JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 46, No. 5, May 2023

Engineering Notes

Fast Model Predictive Control for Spacecraft Rendezvous and Docking with Obstacle Avoidance

Courtney Bashnick* and Steve Ulrich†
Carleton University, Ottawa, Ontario K1S 5B6, Canada

https://doi.org/10.2514/1.G007314

I. Introduction

SPACECRAFT rendezvous and docking (RVD) is a key technology required for on-orbit servicing and space debris removal missions. For the successful execution of such close-proximity operations, a chaser spacecraft must be able to navigate around obstacles, avoid colliding with the target, and react to contingencies to ensure the safety of all spacecraft. Consequently, recent research efforts have focused on real-time, collision-free autonomous RVD (ARVD) guidance algorithms responsible for the path planning of safe, feasible trajectories. In addition to obstacle avoidance capabilities, it is important for the guidance algorithm to minimize the propellant consumption during these operations.

Developing path-planning methods is an art form that involves balancing the complexity and related computational expense of the algorithm with the performance, robustness, and optimality of the solution. Several methods that analytically solve for the trajectory, e.g., glideslope [1], artificial potential functions [2], and Lyapunov vector fields [3], are computationally inexpensive and are therefore real-time implementable, but the proposed solution is nonoptimal. Analytical methods can be modified to increase their optimality (see, e.g., [4,5]), but still achieve suboptimal performance as compared to optimization-based methods [6]. Recently, deep reinforcement learning [7–9] has been applied to spacecraft RVD maneuvers by solving for velocity commands that are tracked by a conventional controller. The trained guidance policy achieves suboptimal, collision-free trajectories and has been shown to be real-time implementable and robust to unmodeled disturbances in experiments on an air-bearing testbed.

The challenge of employing optimization-based methods is their computational expense, which, if too large, limits the real-time implementability of the algorithm. An optimization problem can be formulated as a nonlinear programming (NLP) problem using approaches such as inverse dynamics in the virtual domain [10] or model predictive control (MPC) with nonlinear constraints [11–13], i.e., nonlinear MPC (NMPC). In general, NLP problems offer flexibility in directly handling nonlinear dynamics and nonconvex constraint equations that are inherent to spacecraft RVD maneuvers, but lack convergence guarantees and require complex numerical solvers [14]. Convexifying all elements of the NLP problem, i.e., generating a convex programming problem, gained popularity as a means for real-time, optimal trajectory generation as this class of problems enjoys polynomial-scaling runtime in the problem size and guarantees for globally optimal solutions [14]. In [15–17], the convex programming problem is generated through local linearization of all nonconvex constraints in the original nonlinear problem and is iteratively solved as a second-order cone programming problem. Quadratic programming has also been used to solve for spacecraft RVD under a linear quadratic (LQ) MPC framework to a static [18,19] and rotating target [20,21] with obstacle avoidance. The obstacle constraints, which naturally are nonconvex, must be recast as linear constraint equations for which various methods have been proposed, e.g., rotating hyperplanes [18,19], dual hyperplanes [12], and a direct linearization method [22].

When the guidance problem is posed as an optimization problem, an algorithm or solver must be employed to calculate the solution. A variety of commercial solvers have been used to solve for optimal spacecraft RVD trajectories in the literature: MATLAB's fmincon has been used by [10], MATLAB's quadprog used by [22], Interior Point OPTimizer (IPOPT) [23] used by [12,13,24], MOSEK used by [15–17], CPLEX used by [25], and CVXGEN used by [19]. Authors may also develop custom solvers to solve their optimal control problem (OCP). The so-called Fast MPC was proposed formally by Wang and Boyd [26] and describes an algorithm to create a gradient-based solver employing the infeasible start Newton method that exploits the structure of the LQ-MPC problem for fast computation. Similar algorithms are being implemented on field programmable gate arrays [27], known for their high computational speeds.

In this paper we develop an optimal, real-time Fast MPC guidance algorithm with obstacle avoidance capabilities specific to RVD operations. The performance of the Fast MPC solver is compared to commercially available solvers, IPOPT and MATLAB's fmincon, in two- and three-dimensional simulations. To the best of the authors' knowledge, a comparison of the performance of different solvers for spacecraft rendezvous operations has not been published beyond a brief qualitative comparison of solvers in [28], which focuses on experimental implementation on low computational power hardware. Since the speed of the solver is paramount to the real-time implementability of a guidance algorithm, the computation time of each solver is compared in this work, as well as the total impulse of the calculated solution. The Fast MPC guidance algorithm is validated to work in real-time on a physical testbed. In this context, the original contributions of this work are 1) the development of an optimal, real-time Fast MPC guidance algorithm with moving obstacle avoidance capabilities for ARVD operations, 2) the quantitative comparison of the results and performance obtained by the Fast MPC solver against existing solvers, and 3) experimental validation of the optimal guidance algorithm solved using the Fast MPC solver demonstrating its real-time capabilities.

This paper is organized as follows: Sec. II describes the RVD scenario considered and the OCP to be solved, Sec. III reviews Wang and Boyd's [26] Fast MPC algorithm specifically applied to the spacecraft RVD problem, Sec. IV presents numerical simulations in two and three dimensions, Sec. V presents the experimental results, and Sec. VI concludes this paper.

II. Rendezvous Problem Formulation

This work considers a chaser spacecraft that rendezvous with a target spacecraft in the presence of moving obstacles. In the three-dimensional problem, the chaser rendezvous to a final holding distance modelled as a sphere around the target. In the two-dimensional problem, the chaser additionally docks with the rotating and translating target. To do so, the

Received 24 October 2022; revision received 0 ; accepted for publication 29 January 2023; published online 3 March 2023.

*Ph.D. Student, Department of Mechanical and Aerospace Engineering, 1125 Colonel By Drive.

†Associate Professor, Department of Mechanical and Aerospace Engineering, 1125 Colonel By Drive. Senior Member AIAA.

chaser is guided to a holding position on a dynamic collision avoidance constraint around the target and an entry corridor activates when the chaser nears the target for safe docking. The obstacle avoidance is accomplished with nonlinear ellipsoidal constraints for a nonlinear OCP (NMPC framework), and linearized hyperplane constraints for a linear OCP (LQ-MPC framework). It is assumed that the target and obstacles have constant translational and angular velocities around their center of mass. In experimental validation, the target and obstacle are static.

A. Optimal Control Problem

The OCP to be solved that minimizes the state error of the chaser spacecraft $\boldsymbol{x} \in \mathbb{R}^n$ from its desired state $\boldsymbol{x}_t \in \mathbb{R}^n$ and the commanded control effort $\boldsymbol{u} \in \mathbb{R}^m$ at each iteration over a horizon N starting from the current sampling instant k is written as

$$\text{minimize } J = \left(\boldsymbol{x}^{(k+N)} - \boldsymbol{x}_t^{(k+N)}\right)^T \boldsymbol{Q}_f \left(\boldsymbol{x}^{(k+N)} - \boldsymbol{x}_t^{(k+N)}\right) + (\boldsymbol{u}^{(k)})^T \boldsymbol{R}\boldsymbol{u}^{(k)} + \sum_{\tau=k+1}^{k+N-1} \left(\boldsymbol{x}^{(\tau)} - \boldsymbol{x}_t^{(\tau)}\right)^T \boldsymbol{Q} \left(\boldsymbol{x}^{(\tau)} - \boldsymbol{x}_t^{(\tau)}\right) + (\boldsymbol{u}^{(\tau)})^T \boldsymbol{R}\boldsymbol{u}^{(\tau)} \quad (1a)$$

$$\text{subject to } \boldsymbol{x}^{(\tau+1)} = \boldsymbol{A}\boldsymbol{x}^{(\tau)} + \boldsymbol{B}\boldsymbol{u}^{(\tau)}, \quad \tau = k, \ldots, k+N-1 \quad (1b)$$

$$f_i\left(\boldsymbol{x}^{(\tau)}, \boldsymbol{u}^{(\tau)}\right) \le 0, \quad \tau = k, \ldots, k+N \quad (1c)$$

In Eq. (1a), J is the cost function with state and control weighting matrices $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$, respectively, and $\boldsymbol{Q}_f \in \mathbb{R}^{n\times n}$ is the terminal state weighting matrix and is chosen as the solution of the discrete Riccati equation to ensure local stability [29]. The linear system dynamics in Eq. (1b) are in the form of a discrete-time state space model with the state and control matrices denoted by $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. Equation (1c) quantifies the generalized inequality constraints, given by f_i. Linear inequality constraints that are state-control-separable take the form

$$\begin{cases} \boldsymbol{F}_x^{(\tau)} \boldsymbol{x}^{(\tau)} \le \boldsymbol{f}_x^{(\tau)}, \tau = k+1, \ldots, k+N \\ \boldsymbol{F}_u^{(\tau)} \boldsymbol{u}^{(\tau)} \le \boldsymbol{f}_u^{(\tau)}, \tau = k, \ldots, k+N-1 \end{cases} \quad (2)$$

with problem data $\boldsymbol{F}_x \in \mathbb{R}^{l_x \times n}$, $\boldsymbol{F}_u \in \mathbb{R}^{l_u \times m}$, $\boldsymbol{f}_x \in \mathbb{R}^{l_x}$, and $\boldsymbol{f}_u \in \mathbb{R}^{l_u}$, where l_x and l_u specify the number of state and control constraints, respectively. The philosophy of this MPC design is to, repetitively, recompute the solution every sampling instant in which only the commanded control effort at the current sampling instant $\boldsymbol{u}^{(k)}$ is applied. At the next sampling instant, the solution is recomputed using the current state as the initial condition.

B. Model

Assuming a small separation distance and a target spacecraft on a circular orbit, Clohessy–Wiltshire (CW) differential equations [30] are used as the relative motion model for three-dimensional simulations,

$$\begin{cases} \ddot{x} - 3n^2 x - 2n\dot{y} = \dfrac{u_x}{m} \\ \ddot{y} + 2n\dot{x} = \dfrac{u_y}{m} \\ \ddot{z} + n^2 z = \dfrac{u_z}{m} \end{cases} \quad (3)$$

where x, y, and z are the radial, in-track, and cross-track separations of the spacecraft; $u_{()}$ is the control force in the specified direction; m is the chaser spacecraft mass; and $n = \sqrt{\mu_\oplus / a^3}$ is the mean motion of the target spacecraft, where $\mu_\oplus = 3.986 \times 10^5$ km^3/s^2 is the gravitational parameter of the Earth and a is the semimajor axis of the target's orbit.

As is common in short timescale and separation distance proximity operations research, orbital effects are ignored and the CW equations can further be simplified to planar dynamics with three translational degrees of freedom (DOF) [6–8,12,13,24]. The translational DOF in z is replaced by a DOF to rotate about the z axis to give the relative motion model in two dimensions:

$$\begin{cases} \ddot{x} = \dfrac{u_x}{m} \\ \ddot{y} = \dfrac{u_y}{m} \\ \ddot{\theta} = \dfrac{\tau}{I_z} \end{cases} \quad (4)$$

where I_z is the moment of inertia about the z axis and τ is the control torque.

The continuous-time models in Eqs. (3) and (4) are discretized using a zero-order hold to be written in the form of a discrete-time state-space equation as given by Eq. (1b).

C. Inequality Constraints

The following operational and path constraints are considered in the OCP to be solved.

1. *Maximum Thrust Limitations*

The thrusters onboard the chaser spacecraft are limited in the amount of thrust they can provide. At each step within the horizon, there is a maximum thrust $u_{\max}$ that applies to all directions, that is,

$$\left|u_x^{(k)}\right|, \left|u_y^{(k)}\right|, \quad \text{and} \quad \left|u_z^{(k)}\right| \le u_{\max} \quad (5)$$

A similar maximum torque $\tau_{\max}$ limitation is applied in the two-dimensional case.

2. *Obstacle Avoidance*

As shown in Fig. 1, an ellipsoidal [11] keep-out-zone (KOZ) is drawn around the obstacle in two and three dimensions that identifies

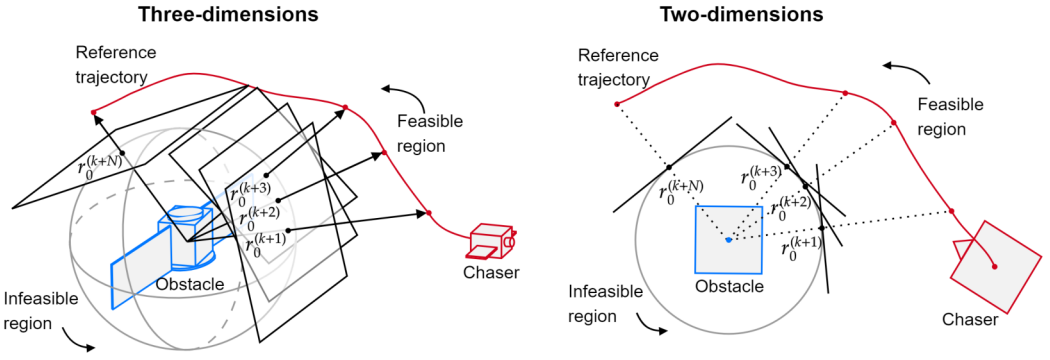


Fig. 1 An ellipsoidal keep-out-zone can be linearized using direct linearization.

the region the chaser spacecraft must remain outside of to safely avoid collisions.

This nonlinear obstacle avoidance constraint used in the NMPC framework is given by

$$1 \leq \left(\boldsymbol{r}^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)}\right)^T \boldsymbol{S}\left(\boldsymbol{r}^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)}\right) \tag{6}$$

where $\boldsymbol{r}_{\text{obj}}^{(k)}$ is the obstacle position, $\boldsymbol{r}^{(k)}$ is the position of the chaser, and the ellipsoid with axes a, b, and c has a shape matrix defined by $\boldsymbol{S} = \text{diag}(a^{-2}, b^{-2}, c^{-2})$.

Direct linearization [22] convexifies the ellipsoidal obstacle constraint by taking the first-order Taylor series expansion of the nonlinear constraint term around the point on the boundary of the KOZ between the obstacle's center and the position of the chaser, denoted by $\boldsymbol{r}_0^{(k)}$, as shown in Fig. 1. Here, the reference trajectory of the chaser from the previous iteration is used to determine the expansion points, $\boldsymbol{r}_0^{(k)}$, over the horizon. Defining the right-hand side of Eq. (6) as f, the linearized version of the obstacle constraint can be written as

$$1 \leq f\left(\boldsymbol{r}_0^{(k)}\right) + \left.\frac{\partial f}{\partial \boldsymbol{r}^{(k)}}\right|_{\boldsymbol{r}_0^{(k)}} \left(\boldsymbol{r}^{(k)} - \boldsymbol{r}_0^{(k)}\right) \tag{7}$$

The authors note a mistake in this formula in [22], where the last term is written with $\boldsymbol{r}_{\text{obj}}$ instead of $\boldsymbol{r}_0$. Evaluating the terms of Eq. (7) with the defined f function, the linearized obstacle constraint equation used in the LQ-MPC framework is

$$1 \leq 2\left(\boldsymbol{r}_0^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)}\right)^T \boldsymbol{S}\left(\boldsymbol{r}^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)}\right) - \left(\boldsymbol{r}_0^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)}\right)^T \boldsymbol{S}\left(\boldsymbol{r}_0^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)}\right) \tag{8}$$

3. *Dynamic Target Collision Constraint*

A dynamic holding radius [13] as depicted by the solid gray circle in Fig. 2 is implemented as a collision avoidance constraint with radius r_{hold} centered at the target spacecraft position $\boldsymbol{r}_{\text{tar}}^{(k)}$,

$$\left(r_{\text{hold}}^{(k)}\right)^2 \leq \left(\boldsymbol{r}^{(k)} - \boldsymbol{r}_{\text{tar}}^{(k)}\right)^T \left(\boldsymbol{r}^{(k)} - \boldsymbol{r}_{\text{tar}}^{(k)}\right) \tag{9}$$

and can be linearized to the form of Eq. (8). The desired holding position of the chaser spacecraft $\boldsymbol{x}_t^{(k)}$, as referenced in Eq. (1a), is defined to be on the holding radius directly in front of the target's docking port, as shown in Fig. 2.

The holding radius is a set value at the beginning of the rendezvous, $r_{\text{hold},0}$, and decreases throughout the mission based on two conditions [13]: first, the chaser spacecraft is within a certain distance η from the holding position $\boldsymbol{x}_t^{(k)}$; second, the chaser is within a certain attitude error ζ of the docking port. The condition on the attitude of the chaser ensures that there is precise attitude pointing, and the slow decrease in holding radius ensures that there is soft docking. If the position and attitude error conditions are met, then the holding radius at the next sampling instant is reduced by a factor of $\gamma \in (0, 1)$, i.e., $r_{\text{hold}}^{(k+1)} = \gamma r_{\text{hold}}^{(k)}$. The holding radius reduces in size until it reaches the lower limit $r_{\text{hold,min}}$, which is defined as the distance between the center of mass of the target and chaser platforms in the docking position, depicted by the dotted circle in Fig. 2.

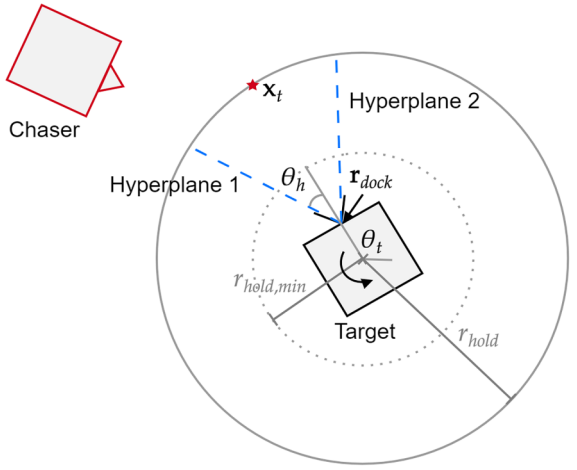


Fig. 2 Desired position on the dynamic holding radius and entry cone constraints.

4. *Entry Cone*

An entry corridor [12,13] that guides the chaser to the docking port is activated once the chaser spacecraft is within a certain distance r_{cone} from the target. The entry cone shown in Fig. 2 consists of two hyperplanes that intersect at the base of the target's docking port and create a triangular feasible region with a half angle θ_h. The constraint for each hyperplane is given by

$$\begin{cases} -\hat{\boldsymbol{n}}_{c1}^{(k)} \cdot \boldsymbol{r}^{(k)} \leq -\hat{\boldsymbol{n}}_{c1}^{(k)} \cdot \boldsymbol{r}_{\text{dock}}^{(k)} \\ \hat{\boldsymbol{n}}_{c2}^{(k)} \cdot \boldsymbol{r}^{(k)} \leq \hat{\boldsymbol{n}}_{c2}^{(k)} \cdot \boldsymbol{r}_{\text{dock}}^{(k)} \end{cases} \tag{10}$$

where $\boldsymbol{r}_{\text{dock}}^{(k)}$ is the position of the docking port and $\hat{\boldsymbol{n}}^{(k)}$ is the hyperplane normal vector given by $\hat{\boldsymbol{n}}_{c1}^{(k)} = [\sin(\theta_t^{(k)} + \theta_h), -\cos(\theta_t^{(k)} + \theta_h)]$ and $\hat{\boldsymbol{n}}_{c2}^{(k)} = [\sin(\theta_t^{(k)} - \theta_h), -\cos(\theta_t^{(k)} - \theta_h)]$.

III. Fast MPC

This section reviews Wang and Boyd's [26] Fast MPC algorithm, which is used to solve a convex optimization problem using a gradient-based method called the infeasible start Newton method.

A. Compact Problem Formulation

The OCP given in Eq. (1) is considered convex when the inequality constraints are linear, such as those given by Eq. (2). It is convenient to write such a problem in a compact form by defining the overall optimization column vector $\boldsymbol{z} \in \mathbb{R}^{n_z}$, $n_z = (n+m)N$, that contains the states and controls, $\boldsymbol{x}$ and $\boldsymbol{u}$, of the system at each sampling instant over the horizon, $\boldsymbol{z} \triangleq [\boldsymbol{u}^{(k)}\ \boldsymbol{x}^{(k+1)}\ \boldsymbol{u}^{(k+1)}\ \ldots\ \boldsymbol{x}^{(k+N-1)}\ \boldsymbol{u}^{(k+N-1)}\ \boldsymbol{x}^{(k+N)}]^T$. The target vector $\boldsymbol{z}_t \in \mathbb{R}^{n_z}$ is similarly defined to contain information about the desired states of the chaser spacecraft over the horizon, $\boldsymbol{z}_t \triangleq [0\ \boldsymbol{x}_t^{(k+1)}\ 0\ \ldots\ \boldsymbol{x}_t^{(k+N-1)}\ 0\ \boldsymbol{x}_t^{(k+N)}]^T$. Then, the convex OCP based on Eq. (1) can be written compactly as

$$\begin{aligned} &\text{minimize} && J(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{z}_t)^T \boldsymbol{H}(\boldsymbol{z} - \boldsymbol{z}_t) \\ &\text{subject to} && \boldsymbol{C}\boldsymbol{z} = \boldsymbol{b} \\ & && \boldsymbol{P}\boldsymbol{z} \leq \boldsymbol{h} \end{aligned} \tag{11}$$

where the matrices $\boldsymbol{H} \in \mathbb{R}^{n_z \times n_z}$, $\boldsymbol{C} \in \mathbb{R}^{n_{h_i} \times n_z}$, and $\boldsymbol{P} \in \mathbb{R}^{n_{f_i} \times n_z}$ are structured along the diagonal with $n_{h_i} = nN$ and $n_{f_i} = (l_x + l_u)N$. The column vectors $\boldsymbol{b} \in \mathbb{R}^{n_{h_i}}$ and $\boldsymbol{h} \in \mathbb{R}^{n_{f_i}}$ are defined from the appropriate constraint equations.

The thrust constraints in Eq. (5) can be written compactly by defining a matrix $\boldsymbol{U} \in \mathbb{R}^{mN \times n_z}$ with the identity matrix repeated along the diagonal and a column vector $\boldsymbol{k} \in \mathbb{R}^{mN}$ built from the maximum thrust and torque limits. Then, Eq. (5) can be written as $\pm \boldsymbol{U}\boldsymbol{z} \leq \boldsymbol{k}$, Similarly, the obstacle avoidance constraints can be written compactly by defining $\boldsymbol{E}^{(k)} \triangleq (\boldsymbol{r}_0^{(k)} - \boldsymbol{r}_{\text{obj}}^{(k)})^T \boldsymbol{S}$ such that Eq. (8) becomes

$$-2\boldsymbol{E}^{(k)}\boldsymbol{r}^{(k)} \leq -1 - \boldsymbol{E}^{(k)}\left(\boldsymbol{r}_0^{(k)} + \boldsymbol{r}_{\text{obj}}^{(k)}\right) \tag{12}$$

A matrix $\boldsymbol{D} \in \mathbb{R}^{N \times n_z}$ is built from the blocks of $\boldsymbol{E}^{(k)}$ to apply this constraint to each step in the horizon, and column vectors $\bar{\boldsymbol{r}}_0 \in \mathbb{R}^{n_z}$ and $\bar{\boldsymbol{r}}_{\text{obj}} \in \mathbb{R}^{n_z}$ are built that contain the expansion points $\boldsymbol{r}_0^{(k)}$ and positions of the obstacle $\boldsymbol{r}_{\text{obj}}^{(k)}$, respectively, over the horizon.

The compact linearized obstacle avoidance constraint is then written as

$$-2\boldsymbol{D}\boldsymbol{z} \le -\mathbf{1}_N - \boldsymbol{D}(\bar{\boldsymbol{r}}_0 + \bar{\boldsymbol{r}}_{\text{obj}}) \tag{13}$$

where $\mathbf{1}_N$ is a column vector of ones with length N. The entry cone constraints in Eq. (10) can be written compactly by defining a matrix $\boldsymbol{G}_{ci} \in \mathbb{R}^{N\times n_z}$ that contains the normals of the hyperplane i over the horizon and a column vector $\bar{\boldsymbol{r}}_{\text{dock}} \in \mathbb{R}^{n_z}$ that contains the position of the apex of the entry cone over the horizon. Then, Eq. (10) can be written as

$$\begin{cases} -\boldsymbol{G}_{c1}\boldsymbol{z} \le -\boldsymbol{G}_{c1}\bar{\boldsymbol{r}}_{\text{dock}} \\ \boldsymbol{G}_{c2}\boldsymbol{z} \le \boldsymbol{G}_{c2}\bar{\boldsymbol{r}}_{\text{dock}} \end{cases} \tag{14}$$

The compact matrices $\pm\boldsymbol{U}$, $-2\boldsymbol{D}$, $-\boldsymbol{G}_{c1}$, and $\boldsymbol{G}_{c2}$ are combined and their rows reordered into a block diagonal structure to form the matrix $\boldsymbol{P}$ of Eq. (11). The corresponding inequality values are similarly combined and reordered to form the column vector $\boldsymbol{h}$.

B. Infeasible Start Newton Method

Interior-point methods solve a quadratic programming problem by reducing it to a sequence of linear equality constrained problems and applying Newton's method. Through expanding the compact cost function and ignoring any terms that do not rely on z, an equivalent cost function for the OCP in Eq. (11) is given by

$$J(z) = z^T\boldsymbol{H}z - 2z_t^T\boldsymbol{H}z \tag{15}$$

The inequality constraints can be written implicitly in the objective function using a logarithmic barrier ϕ,

$$\phi(z) \triangleq -\sum_{i=1}^{n_{f_i}} \log([\boldsymbol{h} - \boldsymbol{P}z]_i) \tag{16}$$

where $[\cdot]_i$ denotes the ith row of the constraint equation matrix. The augmented problem can be posed as

$$\begin{aligned} \text{minimize} \quad & J(z) = z^T\boldsymbol{H}z - 2z_t^T\boldsymbol{H}z - \kappa\sum_{i=1}^{n_{f_i}} \log([\boldsymbol{h} - \boldsymbol{P}z]_i) \\ \text{subject to} \quad & \boldsymbol{C}z = \boldsymbol{b} \end{aligned} \tag{17}$$

where κ is a tuning parameter that gives the weight of the inequality constraints in the cost function. As $\kappa \to 0$, the augmented problem approaches the original problem that was posed in Eq. (11). The Lagrangian L associated with the augmented problem is

$$L(z,\boldsymbol{\nu}) = z^T\boldsymbol{H}z - 2z_t^T\boldsymbol{H}z - \kappa\sum_{i=1}^{n_{f_i}} \log([\boldsymbol{h} - \boldsymbol{P}z]_i) + \boldsymbol{\nu}^T(\boldsymbol{C}z - \boldsymbol{b}) \tag{18}$$

where $\boldsymbol{\nu}$ is the vector of Lagrange multipliers associated with the equality constraints. The optimal solution of the problem, given by $(z^*, \boldsymbol{\nu}^*)$, is found by setting the gradient of the Lagrangian with respect to z equal to zero,

$$2\boldsymbol{H}z^* - 2\boldsymbol{H}z_t + \kappa\boldsymbol{P}^T\boldsymbol{d} + \boldsymbol{C}^T\boldsymbol{\nu}^* = 0 \tag{19}$$

where the vector $\boldsymbol{d} \in \mathbb{R}^{n_{fi}}$ has rows of $d_i = 1/(h_i - \boldsymbol{P}_i z)$, where h_i and $\boldsymbol{P}_i$ are the rows of $\boldsymbol{h}$ and $\boldsymbol{P}$, respectively. The optimality conditions for Eq. (17) are used to define the dual residual $\boldsymbol{r}_d \in \mathbb{R}^{n_z}$ and primal residual $\boldsymbol{r}_p \in \mathbb{R}^{n_{h_i}}$,

$$\begin{cases} \boldsymbol{r}_d \triangleq 2\boldsymbol{H}z - 2\boldsymbol{H}z_t + \kappa\boldsymbol{P}^T\boldsymbol{d} + \boldsymbol{C}^T\boldsymbol{\nu} \\ \boldsymbol{r}_p \triangleq \boldsymbol{C}z - \boldsymbol{b} \end{cases} \tag{20}$$

which equal zero at the optimal solution. The residual $\boldsymbol{r}$ is defined as a stacked variable of the dual and primal residuals, $\boldsymbol{r} \triangleq [\boldsymbol{r}_d \quad \boldsymbol{r}_p]^T$, and is used to find the search directions Δz and $\Delta\boldsymbol{\nu}$ that reduce the augmented cost. To derive an expression for these search directions, called the primal-dual Newton step, the first-order approximation of the residual expanded at the next state is set equal to zero with resulting expression

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{C}^T \\ \boldsymbol{C} & 0 \end{bmatrix}\begin{bmatrix} \Delta z \\ \Delta\boldsymbol{\nu} \end{bmatrix} = -\begin{bmatrix} \boldsymbol{r}_d \\ \boldsymbol{r}_p \end{bmatrix} \tag{21}$$

where $\boldsymbol{\Phi} \triangleq \nabla^2 J = 2\boldsymbol{H} + \kappa\boldsymbol{P}^T\text{diag}(\boldsymbol{d})^2\boldsymbol{P}$ is the Hessian of the augmented cost function.

The search directions can be calculated from the system of linear equations in Eq. (21) using block elimination. By defining the Schur complement $\boldsymbol{Y} \triangleq \boldsymbol{C}\boldsymbol{\Phi}^{-1}\boldsymbol{C}^T \in \mathbb{R}^{n_{h_i}\times n_{h_i}}$ and vector $\boldsymbol{\beta} \triangleq -\boldsymbol{r}_p + \boldsymbol{C}\boldsymbol{\Phi}^{-1}\boldsymbol{r}_d \in \mathbb{R}^{n_{h_i}}$ and rearranging Eq. (21), the search directions $\Delta\boldsymbol{\nu}$ and Δz are solved from

$$\boldsymbol{Y}\Delta\boldsymbol{\nu} = -\boldsymbol{\beta} \tag{22}$$

$$\boldsymbol{\Phi}\Delta z = -\boldsymbol{r}_d - \boldsymbol{C}^T\Delta\boldsymbol{\nu} \tag{23}$$

The authors note a mistake in [26], where Eq. (23) has written $\boldsymbol{\nu}$ instead of $\Delta\boldsymbol{\nu}$.

To determine the step size t that the next solution iteration will take along the search direction, a backtracking line search is used [31]. The step size t is initialized to a value of unity and repeatedly decreased by a value of $\beta \in (0, 1)$ if the inequality constraints are violated at the next solution iteration. The step size is then refined further to decrease the residual of the current solution scaled by a factor of $\alpha \in (0, 1/2)$ smaller.

Finally, the search direction and step size are used to calculate the next solution iteration. The algorithm is repeated until the residual is below a user-defined tolerance ϵ or the maximum number of iterations has been reached. The entire process is repeated for decreasing values of κ, which are used to weigh the inequality constraints in the augmented cost function in Eq. (17). The infeasible start Newton method is summarized in Algorithm 1.

Algorithm 1 Infeasible start Newton method

for $\kappa = \{\kappa_1, \kappa_2, \kappa_3, \ldots\}$ with $\{\kappa_i \in \mathbb{R}^+ | \kappa_{i+1} < \kappa_i\}$ **do**
  repeat
    1) *Compute primal and dual Newton steps*, Δz and $\Delta\boldsymbol{\nu}$
      1.1) Form the Schur complement $\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{\Phi}^{-1}\boldsymbol{C}^T$ and $\boldsymbol{\beta} = -\boldsymbol{r}_p + \boldsymbol{C}\boldsymbol{\Phi}^{-1}\boldsymbol{r}_d$
      1.2) Determine $\Delta\boldsymbol{\nu}$ by solving $\boldsymbol{Y}\Delta\boldsymbol{\nu} = -\boldsymbol{\beta}$
      1.3) Determine Δz by solving $\boldsymbol{\Phi}\Delta z = -\boldsymbol{r}_d - \boldsymbol{C}^T\Delta\boldsymbol{\nu}$
    2) *Backtracking line search* on $\|\boldsymbol{r}\|_2$; $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$
      $t := 1$
      repeat
        $t := \beta t$
      until all $\boldsymbol{h} - \boldsymbol{P}(z + t\Delta z) > 0$
      while $\|\boldsymbol{r}(z + t\Delta z, \boldsymbol{\nu} + t\Delta\boldsymbol{\nu})\|_2 > (1 - \alpha t)\|\boldsymbol{r}(z, \boldsymbol{\nu})\|_2$ **do**
        $t := \beta t$
    3) *Update* $z := z + t\Delta z$ and $\boldsymbol{\nu} := \boldsymbol{\nu} + t\Delta\boldsymbol{\nu}$
  until ($\|\boldsymbol{r}(z, \boldsymbol{\nu})\|_2 \le \epsilon$ and $\boldsymbol{C}z = \boldsymbol{b}$) or *maximum iterations exceeded*
end

C. Fast Computation of the Primal-Dual Newton Step

As explained in [26], through exploiting the block diagonal structure of the matrices in the problem, step 1 of Algorithm 1 can be computed quickly. First, it is recognized that the inverse matrix $\boldsymbol{\Phi}^{-1}$ is block diagonal with submatrices $\tilde{\boldsymbol{Q}}^{(j)}$ and $\tilde{\boldsymbol{R}}^{(j)}$:

$$\boldsymbol{\Phi}^{-1} = \text{diag}(\tilde{\boldsymbol{R}}^{(0)}, \tilde{\boldsymbol{Q}}^{(1)}, \tilde{\boldsymbol{R}}^{(1)}, \ldots, \tilde{\boldsymbol{Q}}^{(N-1)}, \tilde{\boldsymbol{R}}^{(N-1)}, \tilde{\boldsymbol{Q}}_f) \tag{24}$$

The matrix $\boldsymbol{\Phi}$ has a similar form, but with submatrices denoted by $\bar{\boldsymbol{Q}}^{(j)}$ and $\bar{\boldsymbol{R}}^{(j)}$. Next, the Cholesky factorization of each submatrix within $\boldsymbol{\Phi}$ is computed, i.e., $\bar{\boldsymbol{R}}^{(j)} = \boldsymbol{L}_{\bar{\boldsymbol{R}}_j}\boldsymbol{L}_{\bar{\boldsymbol{R}}_j}^T$ and $\bar{\boldsymbol{Q}}^{(j)} = \boldsymbol{L}_{\bar{\boldsymbol{Q}}_j}\boldsymbol{L}_{\bar{\boldsymbol{Q}}_j}^T$. Then, forward and backward substitution is used within the expressions of the submatrices $\boldsymbol{Y}_{ij} \in \mathbb{R}^{n\times n}$ of $\boldsymbol{Y}$,

$$\begin{cases} \boldsymbol{Y}_{11} = \boldsymbol{B}_d\tilde{\boldsymbol{R}}^{(0)}\boldsymbol{B}_d^T + \tilde{\boldsymbol{Q}}^{(1)} \\ \boldsymbol{Y}_{ii} = \boldsymbol{A}_d\tilde{\boldsymbol{Q}}^{(i-1)}\boldsymbol{A}_d^T + \boldsymbol{B}_d\tilde{\boldsymbol{R}}^{(i-1)}\boldsymbol{B}_d^T + \tilde{\boldsymbol{Q}}^{(i)} \\ \boldsymbol{Y}_{i,i+1} = \boldsymbol{Y}_{i+1,i}^T = -\tilde{\boldsymbol{Q}}^{(i)}\boldsymbol{A}_d^T \end{cases} \tag{25}$$

which are then used to build the matrix $\boldsymbol{Y}$,

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{Y}_{11} & \boldsymbol{Y}_{12} & 0 & \dots & 0 & 0 \\ \boldsymbol{Y}_{21} & \boldsymbol{Y}_{22} & \boldsymbol{Y}_{23} & \dots & 0 & 0 \\ 0 & \boldsymbol{Y}_{32} & \boldsymbol{Y}_{33} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \boldsymbol{Y}_{N-1,N-1} & \boldsymbol{Y}_{N-1,N} \\ 0 & 0 & 0 & \dots & \boldsymbol{Y}_{N,N-1} & \boldsymbol{Y}_{NN} \end{bmatrix} \tag{26}$$

without the need for direct inversion of the matrix $\boldsymbol{\Phi}$. In using the expressions of Eq. (25), it is necessary to exploit the fact that each submatrix within $\boldsymbol{\Phi}$ is the inverse of the corresponding submatrix in $\boldsymbol{\Phi}^{-1}$, i.e., $\tilde{\boldsymbol{R}}^{(j)} = (\bar{\boldsymbol{R}}^{(j)})^{-1} = (\boldsymbol{L}_{\bar{\boldsymbol{R}}_j}\boldsymbol{L}_{\bar{\boldsymbol{R}}_j}^T)^{-1}$.

Once $\boldsymbol{Y}$ has been formed, it must be factored via Cholesky factorization, i.e., $\boldsymbol{Y} = \boldsymbol{L}\boldsymbol{L}^T$. Then, the dual step $\Delta\boldsymbol{\nu}$ can be calculated from Eq. (22) via forward and backward substitution. The primal step Δz can be directly calculated from Eq. (23) since the inverse matrix $\boldsymbol{\Phi}^{-1}$, Eq. (24), can be built from its subblocks that are computed using forward and backward substitution during the construction of the Schur complement $\boldsymbol{Y}$.

D. Additional Variations

As suggested in [26], Algorithm 1 can be computed much faster through two simple variations: first, the number of loops for κ is limited to decrease the number of Newton iterations required to solve the problem and to improve warm starting from the previous solution; second, the maximum number of iterations is significantly limited. As a result, the solution may not be primal feasible, but the inequality constraints are satisfied. Since only the control force for the first sampling instant within the horizon is applied and the rest of the plan is not used, it is not imperative that the plan be perfectly computed.

In this work, Algorithm 1 is used with a fixed κ value and a maximum iteration count of 10. There are two variations on this solver: The first, referred to as ISNM, computes the search directions directly from Eqs. (22) and (23) using the MATLAB backslash notation, \, to invert the matrices. The second, referred to as Fast MPC, computes the search directions following the approach outlined in Sec. III.C, whereby the inversion of matrices is avoided through use of Cholesky factorization and forward and backward substitution.

IV. Simulation Results

This section presents the simulations performed, whereby the nonlinear OCP is solved using IPOPT and the linear OCP is solved using IPOPT, fmincon, ISNM, and Fast MPC. Simulations were executed using MATLAB on a Quad Core 1.99 GHz Intel Core i7 64-bit CPU with 16 GB RAM.

A. Three-Dimensional Simulations

The simulation considers a large chaser spacecraft ($m = 6500$ kg) that rendezvous with a target satellite in low Earth orbit (705 km altitude). In the local-vertical–local-horizontal frame, the target spacecraft is located at the origin, so the desired holding position for the chaser spacecraft $\boldsymbol{x}_t^{(k)}$ in the OCP cost function is zero. The rendezvous mission terminates when the chaser reaches the holding distance from the target, chosen as 2.3 m.

The ISNM and Fast MPC solvers are implemented with $\kappa = 0.05$ and parameters for the backtracking line search selected as $\alpha = 0.01$ and $\beta = 0.95$. The solution to the OCP is found when the residual is below the tolerance ϵ of 10^{-9} or the maximum iteration limit of 10 is reached. The two obstacle KOZs are modeled as spheres with radii of 20 and 50 m. The maximum thrust limit $u_{\max}$ is set as 20 N. Additional model parameters are listed in Table 1.

The scenario is modeled for when obstacles are present and absent, as shown in Fig. 3, where the path of the chaser is plotted by the various red lines, the target is depicted by a black circle at the origin, and the obstacles are plotted in blue. This simulation demonstrates the successful moving obstacle avoidance capabilities of the algorithm as solved by each solver. Furthermore, it is clear that the custom solvers successfully calculate a solution that is comparable to the solutions of the commercially available solvers, with minor differences in the trajectories attributed to the different numerical algorithms being employed. The solution of IPOPT NMPC has the lowest total impulse (32.93 kN). The custom solvers find a similar total impulse of 33.80 and 33.98 kN for ISNM and Fast MPC, respectively. The largest total impulse is commanded by fmincon (34.33 kN).

The computation time of the solvers per iteration step is shown in Fig. 4. It is clear that IPOPT is the slowest of the solvers. As expected, the custom solvers are generally the fastest, with averages of 0.019 and 0.021 s for ISNM and Fast MPC, respectively, when the periods of high computation time are excluded. These large increases in computation effort may be attributed to the planning of maneuvers that have a significant change in the control effort, which certainly includes avoiding moving obstacles. In such cases, the algorithm needs to refine its line search significantly to ensure feasibility of the next solution (Algorithm 1, step 2) and in turn requires a longer computation time. When including the high computation periods, the average time for the custom solvers is higher than fmincon, as shown in Fig. 5.

B. Two-Dimensional Simulations

The two-dimensional simulations have model parameters based on the Spacecraft Robotics and Control Laboratory's Spacecraft Proximity Operations Testbed (SPOT) with equations developed in the inertial reference frame with origin fixed to the table. The maximum thrust limit is 0.15 N and the circular obstacle KOZ radius is 0.47 m.

Table 1 Model parameters for the three-dimensional simulations

Parameter		Value
Initial states $(x, y, z, \dot{x}, \dot{y}, \dot{z})$	Chaser, m, m/s	(50, −1000, 10, 0, 0, 0)
	Obstacle 1, m, m/s	(−30, −800, 10, −0.075, −0.005, 0.015)
	Obstacle 2, m, m/s	(−100, 0, −40, 0.07, −0.16, 0.01)
Problem variables	Horizon N	30
	Discretization time T_d, s	15
	$\boldsymbol{Q}$	diag(1, 1, 1, 100, 100, 100)
	$\boldsymbol{R}$	diag(100, 100, 100)

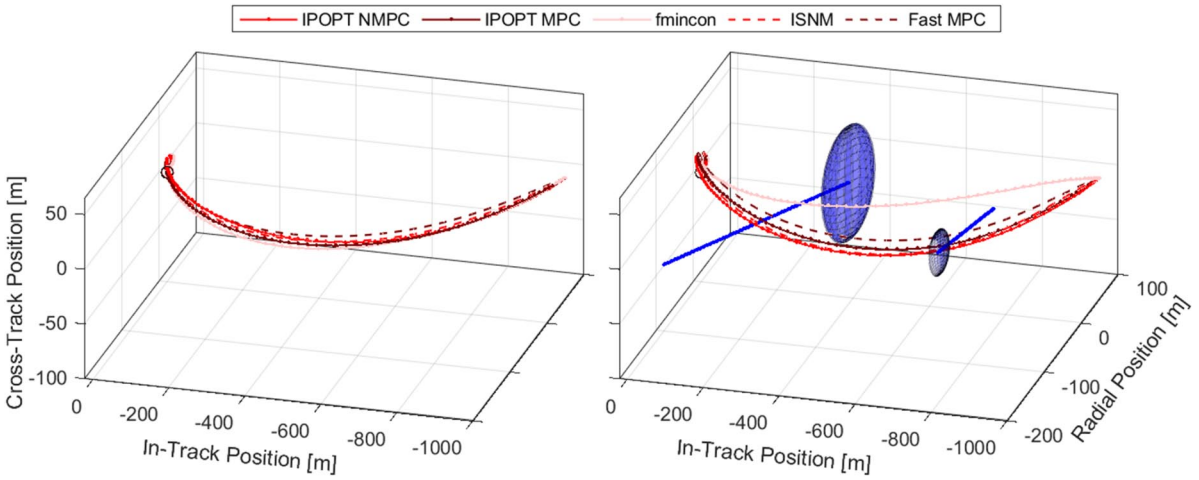


Fig. 3 Trajectories of the solutions for each solver for the three-dimensional scenario without (left) and with (right) obstacles.

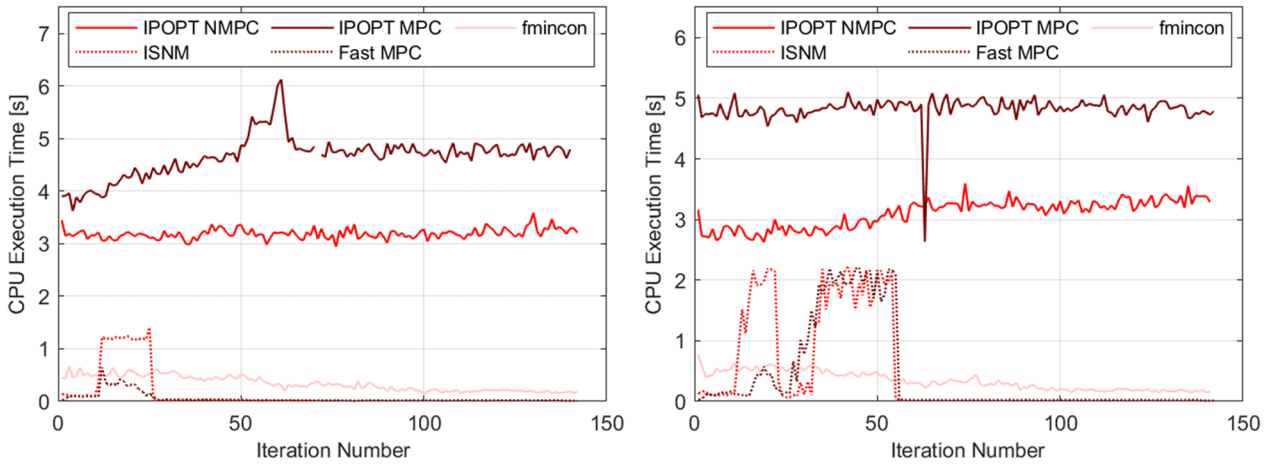


Fig. 4 The computation time of each solver for the three-dimensional simulations.

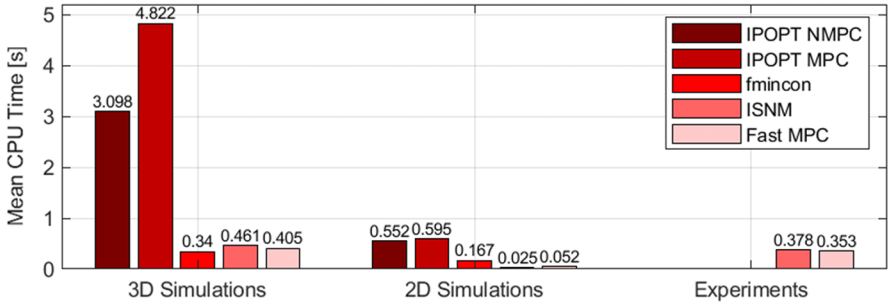


Fig. 5 Average computation time for each solver for the scenarios including obstacles.

The dynamic holding radius around the target spacecraft is initialized to 1.15 m and decreases by a factor of $\gamma = 0.95$ with $\eta = \sqrt{0.5}$ m and $\zeta = 10°$. The entry cone has a $\theta_h = 20°$ half angle and activates when the chaser is $r_{\text{cone}} = 0.75$ m from the docking position. The ISNM and Fast MPC solvers are implemented with the same parameters as the three-dimensional simulations. Additional model parameters, including the initial states given with translational units of m and m/s and rotational units of ° and °/s, are listed in Table 2. Note that the chaser spacecraft always begins at rest, while the target and obstacles have nonzero initial velocities. The rotation rates of the target represent a slow-spinning motion (< 5°/s), yet the chosen rates are faster than previously reported MPC experiments [13].

Two scenarios are modeled, each simulated without and with obstacles, as shown in Figs. 6 and 7 for test cases A and B, respectively. The chaser is depicted by the red platform and its trajectory is plotted in red for each solver. The target and obstacles are plotted in black and blue, respectively. Notably, the docking port of the chaser and target spacecraft is off-axis, which reflects the physical SPOT platforms.

The simulations are depicted at four different snapshots: the first snapshot at the initial condition has the highest transparency, the final snapshot in the docked position is the most bold, and two additional intermediate snapshots show the progress of each platform. The entry cone is depicted by the gray hyperplanes when the constraint is active.

For both test cases with obstacles present, the custom solvers calculate the solution with the lowest total impulse. Unlike in the three-dimensional simulations, the custom solvers do not experience periods of high computation time and therefore have the lowest average computation time, as shown in Fig. 5.

Table 2 Model parameters for the two-dimensional simulations

Parameter		Value	
		Test case A	Test case B
Initial states $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$	Chaser	(3.0, 1.3, 180, 0, 0, 0)	(3.2, 2.2, 90, 0, 0, 0)
	Target	(1.1, 1.3, 270, 0.015, −0.008, 4.0)	(0.2, 0.2, 0, 0.02, 0.01, −2.5)
	Obstacle 1	(1.0, 2.0, 0, 0, −0.005, 0)	(1.3, 1.0, 0, 0.01, −0.01, 0)
	Obstacle 2	(1.0, 0.5, 0, 0.01, 0, 0)	(2.1, 1.0, 0, 0.012, 0.01, 0)
Problem variables	Horizon N	15	
	Discretization time T_d, s	2	
	$\boldsymbol{Q}$	diag(10, 10, 10, 300, 300, 500)	
	$\boldsymbol{R}$	diag(10, 10, 10)	

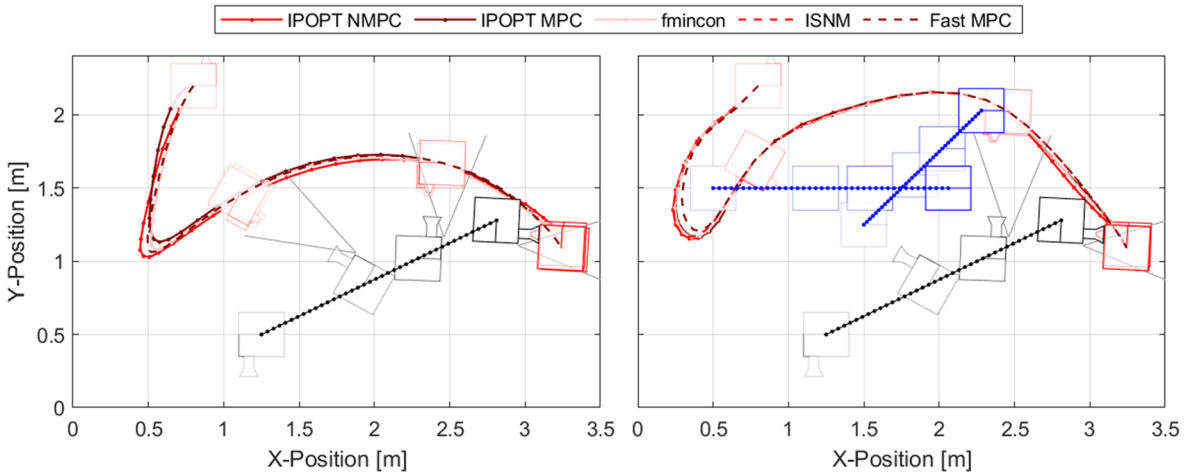


Fig. 6 Trajectories of the solutions for each solver for the two-dimensional scenario test case A without (left) and with (right) obstacles.

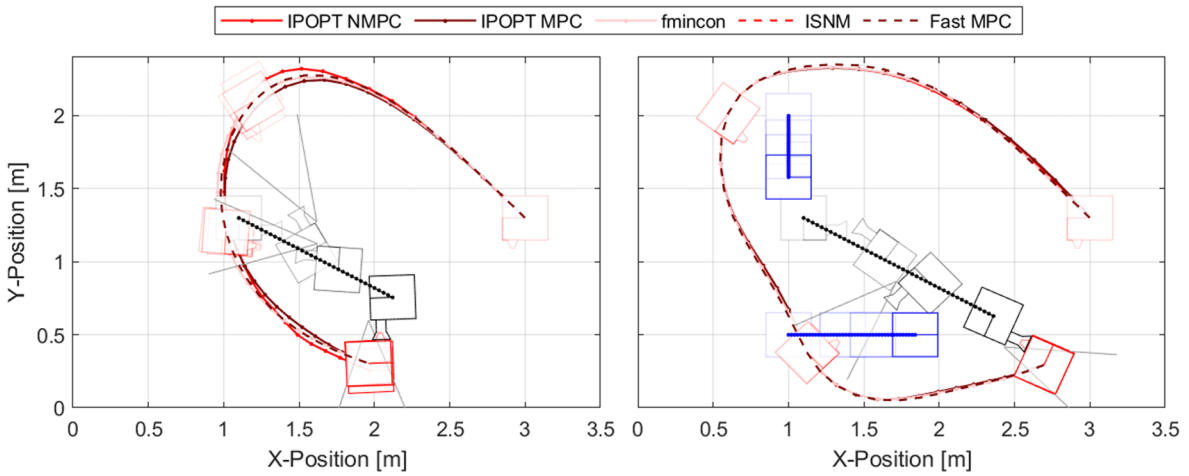


Fig. 7 Trajectories of the solutions for each solver for the two-dimensional scenario test case B without (left) and with (right) obstacles.

V. Experimental Validation

The experiments were conducted using the Spacecraft Robotics and Control Laboratory's SPOT facility, as depicted in Fig. 8, which was recently upgraded to include a third floating platform. The controllers on the platforms are based on a Simulink diagram that interfaces with the experiment hardware. The diagram is converted, compiled, and then executed on a Raspberry Pi 3 computer, which runs a Quad Core 1.2 GHz Broadcom BCM2837 64-bit CPU with 1 GB RAM. Due to limitations of the MATLAB Code Generation process, only the custom solvers can be implemented on the SPOT platforms. Further limitations in the facility infrastructure have the consequence that the guidance algorithm can only be validated with static target and obstacle platforms. The experiments use the same model parameters as the two-dimensional simulations described in Sec. IV.B, with the exception that a horizon of 10 is used, the entry cone half angle is 45°, and the κ value of the Fast MPC solver is modified to 0.1.

The experiment was performed with an initial chaser state of (3 m, 2 m, 0°), target state of (0.4973 m, 0.4919 m, 318.16°), and obstacle state of (1.4897 m, 1.2410 m, 0°). The result of five experimental trials of the ISNM and Fast MPC algorithms is shown in Figs. 9 and 10, respectively, plotted in red for experiments without (left) and with (right) the obstacle platform. The simulated trajectory in each scenario is overlaid the experimental data in dark red, showing good

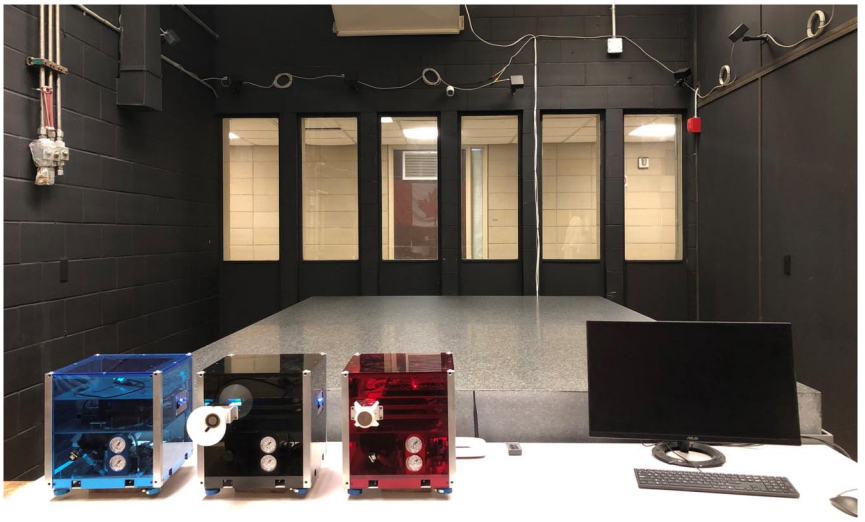


Fig. 8 The spacecraft proximity operations testbed.

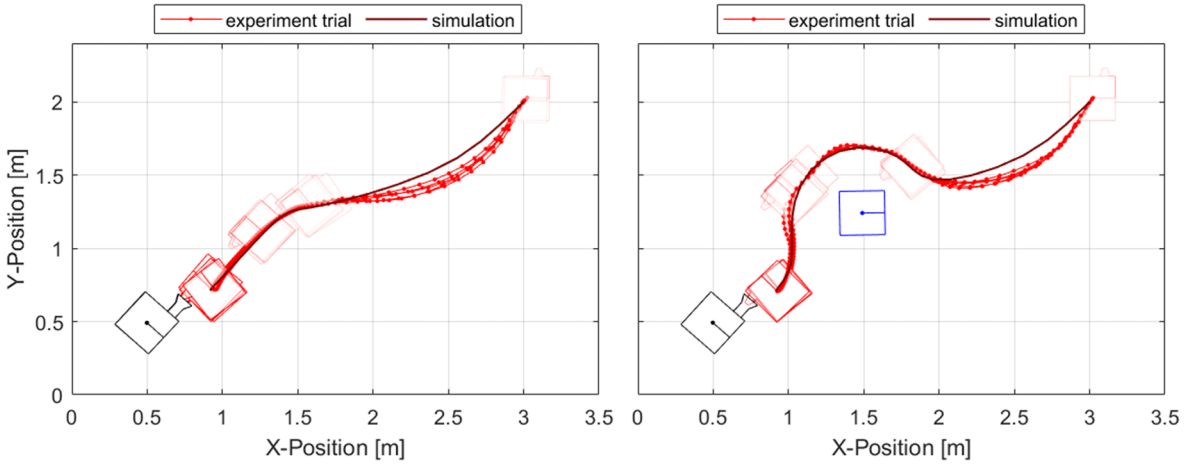


Fig. 9 The experimental results of five trials using ISNM.

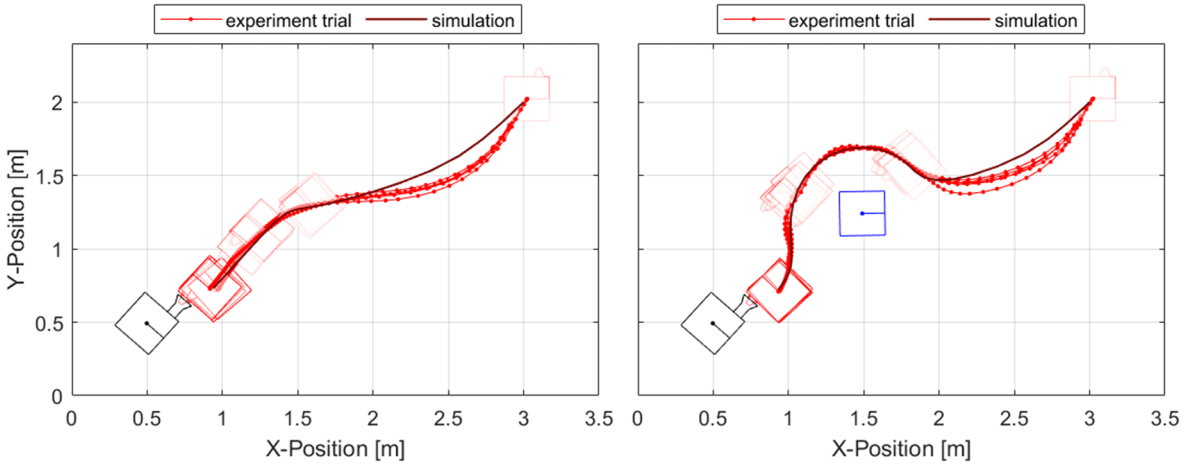
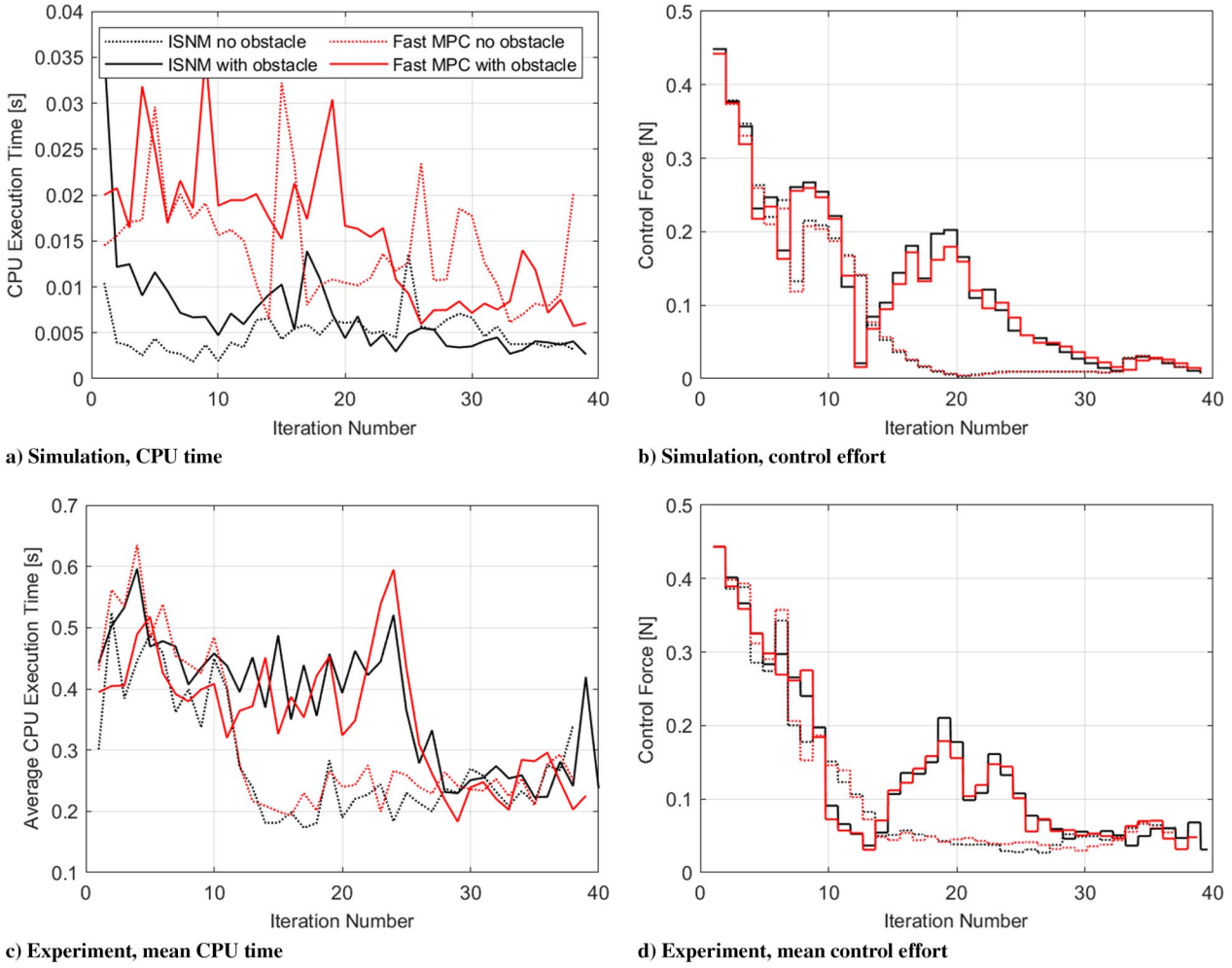


Fig. 10 The experimental results of five trials using Fast MPC.

agreement. The trends in computation time and commanded control effort of the solvers are compared between simulations and experiments (using the average of five trials) in Fig. 11.

The experimental results show that the proposed guidance algorithm solved using the custom solvers is real-time implementable with successful collision avoidance capabilities. The solvers have similar computation times on the Raspberry Pi 3, shown in Fig. 11c. These computation times are 20 to 50 times larger than simulated results due to the different processors used. As shown in Figs. 11b and 11d, the commanded control effort of the solvers agree in both simulations and experiments, and the experimental trends show excellent correlation to simulated expectations. Discrepancies



a) Simulation, CPU time

b) Simulation, control effort

c) Experiment, mean CPU time

d) Experiment, mean control effort

Fig. 11 Comparison of the mean computation time and mean control effort between simulations and experiments.

between the simulated and experimental trajectories could be attributed to a number of factors, including a persistent disturbance caused by a significant leak in the plumbing of the chaser platform.

VI. Conclusions

In this paper a custom gradient-based solver was developed for a real-time, fuel-optimal, collision-free spacecraft RVD guidance algorithm based on the LQ-MPC framework. Simulations in three dimensions considered maximum thrust limitations and moving obstacle avoidance constraints when rendezvousing to a set holding distance from the target. Two-dimensional simulations additionally considered a dynamic holding radius and an entry corridor to facilitate safe docking to a rotating and translating target. The performance of the custom solver was compared to commercially available solvers, IPOPT and fmincon, and additionally a nonlinear MPC problem solved by IPOPT. Results show that the custom solver has the lowest average computation time per iteration, but is susceptible to periods of high computation time when computing maneuvers with a large change in control effort. The real-time implementability of the guidance algorithm solved with the custom solver was validated on an air-bearing testbed in experiments with a static target and obstacle. Future work at the Spacecraft Robotics and Control Laboratory includes modifying the line search algorithm of the custom solver to minimize the periods of high computation time, as well as experimentally validating the custom solver with a rotating/translating target and moving obstacle.

Acknowledgments

This research was financially supported in part by the Natural Sciences and Engineering Research Council of Canada Alexander Graham Bell Canada Graduate Scholarship-Master's award and through the New Technologies for Canadian Observatories Collaborative Research and Training Experience program.

References

[1] Nolet, S., "Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite," Ph.D. Thesis, Massachusetts Inst. of Technology, Cambridge, MA, 2007.

[2] Lopez, I., and Mclnnes, C. R., "Autonomous Rendezvous Using Artificial Potential Function Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 2, 1995, pp. 237–241. https://doi.org/10.2514/3.21375

[3] Hough, J., and Ulrich, S., "Cascaded Lyapunov Vector Fields for Acceleration-Constrained Spacecraft Path Planning," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 11, 2022, pp. 2076–2090. https://doi.org/10.2514/1.G006260

[4] McCamish, S. B., Romano, M., Nolet, S., Edwards, C. M., and Miller, D. W., "Flight Testing of Multiple-Spacecraft Control on SPHERES During Close-Proximity Operations," *Journal of Spacecraft and Rockets*, Vol. 46, No. 6, 2009, pp. 1202–1213. https://doi.org/10.2514/1.43563

[5] Muñoz, J. D., "Rapid Path-Planning Algorithms for Autonomous Proximity Operations of Satellites," Ph.D. Thesis, Univ. of Florida, Gainesville, FL, 2011.

[6] Zappulla, R., Park, H., Virgili-Llop, J., and Romano, M., "Real-Time Autonomous Spacecraft Proximity Maneuvers and Docking Using an Adaptive Artificial Potential Field Approach," *IEEE Transactions on Control Systems Technology*, Vol. 27, No. 6, 2018, pp. 2598–2605. https://doi.org/10.1109/TCST.2018.2866963

[7] Hovell, K., and Ulrich, S., "On Deep Reinforcement Learning for Spacecraft Guidance," *AIAA Scitech 2020 Forum*, AIAA Paper

2020-1600, Jan. 2020.
https://doi.org/10.2514/6.2020-1600

[8] Hovell, K., and Ulrich, S., "Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 254–264.
https://doi.org/10.2514/1.A34838

[9] Hovell, K., and Ulrich, S., "Laboratory Experimentation of Spacecraft Robotic Capture Using Deep-Reinforcement-Learning–Based Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 11, 2022, pp. 2138–2146.
https://doi.org/10.2514/1.G006656

[10] Boyarko, G. A., "Spacecraft Guidance Strategies for Proximity Maneuvering and Close Approach with a Tumbling Object," Ph.D. Thesis, Naval Postgraduate School, Monterey, CA, 2010.

[11] Jewison, C., Erwin, R. S., and Saenz-Otero, A., "Model Predictive Control with Ellipsoid Obstacle Constraints for Spacecraft Rendezvous," *IFAC-PapersOnLine*, Vol. 48, No. 9, 2015, pp. 257–262.
https://doi.org/10.1016/j.ifacol.2015.08.093

[12] Park, H., Zagaris, C., Virgili Llop, J., Zappulla, R., Kolmanovsky, I., and Romano, M., "Analysis and Experimentation of Model Predictive Control for Spacecraft Rendezvous and Proximity Operations with Multiple Obstacle Avoidance," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2016-5273, Sept. 2016.
https://doi.org/10.2514/6.2016-5273

[13] Park, H., Zappulla, R., Zagaris, C., Virgili-Llop, J., and Romano, M., "Nonlinear Model Predictive Control for Spacecraft Rendezvous and Docking with a Rotating Target," *27th AAS/AIAA Spaceflight Mechanics Meeting*, Vol. 2, AAS Paper 17-496, Univelt, San Diego, CA, Feb. 2017.

[14] Malyuta, D., Reynolds, T. P., Szmuk, M., Lew, T., Bonalli, R., Pavone, M., and Acikmese, B., "Convex Optimization for Trajectory Generation," arXiv Preprint, June 2021.
https://doi.org/10.48550/arXiv.2106.09125

[15] Lu, P., and Liu, X., "Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 375–389.
https://doi.org/10.2514/1.58436

[16] Liu, X., and Lu, P., "Robust Trajectory Optimization for Highly Constrained Rendezvous and Proximity Operations," *AIAA Guidance, Navigation, and Control (GNC) Conference*, AIAA Paper 2013-4720, Aug. 2013.
https://doi.org/10.2514/6.2013-4720

[17] Liu, X., and Lu, P., "Solving Nonconvex Optimal Control Problems by Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 750–765.
https://doi.org/10.2514/1.62110.

[18] Petersen, C., Jaunzemis, A., Baldwin, M., Holzinger, M., and Kolmanovsky, I., "Model Predictive Control and Extended Command Governor for Improving Robustness of Relative Motion Guidance and Control," *24th AAS/AIAA Spaceflight Mechanics Meeting*, AAS Paper 14-249, Univelt, San Diego, CA, Jan. 2014, pp. 701–718.

[19] Weiss, A., Baldwin, M., Erwin, R. S., and Kolmanovsky, I., "Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies," *IEEE Transactions on Control Systems Technology*, Vol. 23, No. 4, 2015, pp. 1638–1647.
https://doi.org/10.1109/TCST.2014.2379639

[20] Park, H., Di Cairano, S., and Kolmanovsky, I., "Linear Quadratic Model Predictive Control Approach to Spacecraft Rendezvous and Docking," *Proceedings of 21st AAS/AIAA Space Flight Mechanics Meeting, Spaceflight Mechanics, Part III of Advances in the Astronautical Sciences*, Vol. 140, AAS Paper 11-142, Univelt, Inc., Escondido, CA, Feb. 2011, pp. 565–584.

[21] Di Cairano, S., Park, H., and Kolmanovsky, I., "Model Predictive Control Approach for Guidance of Spacecraft Rendezvous and Proximity Maneuvering," *International Journal of Robust and Nonlinear Control*, Vol. 22, No. 12, 2012, pp. 1398–1427.
https://doi.org/10.1002/rnc.2827

[22] Zagaris, C., Park, H., Virgili-Llop, J., Zappulla, R., Romano, M., and Kolmanovsky, I., "Model Predictive Control of Spacecraft Relative Motion with Convexified Keep-Out-Zone Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 9, 2018, pp. 2054–2062.
https://doi.org/10.2514/1.G003549

[23] Wächter, A., and Biegler, L. T., "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, No. 1, 2006, pp. 25–57.
https://doi.org/10.1007/s10107-004-0559-y

[24] Virgili-Llop, J., Zagaris, C., Park, H., Zappulla, R., and Romano, M., "Experimental Evaluation of Model Predictive Control and Inverse Dynamics Control for Spacecraft Proximity and Docking Maneuvers," *CEAS Space Journal*, Vol. 10, No. 1, 2018, pp. 37–49.
https://doi.org/10.1007/s12567-017-0155-7

[25] Richards, A., Schouwenaars, T., How, J. P., and Feron, E., "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 755–764.
https://doi.org/10.2514/2.4943

[26] Wang, Y., and Boyd, S., "Fast Model Predictive Control Using Online Optimization," *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, 2009, pp. 267–278.
https://doi.org/10.1109/TCST.2009.2017934

[27] Hartley, E. N., and Maciejowski, J. M., "Field Programmable Gate Array Based Predictive Control System for Spacecraft Rendezvous in Elliptical Orbits," *Optimal Control Applications and Methods*, Vol. 36, No. 5, 2015, pp. 585–607.
https://doi.org/10.1002/oca.2117

[28] Mammarella, M., Lorenzen, M., Capello, E., Park, H., Dabbene, F., Guglieri, G., Romano, M., and Allgöwer, F., "An Offline-Sampling SMPC Framework with Application to Autonomous Space Maneuvers," *IEEE Transactions on Control Systems Technology*, Vol. 28, No. 2, 2020, pp. 388–402.
https://doi.org/10.1109/TCST.2018.2879938

[29] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, No. 6, 2000, pp. 789–814.
https://doi.org/10.1016/S0005-1098(99)00214-9

[30] Clohessy, W. H., and Wiltshire, R. S., "Terminal Guidance System for Satellite Rendezvous," *Journal of the Aerospace Sciences*, Vol. 27, No. 9, 1960, pp. 653–658.
https://doi.org/10.2514/8.8704

[31] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge Univ. Press, Cambridge, England, U.K., 2004, pp. 531–535.