attitude quaternion. These components are described with respect to (w.r.t.) the inertial frame $\mathcal{I}$, and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$ is the angular velocity described w.r.t. the body frame $\mathcal{B}$. These elements constitute the state vector $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{q}, \boldsymbol{\omega}]$. The scalar m is the system's mass, $\mathbf{J}$ is the moments of inertia matrix and $\mathbf{F}$ and $\mathbf{M}$ are vectors representing the thrust force and torque generated by the rotors w.r.t. the body frame. $\mathbf{R} \in SO(3)$ is the rotation matrix obtained by considering right-hand rotation and defined by

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_zq_w & 2q_xq_z+2q_yq_w \\ 2q_xq_y+2q_zq_w & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_xq_w \\ 2q_xq_z-2q_yq_w & 2q_yq_z+2q_xq_w & 1-2q_x^2-2q_y^2 \end{bmatrix} \tag{6}$$

and $\mathbf{Q}$ is a 4×3 matrix defined by

$$\mathbf{Q}(\mathbf{q}) = \begin{bmatrix} q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \\ -q_x & -q_y & -q_z \end{bmatrix} \tag{7}$$

Propulsive system model

The propulsive system model follows the approach in [5] where each rotor i generates a scalar thrust f_i and a torque τ_i described by

$$f_i = K_1 u_i, \quad \tau_i = w_i K_2 u_i, \quad i \in \{1, \dots, 6\} \tag{8}$$

where K_1 and K_2 are constants used to describe the propeller properties, w_i is either -1 or 1 depending on whether the propeller rotates clockwise or anticlockwise for a positive forward thrust $f_i > 0$. The actuation signal u_i, expressed in rps, is defined by $u_i = \text{sgn}(n_i)\, n_i^2$, where sgn() is a sign function, to achieve a linear relation between the actuation and forces/moments.

Each rotor is represented by a relative position to the Center of Mass (CoM), expressed by the vector $\mathbf{r}_i$, orthogonal to the z axis w.r.t. $\mathcal{B}$, and the unit vector $\hat{u}_i$, aligned with the propeller orientation axis. Both are uniquely defined by the angles θ_i and ϕ_i through the following expressions

$$\mathbf{r}_i = \begin{pmatrix} d\cos(\theta_i) \\ d\sin(\theta_i) \\ 0 \end{pmatrix} \quad \hat{u}_i = \begin{pmatrix} \sin(\theta_i)\sin(\phi_i) \\ -\cos(\theta_i)\sin(\phi_i) \\ \cos(\theta_i) \end{pmatrix}, \quad i \in \{1, \dots, 6\} \tag{9}$$

where $d = ||\mathbf{r}_i||$ is the distance from the propeller to the CoM.

The resulting thrust $\mathbf{F}_i$ and torque $\mathbf{M}_i$ are obtained

$$\begin{pmatrix} \mathbf{F}_i \\ \mathbf{M}_i \end{pmatrix} = \mathbf{a}_i u_i, \quad i \in \{1, \dots, 6\} \tag{10}$$

where $\mathbf{a}_i$ represents the components of force and torque of each rotor i and is defined by

$$\mathbf{a}_i = \begin{pmatrix} K_1 \hat{u}_i \\ K_1 \mathbf{r}_i \times \hat{u}_i - w_i K_2 \hat{u}_i \end{pmatrix}, \quad i \in \{1, \dots, 6\} \tag{11}$$

The resulting net force and torque will be the sum of the contributions of the 6 rotors. In matrix form

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{M} \end{pmatrix} = \mathbf{A}\mathbf{u} \tag{12}$$

where $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_6]$ is a square matrix hereby called actuation matrix and $\mathbf{u} = [u_1 \dots u_6]^T$ is the actuation input.

Space CoBot with payload model
To redefine the model with payload, one can resort to the *Parallel axis* theorem to compute the resulting inertial matrix around the CoM of the resulting system.

In order to perform this modification, rigid body conditions are considered. An illustration of the considered system can be observed in Figure 2 below.

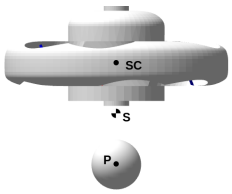

Figure 2: Space CoBot with payload configuration.

The new CoM represented by point S presents a displacement in the z axis only w.r.t. $\mathcal{B}$. In general, the CoM can be found by vector addition of the weighted position vectors which point to the CoM of each object in a system.

Following [26], the *Parallel axis* theorem states

$$\mathbf{J}_O = \mathbf{J}_G + m_{system}\left(\mathbf{r}_G^T\mathbf{r}_G\mathbf{I} - \mathbf{r}_G\mathbf{r}_G\right) \tag{13}$$

where $\mathbf{r}_G$ is a vector representing the position of a given point G relative to a given point O. $\mathbf{I}$ is the identity matrix.

The desired inertia can then be obtained by

$$\mathbf{J}_S = \mathbf{J}_{T1} + \mathbf{J}_{T2} \tag{14}$$

where $\mathbf{J}_{T1}$ and $\mathbf{J}_{T2}$ are defined by

$$\mathbf{J}_{T1} = \mathbf{J}_{SC} + m_{space\ cobot}\left(\mathbf{r}_{SC}^T\mathbf{r}_{SC}\mathbf{I} - \mathbf{r}_{SC}\mathbf{r}_{SC}\right) \tag{15}$$

and

$$\mathbf{J}_{T2} = \mathbf{J}_P + m_{payload}\left(\mathbf{r}_P^T\mathbf{r}_P\mathbf{I} - \mathbf{r}_P\mathbf{r}_P\right) \tag{16}$$

The deviation in the CoM requires an adjustment in the position vector of each rotor as seen in Figure 3 and described by

$$\mathbf{r}_{p_i} = \mathbf{c}_{off} + \mathbf{r}_i \tag{17}$$

The resulting model is then

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v} \\ \dot{\mathbf{v}} = (1/m_{system})\,\mathbf{R}(\mathbf{q})\,\mathbf{F} \\ \dot{\mathbf{q}} = (1/2)\,\mathbf{Q}(\mathbf{q})\,\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} = \mathbf{J}_S^{-1}(\mathbf{M} - \boldsymbol{\omega} \times \mathbf{J}_S\boldsymbol{\omega}) \end{cases} \tag{18}$$

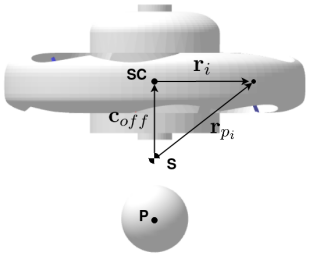

Figure 3: Visualization of vector operation to obtain rotor positions with payload.

Formation with payload model
This formation will be considered as a single system where all elements are rigidly linked, comprising a rigid body. In this case, the model will be defined around the CoM of the complete system. The resulting configuration can be observed in Figure 4.

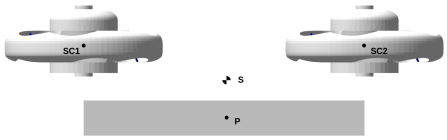

Figure 4: Formation with payload configuration.

The total inertia of the system can be calculated by summing the inertia of each element expressed around point S, the CoM of the system, using the same method mentioned in the previous section resulting in the following expression

$$\mathbf{J}_S = \mathbf{J}_{T1} + \mathbf{J}_{T2} + \mathbf{J}_{T3} \tag{19}$$

where $\mathbf{J}_{T1}$, $\mathbf{J}_{T2}$ and $\mathbf{J}_{T3}$ are individual inertial elements expressed in point S.

The deviation in the CoM also requires an adjustment in the position vector of each rotor as seen in Figure 5 and described by

$$\mathbf{r}_{f_i} = \mathbf{r}_i + \mathbf{c}_{off} \tag{20}$$

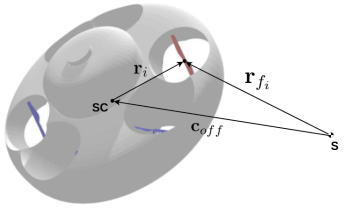

Figure 5: Visualization of vector operation to obtain rotor positions in formation.

The resulting actuation model of the individual Space CoBot can then be defined by

$$\begin{pmatrix}\mathbf{F}_i\\ \mathbf{M}_i\end{pmatrix} = \mathbf{a}_i u_i \quad \mathbf{a}_i = \begin{pmatrix} K_1\hat{u}_i \\ K_1\mathbf{r}_{f_i}\times\hat{u}_i - w_i K_2\hat{u}_i\end{pmatrix} \tag{21}$$

The resulting force and torque will be the sum of the contributions of all 12 rotors in the formation. In matrix form

$$\begin{pmatrix}\mathbf{F}_1+\mathbf{F}_2\\ \mathbf{M}_1+\mathbf{M}_2\end{pmatrix} = \mathbf{A}_1\mathbf{u}_1 + \mathbf{A}_2\mathbf{u}_2 \tag{22}$$

where $\mathbf{A}_i = [\mathbf{a}_{i_1} \dots \mathbf{a}_{i_6}]$ are the actuation matrices of each Space CoBot and $\mathbf{u}_1$ and $\mathbf{u}_2$ are individual control inputs corresponding to each Space CoBot, which compose the actuation input $\mathbf{u} = \begin{bmatrix}\mathbf{u}_1^T & \mathbf{u}_2^T\end{bmatrix}^T$.

The dynamics of the system described through the CoM of the system is then obtained by summing the contributions of each Space CoBot and are defined by

$$\begin{cases}\dot{\mathbf{p}} = \mathbf{v}\\ \dot{\mathbf{v}} = (1/m_{system})\,\mathbf{R}(\mathbf{q})\,(\mathbf{F}_1+\mathbf{F}_2)\\ \dot{\mathbf{q}} = (1/2)\,\mathbf{Q}(\mathbf{q})\,\boldsymbol{\omega}\\ \dot{\boldsymbol{\omega}} = \mathbf{J}_S^{-1}(\mathbf{M}_1+\mathbf{M}_2-\boldsymbol{\omega}\times\mathbf{J}_S\boldsymbol{\omega})\end{cases} \tag{23}$$

Note that the states of the model mentioned above in (23) are described in the CoM of the formation while each Space CoBot has sensors positioned in their own CoM. Therefore, this information is transformed to the CoM of the formation by following Euler's equations

$$\begin{aligned}\mathbf{p}_A &= \mathbf{p}_B + \mathbf{r}_{BA}\\ \mathbf{v}_A &= \mathbf{v}_B + \boldsymbol{\omega}_{b/i}\times\mathbf{r}_{BA}\\ \mathbf{a}_A &= \mathbf{a}_B + \boldsymbol{\alpha}_{b/i}\times\mathbf{r}_{BA} + \boldsymbol{\omega}_{b/i}\times(\boldsymbol{\omega}_{b/i}\times\mathbf{r}_{BA})\end{aligned} \tag{24}$$

NLP formulation

To achieve a NMPC formulation, an NLP must be formulated. To perform the problem discretization, multiple shooting is used.

First the controls are discretized piecewise on a given time grid

$$\mathbf{u}(t) = \mathbf{q}_i \quad \text{for} \quad t\in[t_i, t_{i+1}] \tag{25}$$

Then the ODE is solved on each interval $[t_i, t_{i+1}]$ independently, starting with an artificial initial value $\mathbf{s}_i$, the so-called shooting node [9]

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}_i(t), \mathbf{q}_i), \quad t\in[t_i, t_{i+1}], \tag{26}$$

$$\mathbf{x}_i = \mathbf{s}_i \tag{27}$$

By numerically solving these initial value problems, the state trajectory pieces $\mathbf{x}_i(t, \mathbf{s}_i, \mathbf{q}_i)$ are obtained [20].

Simultaneously with the decoupled ODE solution, any integrals in the cost function are also numerically computed at each point of the grid

$$l_i(\mathbf{s}_i, \mathbf{q}_i) := \int_{t_i}^{t_{i+1}} \ell(\mathbf{x}_i(t_i, \mathbf{s}_i, \mathbf{q_i}), \mathbf{q}_i)dt \tag{28}$$

In order to ensure dynamic feasibility, the continuity conditions $\mathbf{s}_{i+1} = \mathbf{x}_i(t_{i+1}, \mathbf{s}_i, \mathbf{q}_i)$ are imposed.

The 4^{th} order Runge-Kutta integration method applied and described in [27] is used to numerically solve the integrals.

Thus, arriving at the following NLP formulation

$$\underset{\mathbf{s},\mathbf{q}}{\text{minimize}} \quad \sum_{i=0}^{N} l_i(\mathbf{s}_i, \mathbf{q}_i) + J(\mathbf{s}_{N+1}) \tag{29}$$

$$\text{subject to} \quad \mathbf{s}_0 - \mathbf{x}_0 = 0, \tag{30}$$

$$\mathbf{s}_{i+1} - \mathbf{x}_i(t_{i+1}, \mathbf{s}_i, \mathbf{q}_i) = 0, \quad i = 0, \dots, N, \tag{31}$$

$$h(\mathbf{s}_i, \mathbf{q}_i) \geq 0, \quad i = 0, \dots, N, \tag{32}$$

$$g(\mathbf{s}_{N+1}) = 0. \tag{33}$$

All optimization variables can be summarized as $\mathbf{z} := (\mathbf{s}_0, \mathbf{q}_0, \dots, \mathbf{s}_{N+1}, \mathbf{q}_N)$.

Sequential Quadratic Programming

The SQP considered now, augments the objective function through the Lagrangian function $\mathcal{L}$ as in [28]. The objective is to apply Newton's method to the Karush-Kuhn-Tucker conditions as in [29] resulting in the following linear system

$$\begin{bmatrix}\mathbf{H}_L & \nabla_z\mathbf{C}^T\\ \nabla_z\mathbf{C} & 0\end{bmatrix}\begin{bmatrix}\Delta\mathbf{z}\\ -\boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}-\nabla_z F_J\\ -\mathbf{C}\end{bmatrix} \tag{34}$$

where $\mathbf{H}_L$ is the Hessian matrix of the Lagrangian, C is the set of equality and inequality constraints and F_J is the objective function being augmented.

According to [29], in this approach the search direction $\Delta\mathbf{z}$ can be computed by solving the Quadratic Programming (QP) subproblem

$$\text{minimize} \quad \frac{1}{2}\Delta\mathbf{z}^T\mathbf{H}_L\Delta\mathbf{z} + \nabla_z F_J^T\Delta\mathbf{z} \tag{35}$$

$$\text{with respect to} \quad \Delta\mathbf{z}\in\mathbb{R}^{n_z} \tag{36}$$

$$\text{subject to} \quad \nabla_z\mathbf{C}\Delta\mathbf{z} = -\mathbf{C} \tag{37}$$

Note that approximations of the constraints $\mathbf{C}$ are used (37). To increase the computation speed, an approximation of $\mathbf{H}_L$ is computed via the Gauss-Newton approach [27] that avoids the computation of second order derivatives. Additionally, the SQP process is stopped prematurely after one iteration instead of iterating until convergence as in [30].

NMPC controller

In this work, the ACADO toolkit [31] is used to generate a real-time solver in efficient C code, including the QP solver *qpoases* [32], which is the one selected to solve the SQP subproblems (35) in this work.

The generated solver is included in a ROS node, hereafter designated by *controller node*, that will compose the NMPC controller. The optimization horizon will have 20 intervals and a sampling time of 100 *ms*.

Actuation function

It is important to note some details regarding the integration of the ACADO solver. The objective is to work with control inputs expressed in *rpm*. If the optimal control input to be found by the solver is expressed in rpm^2, henceforth designated by $\mathbf{u}$, the set of possible solutions will be significantly big.

In order to remove some complexity of the model provided to the solver, the optimal control input to be found by the solver will be expressed in force, N, described by the force control input $\mathbf{u}_f$ as follows

$$\mathbf{u}_f = \lambda_{rpm}\mathbf{u} \tag{38}$$

where λ_{rpm} is a conversion factor defined by

$$\lambda_{rpm} = \frac{K_1}{60^2} \tag{39}$$

resulting in the modified actuation model

$$\begin{pmatrix}\mathbf{F}\\ \mathbf{M}\end{pmatrix} = \mathbf{A_N}\mathbf{u}_f \tag{40}$$

The control input $\mathbf{u}_f$ is then converted to angular velocity, $\mathbf{u}_{rpm}$, via

$$\mathbf{u}_{rpm} = \text{sgn}(\mathbf{u}_f)\sqrt{\frac{|\mathbf{u}_f|}{\lambda_{rpm}}} \tag{41}$$

The resulting controller node is now represented by Figure 6.

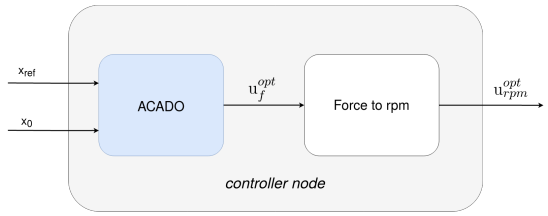

Figure 6: Visualization of the *controller node*.

The models provided to ACADO are based in (5), (18) and (23) for the respective configurations.

Objective function

To define the objective function to be minimized by the solver, a built-in function of ACADO [33] is used to generate the following

$$\sum_{k=0}^{N-1}[h(\mathbf{x}_k,\mathbf{u}_k)-y_k]\mathbf{W}_k[h(\mathbf{x}_k,\mathbf{u}_k)-y_k]^T + [h_N(\mathbf{x}_N,\mathbf{u}_N)-y_N]\mathbf{W}_N[h_N(\mathbf{x}_N,\mathbf{u}_N)-y_N]^T \tag{42}$$

where h are called reference functions and are denoted with $h \in \mathbb{R}^{n_y}$ and $h_N \in \mathbb{R}^{n_y,N}$, $y_k \in \mathbb{R}^{n_y}$ and $y_N \in \mathbb{R}^{n_y,N}$ denote minimizing reference values that can be fixed or time-varying, and $\mathbf{W}_k \in \mathbb{R}^{n_y}$ and $\mathbf{W}_N \in \mathbb{R}^{n_y,N}$ are the weighting matrices.

Position and attitude errors along with control inputs are selected to construct h and h_N. The objective is to set $y_{0,\dots,N} = 0$ so that minimizing the objective function implies reducing the position and attitude errors and selecting low control inputs.

The position error vector $\Delta\mathbf{p} \in \mathbb{R}^3$ can be trivially obtained by subtraction

$$\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}_{ref} \tag{43}$$

The attitude difference, $\Delta\boldsymbol{\Theta} \in \mathbb{R}^3$, is obtained by computing a truncated version of the quaternion error, $\mathbf{q}_e$ described in [34], yielding

$$\Delta\boldsymbol{\Theta} = \begin{bmatrix} q_w^{ref}q_x + q_z^{ref}q_y - q_y^{ref}q_z - q_x^{ref}q_w \\ -q_z^{ref}q_x + q_w^{ref}q_y + q_x^{ref}q_z - q_y^{ref}q_w \\ q_y^{ref}q_x - q_x^{ref}q_y + q_w^{ref}q_z - q_z^{ref}q_w \end{bmatrix} \tag{44}$$

The control inputs are used as is. Recalling that $y_{0,\dots,N} = 0$ the following objective function is formed

$$\sum_{k=0}^{N-1}[\Delta\mathbf{p}_k,\Delta\boldsymbol{\Theta}_k^T,\mathbf{u}_k^T]\mathbf{W}_k[\Delta\mathbf{p}_k,\Delta\boldsymbol{\Theta}_k^T,\mathbf{u}_k^T]^T + [\Delta\mathbf{p}_N,\Delta\boldsymbol{\Theta}_N^T]\mathbf{W}_N[\Delta\mathbf{p}_N,\Delta\boldsymbol{\Theta}_N^T]^T \tag{45}$$

where $\mathbf{W}_k$ is a 12×12 diagonal matrix and $\mathbf{W}_N$ is a 6×6 diagonal matrix.

Note that for the formation with payload configuration, the weighting matrix $\mathbf{W}_k$ is 18×18.

Trajectory Optimization

A new OCP is solved offline with ALTRO [21] to generate a feasible trajectory that will be fed to the NMPC controller as a reference.

TrajectoryOptimization.jl is an open source library implemented in JULIA that implements the ALTRO algorithms. The same models provided to ACADO are used.

The costs are defined by a built-in function and follows the LQR tracking formulation

$$(\mathbf{x}_{N_{TO}}-\mathbf{x}_f)^T\mathbf{Q}_f(\mathbf{x}_{N_{TO}}-\mathbf{x}_f) + \sum_{k=1}^{N_{TO}-1}(\mathbf{x}_k-\mathbf{x}_f)^T\mathbf{Q}(\mathbf{x}_k-\mathbf{x}_f)+\mathbf{u}_k^T\mathbf{R}_u\mathbf{u}_k \tag{46}$$

where $\mathbf{Q}_f$, $\mathbf{Q}$ and $\mathbf{R}_u$ are 13×13 diagonal weighting matrices, associated to states and control, respectively. It is important to note that for the formation with payload configuration, the weighting matrix $\mathbf{R}_u$ is 12×12. This choice of cost function implies that the only reference parameter the cost function requires from the user is the terminal desired state $\mathbf{x}_f$.

Several constraints types are considered in this approach:

Goal constraints,

$$\mathbf{x}_{N_{TO}} = a \tag{47}$$

State bound constraints,

$$\min \leq \mathbf{x}_k \leq \max, \quad k = 1, \dots, N_{TO} \tag{48}$$

Control bound constraints,

$$\min \leq \mathbf{u}_k \leq \max, \quad k = 1, \dots, N_{TO} - 1 \tag{49}$$

Sphere constraints,

$$(x_k - x_c)^2 + (y_k - y_c)^2 + (z_k - z_c)^2 \geq r^2, \quad k = 1, \dots, N_{TO} \tag{50}$$

The resulting objective function is the sum of all cost functions and Lagrangian terms originating from the constraints.

The optimal trajectory $\mathbf{x}^{opt} \in \mathbb{R}^{13 \times N_{TO}}$ achieved by this method is saved in a *.txt* file. The NMPC controller will load this trajectory into an *array* to serve as references.

Simulation Results

The RotorS simulator is used to perform the MRAV simulation, which is implemented within Gazebo, resulting in the simulation environment depicted in figure 7. All results were obtained in a laptop computer with an Intel Core i7-4710HQ @ $2.50GHz$.

For brevity, Space CoBot is henceforth abbreviated to SC.

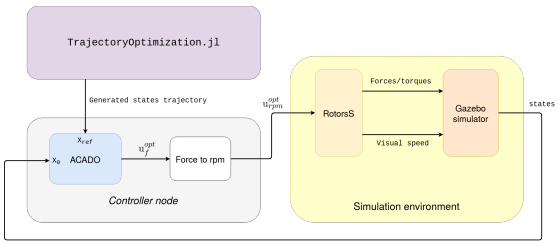

Figure 7: Complete simulation environment scheme.

Trajectory Optimization

To evaluated the NMPC controller different trajectories were generated for each configuration:

1) Single SC single obstacle fly to goal trajectory

2) Single SC single obstacle with payload fly to goal trajectory

3) Single SC two obstacles fly to goal trajectory

4) Formation flight with single obstacle fly to payload and transport it back to start position trajectory

Table 1 presents the solution computation time by ALTRO for the different problems. Evidently, increasingly constrained problems present significant increases in computation time to satisfy similar feasibility conditions. It is important to note that Table 1 presents only the transport phase of the Formation trajectory.

Table 1: Trajectory optimization computation time.

Trajectory	ALTRO
Unconstrained	66 s
State/control bounds	739 s
One obstacle	892 s
One obstacle w/payload	2903 s
Two obstacles	2732 s
Formation w/ payload	2601 s

Table 2: Average tracking errors.

Trajectory	$\|\mathbf{p}\|_e$	$\|\Delta\mathbf{\Theta}\|_e$
SC-Obstacle	$3.0e^{-2}$	$1.5e^{-3}$
SC w/ payload-Obstacle	$1.1e^{-1}$	$2.0e^{-2}$
SC-Two obstacles	$5.0e^{-2}$	$2.4e^{-3}$
Formation SC1-Obstacle	$4.1e^{-2}$	$1.1e^{-2}$
Formation SC2-Obstacle	$3.9e^{-2}$	$1.1e^{-2}$

Trajectory execution

Figures 8, 9, 10 and 11 present a representation of the real-time execution of each trajectory.

The execution of trajectories involving a single SC and a single obstacle in Figures 8 and 9 demonstrate a very good tracking performance. Observing Figure 12, the feedback computation time at each time step, fd, is not very sparse presenting similar averages of $\bar{fd} = 1.8\,ms$ for both experiments. This consistency allows the execution to be smooth with relatively small tracking errors, as observed in Table 2. An increase in attitude error is also visible for the SC with payload. This can be viewed as an attempt from the controller to minimize position tracking error by sacrificing attitude tracking error, given that the higher system mass implies a slower acceleration. In addition, the presence of the payload makes the attitude control more difficult, which introduces errors in position as well.

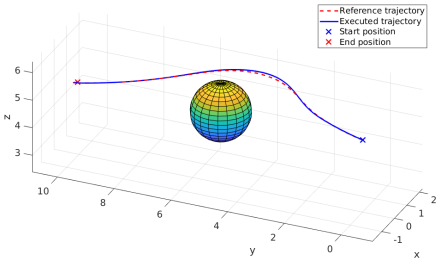

Figure 8: 3D visualization of single obstacle trajectory execution.

The single SC with two obstacles trajectory execution visible in Figure 10 still presents a good tracking performance with a small loss when flying

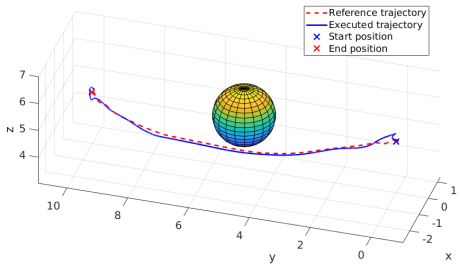

Figure 9: 3D visualization of single obstacle single SC with payload trajectory execution.

between the obstacles. Here the opposite to the previous experiments happens. The controller appears to sacrifice position tracking for attitude tracking as visible in Table 2. This loss is not a result of fd values as they present a similar distribution to previous experiments as can be observed in Figure 12.

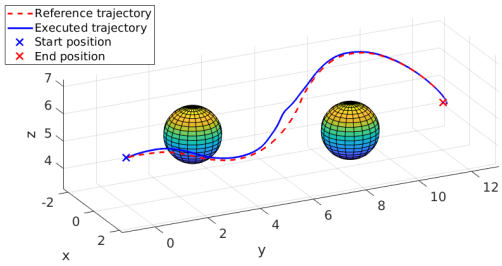

Figure 10: 3D visualization of two obstacles single Space CoBot trajectory execution.

Lastly, the execution of the formation trajectory also presents a reasonable smooth execution. Both vehicles present similar tracking errors as visible in Table 2. However, this comes at the cost of significantly higher control inputs. As mentioned before, control bounds (49) are applied to the trajectory generation problem. This execution greatly exceeds the expected control bounds.

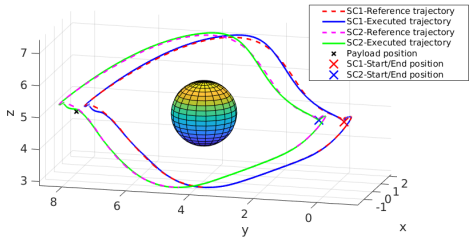

Figure 11: 3D visualization of full formation trajectory execution.

Observing figure 12, the reason becomes apparent as fd not only present a sparse distribution but also a significantly higher $\bar{fd} = 12.5\,ms$. This causes delays in tracking that are compensated by more aggressive corrections.

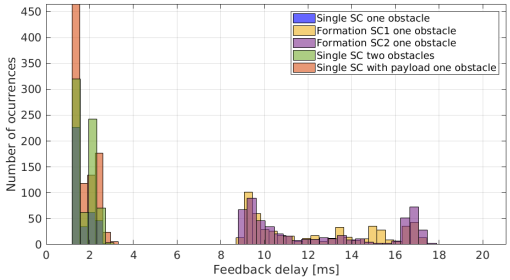

Figure 12: fd distribution for different configurations.

Conclusions

This work presents a study regarding the performance of a NMPC controller for the Space CoBot robot and its possible applications. This was achieved by testing the limits of the real-time formulation to the fullest while trying to perform Control and Guidance tasks in a single formulation, which proved to be unreliable.

The biggest limitation remains the computation time required to obtain control solutions. To mitigate this, a trajectory optimization method was integrated, which allowed better and more complete solutions. However, this method also presents a severe delay in computation and had to be considered as an offline step.

The tests performed throughout this work showing the performance of the NMPC controller define its apparent limitations. Hence, proving that under less complex scenarios the controller should also present a good performance.

Achievements

A good exploration of the NMPC limits was conducted to obtain a controller capable of operating in a real-time frame very close to the microsecond range.

Extensive experimentation was performed to obtain good configurations for different scenarios and different missions. This allows a more detailed model parameterization for each considered configuration. This information can be used to formulate a single model that can be interacted with through a few parameters in order to better adjust to different situations. This means that the same controller can operate with different models without recurring to weighting costs tuning or recompiling any code.

Integrated a trajectory optimization method to take on the Guidance task, that generates feasible trajectories in highly constrained environments.

Future work

The present work can benefit from several further developments to its parts. Experimentation with different error functions could yield interesting results. Better approximations for the model parameters such as K_1 and K_2 can be achieved. Different control architectures, such as a faster NMPC control within a much slower NMPC for trajectory generation as in [16], to allow an increase of the prediction horizon and more importantly, the elimination of the dependence on offline trajectory generation.

Obstacle identification methods would significantly improve the functionalities of the NMPC. For static obstacles and general mapping integrating Octomaps [35] or a similar package would provide means of identifying bounds and obstacles to constrain the trajectory optimization generation. Other vision methods can also be studied to identify additional obstacles that may not be stationary to increase navigation awareness.

Integrating other developing projects such as a robotic arm or similar to provide means of interaction with the surrounding environment would allow more realistic simulation environments.

It would be interesting to perform a study on the energy spent by this NMPC controller when compared to other control methods, as a NMPC might provide close to optimal control where energy usage is minimized, but requires significantly higher computation power.

References

[1] Haiyang Chao, Yongcan Cao, and Yangquan Chen. Autopilots for small unmanned aerial vehicles: A survey. *International Journal of Control, Automation and Systems*, 8(1):36–44, 2010.

[2] Swati Mohan, Alvar Saenz-Otero, Simon Nolet, David W. Miller, and Steven Sell. SPHERES flight operations testing and execution. *Acta Astronautica*, 65(7-8):1121–1132, 2009.

[3] Maria Bualat, Jonathan Barlow, Terrence Fong, Christopher Provencher, Trey Smith, and Allison Zuniga. Astrobee: Developing a free-flying robot for the international space station. *AIAA SPACE 2015 Conference and Exposition*, pages 1–10, 2015.

[4] Pedro Roque and Rodrigo Ventura. Space CoBot: Modular design of an holonomic aerial robot for indoor microgravity environments. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:4383–4390, 2016.

[5] Pedro Roque and Rodrigo Ventura. A space CoBot for personal assistance in space stations. *IJCAI Workshop on Autonomous Mobile Service Robots*, 2016.

[6] Savi Sachdev, Benoit Marcotte, and Graham Gibbs. Canada and the international space station program: Overview and status. *International Astronautical Federation - 55th International Astronautical Congress 2004*, 11(1):7405–7415, 2004.

[7] Shinji Mitani, Masayuki Goto, Ryo Konomura, Yasushi Shoji, Keiji Hagiwara, Shuhei Shigeto, and Nobutaka Tanishima. Int-Ball: Crew-Supportive Autonomous Mobile Camera Robot on ISS/JEM. *IEEE Aerospace Conference Proceedings*, 2019-March:1–15, 2019.

[8] DLR. "CIMON - the intelligent astronaut assistant". *https://www.dlr.de*, 2018.

[9] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. 2011.

[10] Dimitris Kouzoupis, Gianluca Frison, Andrea Zanelli, and Moritz Diehl. Recent Advances in Quadratic Programming Algorithms for Nonlinear Model Predictive Control. *Vietnam Journal of Mathematics*, 46(4):863–882, 2018.

[11] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[12] Wachter Andreas and Lorenz T. Biegler. *Catalogue of an exhibition of the works of Dante Alighieri March to October 1909*, volume 57. 2006.

[13] David Hyunchul Shim, Hoam Chung, H. Jin Kim, and Shankar Sastry. Autonomous exploration in unknown urban environments for unmanned aerial vehicles. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, 8:6381–6388, 2005.

[14] Mina Kamel, Kostas Alexis, Markus Achtelik, and Roland Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on SO(3). *2015 IEEE Conference on Control and Applications, CCA 2015 - Proceedings*, (3):1160–1166, 2015.

[15] Sebastien Gros, Rien Quirynen, and Moritz Diehl. Aircraft control based on fast nonlinear MPC & multiple-shooting. *Proceedings of the IEEE Conference on Decision and Control*, 2012.

[16] Penghong Lin, Songlin Chen, and Chang Liu. Model predictive control-based trajectory planning for quadrotors with state and input constraints. *International Conference on Control, Automation and Systems*, 2016.

[17] Davide Bicego, Jacopo Mazzetto, Ruggero Carli, Marcello Farina, and Antonio Franchi. Nonlinear Model Predictive Control with Actuator Constraints for Multi-Rotor Aerial Vehicles. *arXiv preprint arXiv:1911.08183*, 2019.

[18] William A. Poe and Saeid Mokhatab. *Modeling, Control, and Optimization of Natural Gas Processing Plants*. 2016.

[19] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[20] M Diehl and K Mombaur. *Fast motions in biomechanics and robotics: optimization and feedback control*. 2006.

[21] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. *IEEE International Conference on Intelligent Robots and Systems*, pages 7674–7679, 2019.

[22] David Q. Mayne. Differential Dynamic ProgrammingA Unified Approach to the Optimization of Dynamic Systems. *Control and Dynamic Systems*, 10(C):179–254, 1973.

[23] Siyi Li, Tianbo Liu, Chi Zhang, Dit Yan Yeung, and Shaojie Shen. Learning unmanned aerial vehicle control for autonomous target following. *IJCAI International Joint Conference on Artificial Intelligence*, 2018-July:4936–4942, 2018.

[24] Diego Pardo, Lukas Moller, Michael Neunert, Alexander W. Winkler, and Jonas Buchli. Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning. *IEEE Robotics and Automation Letters*, 1(2):946–953, 2016.

[25] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Astrodynamics Conference, 1986*, (3), 1986.

[26] Christopher G Atkeson, Chae H An, and John M Hollerbach. Estimation of Inertial Parameters of Manipulator Loads and Links Abstract. *The International Journal of Robotics Research*, pages 101–119, 1986.

[27] Sébastien Gros, Mario Zanon, Rien Quirynen, Alberto Bemporad, and Moritz Diehl. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*, (October):1–19, 2016.

[28] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. Efficient numerical methods for nonlinear MPC and moving horizon estimation. *Lecture Notes in Control and Information Sciences*, 384:391–417, 2009.

[29] John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[30] Moritz Diehl, H. Georg Bock, Johannes P. Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[31] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Adjoints estimation methods for impulsive Moon-to-Earth trajectories in the restricted three-body problem. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[32] Hans Joachim Ferreau. *An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control*. PhD thesis, Heidelberg University, 2006.

[33] Boris Houska and Hans Joachim. ACADO Toolkit User ' s Manual, 2011.

[34] Ahmad Bani Younes, Daniele Mortari Prof., James D. Turner, and John L. Junkins Prof. Attitude error kinematics. *Journal of Guidance, Control, and Dynamics*, 37(1):330–335, 2014.

[35] Simon Vanneste, Ben Bellekens, and Maarten Weyn. 3DVFH+: Real-time three-dimensional obstacle avoidance using an octomap. *CEUR Workshop Proceedings*, 1319:91–102, 2014.