Transactions of the Indian National Academy of Engineering
https://doi.org/10.1007/s41403-024-00502-6

ORIGINAL ARTICLE

Comprehensive MPSP for Fast Optimal Control: Algorithm Development and Convergence Analysis

Prem Kumar[1] · Geethu Joseph[2] · Chandra R. Murthy[3] · Radhakant Padhi[4]

Received: 10 October 2023 / Accepted: 8 October 2024

Abstract
A computationally efficient state feedback optimal control synthesis approach, named *Comprehensive Model Predictive Static Programming (C-MPSP)*, is presented in this paper. Using C-MPSP, one can not only handle nonlinear systems but also optimize generic cost functions and impose the necessary path and terminal constraints. It can be applied to various problems, such as terminally constrained problems, regulator problems, tracking problems, and problems with or without path constraints. A rigorous convergence analysis is presented, which, under some mild conditions, proves that the entire iterative process is guaranteed to converge within a limited number of iterations. In addition, C-MPSP is computationally very efficient owing to several key features associated with the MPSP philosophy. Because of this, the C-MPSP algorithm can be applied to synthesize state feedback optimal controllers in real-time for many practical problems. Owing to its simplicity, the algorithm can be coded easily. The applicability and efficiency of the algorithm are illustrated by applying it to optimally guide a two-wheeled differentially-driven mobile robot on a curved road to reach its destination in the presence of state constraints (road boundaries and obstacles) and control constraints.

Keywords Optimal control · Fast optimal control · Model Predictive Static Programming · MPSP · Fast MPC · Mobile robots

Introduction

Optimal control theory is a robust framework to formulate and solve various challenging control synthesis problems. It can not only cater to various constraints, such as isoperimetric constraints over a time window, non-holonomic system dynamics constraint and additional path constraints in both state and control at each time instant, but also optimize a meaningful performance index in the process. Hence a large variety of problems, such as terminally constrained problems, regulator problems, tracking problems, and problems with or without path constraints, can be formulated within this framework. Consequently, it finds applications in many different engineering fields, such as aerospace, electrical, mechanical, robotics, process control, and biomedical, to name a few. Many classic books have been written on it in the past [see, e.g., Kirk (1970), Sage (1968), Bryson and Ho (1975)]. Some recent books have also appeared containing a few recently-developed algorithms and several challenging applications (Hager and Pardalos 2013; Hull 2013; Longuski et al. 2014; Ben-Asher 2010).

The formulation and analysis of an optimal control problem can be viewed from two different angles, namely, (i) classical calculus of variations approach leading to two-point boundary value problems (Kirk 1970), which is an indirect approach, and (ii) dynamic programming approach leading to the famous Hamilton-Jacobi-Bellman partial differential equation (Sage 1968), which is a direct approach. Both of these approaches, however, suffer from the well-known 'curse of complexity' (Ross 2015) and 'curse of dimensionality' (Bryson and Ho 1975) issues, respectively. Hence, unless the problem is fairly simple [e.g., the standard linear quadratic regulator theory (Naidu 2003)], an analytic closed-form solution is not possible in general. Because of this, several numerical methods have also been proposed

✉ Radhakant Padhi
padhi@iisc.ac.in

1 Defence R&D Organisation, Hyderabad, India

2 Signal Processing Systems Group, Delft University of Technology, Delft, Netherlands

3 ECE Department, Indian Institute of Science, Bengaluru, India

4 AE Department, Indian Institute of Science, Bengaluru, India

Published online: 12 December 2024

in the literature to solve nonlinear optimal control problems, such as the shooting method (Morrison et al. 1962) and gradient method (Kirk 1970) for the indirect approach as well as computational procedures for dynamic programming (Larson and Casti 1982) and more recently, reinforcement learning (Yan et al. 2023). All of these, however, are still computationally intensive. Hence, these are unsuitable for online computation of the optimal control trajectory, which remains a key bottleneck for their usage in many real-time decision-making problems.

Another popular approach for solving optimal control problems is the 'transcription philosophy' of direct optimization (Betts 2001). Here, an equivalent discrete static optimization problem is first formulated using a pre-selected time grid, and the discretized problem is then solved using a suitable static optimization algorithm. It turns out that incorporating path inequality constraints on states and/or control is inevitable in many problems for ensuring operational safety, and obtaining a feasible control solution is easier in the transcription approach. Unfortunately, it also leads to a large-dimensional optimization problem and associated problems such as huge computational burden, making it hard to apply in its basic form for many practical problems. Nonetheless, this forms the basis for the hugely-popular model predictive control (MPC) (Wang 2009; Allgöwer and Zheng 2012), which largely enables a 'sub-optimal' control solution with a restricted prediction horizon and even more restricted control horizon (thereby limiting the problem dimension). It can be noted here that a large number of innovations, including convergence and optimality guarantees, have been provided for both linear and nonlinear MPC problems under relevant assumptions (Chen and Allgöwer 1998; Mayne et al. 2000; Zheng et al. 2022). Despite these innovations, it inherently suffers from the associated computational burden, and, hence, has primarily been restricted to slow-varying systems (such as chemical and biomedical process control applications) and that too mainly for regulator problems. In such problems, a coarse grid in time can be employed, restricting the dimension of static optimization. The larger time window also helps in computing the control in real time. Even though literature has appeared for economic MPC (Rawlings et al. 2012), relaxing it from regulator problems and fast MPC (Wang and Boyd 2010), these are still restricted mainly to slow-varying linear systems.

Many practically relevant problems, however, do not enjoy the above advantage. For example, problems in aerospace, mobile robotics, etc. are usually governed by complex nonlinear system dynamics. Moreover, they exhibit fast-changing system dynamics and are often required to meet stringent performance requirements (e.g., zero miss distance and a desired impact angle in missile guidance). For such challenging problems, the available 'fast MPC' algorithms are often found to be inadequate, in the sense that they are not sufficiently fast to be applied for such problems. Keeping such applications in mind, however, innovative optimal control solution approaches have appeared in the aerospace literature over the last decades, such as pseudo-spectral optimal control (Fahroo and Ross 2002; Gong et al. 2008), adaptive critic technique (Balakrishnan and Biega 1996).

One such powerful and innovative approach is the *Model Predictive Static Programming (MPSP)* (Padhi and Kothari 2009; Halbe et al. 2014). In its original form, the MPSP technique solves a class of nonlinear fixed final time optimal control problems, where the goal is to minimize the control effort while ensuring that the output vector satisfies a set of hard constraints at the final time t_f. Owing to fundamental key innovations such as conversion to a low-dimensional static optimization problem only in control variables and recursive computation of the sensitivity matrices that form the core of this algorithm, the MPSP technique has been found to be computationally very efficient. Over the last decade, the original version of MPSP has been extended to include variability in the final time or state (Maity et al. 2016; Ghazaei Ardakani et al. 2019), tracking problems (Kumar et al. 2018) and impulsive nature of the control action (Sakode and Padhi 2014). Inspired by the pseudo-spectral philosophy, the quasi-spectral MPSP (Mondal and Padhi 2018) has also been proposed to further reduce computational time. The MPSP technique has also been applied to a host of challenging problems such as missile guidance (Dwivedi et al. 2011; Oza and Padhi 2012), re-entry guidance (Halbe et al. 2014), mobile robotics (Kumar et al. 2018; Prakash et al. 2022), lunar soft-landing (Sachan and Padhi 2019), chemical process control (le Roux et al. 2014). A recent and comprehensive survey of MPSP and its variants can be found in Padhi et al. (2024).

Despite its utility and several extensions, the MPSP technique suffers from three important restrictions, which limit its application domain. These are: (i) its applicability is limited to optimizing cost functions that are necessarily functions of only the control variables, (ii) its inability to handle general path inequality constraints, e.g., due to obstacles, and (iii) even though numerical results have always been promising, a systematic proof of convergence has never been established. These limitations obviously restrict the class of problems to which the MPSP method can be used. The main aim of this paper is to overcome these limitations and thereby present a comprehensive approach, called *comprehensive MPSP (C-MPSP)*. Specifically, we present an algorithm that can handle general path inequality constraints and encompasses general nonlinear optimal control problems. Moreover, we provide the first rigorous convergence analysis, which in turn allows one to apply C-MPSP to various problems with confidence.

It must be mentioned here that the problem of handling path constraints under the MPSP framework has gained attention recently. Two papers recently appeared almost simultaneously (Kumar et al. 2018; Hong et al. 2019). However, in Kumar et al. (2018), the tracking-oriented MPSP is confined only to tracking problems. Moreover, no convergence analysis has been carried out. In Hong et al. (2019), the problem formulation has been confined to quadratic cost functions with linear and/or quadratic path constraints, followed by solving it using an interior point algorithm. In contrast, this paper does not restrict the problem formulation to any such specialized domains. Instead, it attempts to solve generic nonlinear optimal control problems by a *successive quadratic approximation* of the cost function and *successive linearization* of the nonlinear constraints. In other words, the *successive convexification* of the problem makes C-MPSP very generic and therefore applicable to a wide variety of problems. Moreover, it is better to follow a 'sequential quadratic programming' approach (which is followed here) instead of an interior-point approach in Hong et al. (2019). This is because the quadratic programming approach does not bring in the additional difficult-to-satisfy requirement that the initial guess must satisfy the imposed constraints. In addition, the systematic convergence analysis done in this paper guarantees that the approach will succeed to find the optimal solution as long as (i) *the problem admits a feasible solution* and (ii) the initial guess is sufficiently close to the optimal solution. These are, in general, mild assumptions required for almost all convergence proofs associated with computational algorithms. Note that for the unconstrained case (i.e., with no path constraint), like other MPSP algorithms, the entire iterative process in the C-MPSP too can be carried out in closed form without resorting to any numerical optimization solver.

To demonstrate the utility and applicability of the approach, a differential two-wheeled mobile robot problem has been solved using the proposed comprehensive MPSP approach. The algorithm is successful in quickly finding a collision-free path from an initial position to the destination within the curved road boundaries. Details of this problem and the numerical results are included in Sect. 5. The next section (Sect. 2), develops the C-MPSP algorithm to solve the optimal control problem both without and with the path constraints on the input and the state.

Comprehensive MPSP: Mathematical Details

A generic optimal control problem in discrete time can be formulated as follows:

$$System\ dynamics\ :\ X_{k+1} = F_k\left(X_k, U_k\right), \tag{1}$$

$$Output\ equation\ :\ Y_k = h_k\left(X_k, U_k\right), \tag{2}$$

$$Cost\ function\ :\ J = \sum_{k=0}^{N} L_k\left(X_k, U_k\right), \tag{3}$$

$$Trajectory\ constraints\ at\ k^{th}\ time\ step: \begin{array}{l} g_k^0\left(X_k, U_k\right) \le 0, \\ g_k^1\left(X_k, U_k\right) \le 0, \\ \vdots \\ g_k^{l_k}\left(X_k, U_k\right) \le 0, \end{array} \tag{4}$$

$$Terminal\ output\ constraint\ :\ Y_N = Y_N^*, \tag{5}$$

$$Initial\ state:\ X_0 = X(t_0), \tag{6}$$

where $X_k \in \Re^n$, $U_k \in \Re^r$, $Y_k \in \Re^m$ are the state, control and output vectors respectively at k^{th} time step. Here, X_0 is the initial state at the initial time t_0. The function $L_k(\cdot)$ in the cost function J is assumed to be a real-valued and smooth scalar function of the state and control (in general, it can be nonlinear). Without loss of generality, it is assumed to be a minimization problem; hence, the range of $L_k(\cdot)$ is assumed to be non-negative. Further, $\left[g_k^0,\ g_k^1,\ \ldots,\ g_k^{l_k}\right]^T$ are $(l_k + 1)$ constraints on the state X_k and control U_k at the k^{th} time step. It can be mentioned here that if one formulates an optimal control problem in continuous time, the problem can suitably be discretized first to proceed further.

To solve the optimal control problem contained in Eqs. (1)–(6), a new optimal control solution approach named as *C-MPSP* is presented in this paper. As mentioned in Sect. 1, C-MPSP is an iterative algorithm and requires an initial guess of control history to start the iterations. The state dynamics (1) with the initial state (6) is used to predict the entire state trajectory using a guess control history. Then, the state dynamics, output equation, trajectory constraints, and cost function are linearized along the predicted trajectory to form a quadratic programming problem. Details of this approach are provided in the following subsections.

C-MPSP Algorithm

The proposed C-MPSP algorithm quickly and provably improves the control solution iteratively towards the optimal solution starting from a reasonably good guess. The necessary steps are elaborated below.

First, developing the C-MPSP algorithm requires a reformulation of the problem in (1)–(6). First, the state, control and output at the k^{th} time step during i^{th} iteration are denoted by $X_k^i \in \Re^n$, $U_k^i \in \Re^r$ and $Y_k^i \in \Re^m$, respectively. Here, the superscript i denotes the iteration index. As one moves from the i^{th} iteration to the $(i+1)^{th}$ iteration, a new set of updated state, control, and output histories are

generated. The difference of the state, control, and output variables between two consecutive iterations *i* and $(i+1)$ at the k^{th} time step are as defined as follows

$$\begin{aligned} X_k^{i+1} &\triangleq X_k^i + \Delta X_k^i, \\ U_k^{i+1} &\triangleq U_k^i + \Delta U_k^i, \\ Y_k^{i+1} &\triangleq Y_k^i + \Delta Y_k^i. \end{aligned} \tag{7}$$

Next, the problem is rewritten as follows.

State Dynamics

The state dynamics and output equation, including the iteration index, can now be written as

$$X_{k+1}^i = F_k\left(X_k^i, U_k^i\right), \tag{8}$$

$$Y_k^i = h_k\left(X_k^i, U_k^i\right). \tag{9}$$

Terminal Constraint

The objective of the algorithm during i^{th} iteration is to determine a suitable control history $U_k^{i+1},\ k = 0, 1, \ldots, N$, so that the output at the final time step at the end of i^{th} iteration equals the desired final output Y_N^*. In other words, in iteration *i*, we seek a solution for iteration $i+1$ such that $Y_N^{i+1} = Y_N^*$. This equality condition can be written differently using (7), as

$$\Delta Y_N^i + Y_N^i - Y_N^* = 0. \tag{10}$$

Under small error approximation ($\Delta Y_N^i \to dY_N^i$) the control constraint (10) can be written as below

$$dY_N^i + \Delta Y_N^* = 0, \tag{11}$$

where $\Delta Y_N^* \triangleq Y_N^i - Y_N^*$. This equation represents the equality constraint imposed at the final time step of the proposed reformulation.

Cost Function

Similarly, including the iteration index, the cost function in (3) can be written as

$$J^i = \sum_{k=0}^{N} L_k(X_k^{i+1}, U_k^{i+1}). \tag{12}$$

Note that the state and control histories in the i^{th} iteration are updated in such a way that the updated state and control, i.e., X_k^{i+1} and U_k^{i+1} respectively, should reduce the cost. Hence the notation J^i on the left-hand side.

Path Constraints

Similarly, the path constraints in (4) at k^{th} time step, $k = 1, \ldots, (N-1)$ can be written as

$$\begin{aligned} g_k^0\left(X_k^{i+1}, U_k^{i+1}\right) &\leq 0, \\ g_k^1\left(X_k^{i+1}, U_k^{i+1}\right) &\leq 0, \\ &\vdots \\ g_k^{l_k}\left(X_k^{i+1}, U_k^{i+1}\right) &\leq 0. \end{aligned} \tag{13}$$

Once again, note that these constraints are imposed on the updated states and control to ensure that the converged states' history and control history satisfy the constraints imposed on the state and control vector given by (13).

State and Output Error Computation

To obtain the state deviation with respect to control, the state vector at time step k and iteration $(i+1)$ (i.e., X_k^{i+1}) is expressed by a Taylor series expansion up to first order terms as

$$\begin{aligned} X_k^{i+1} &= F_{k-1}\left(X_{k-1}^{i+1}, U_{k-1}^{i+1}\right), \\ &= X_k^i + \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]^T_{\left(X_{k-1}^i, U_{k-1}^i\right)} \Delta X_{k-1}^i \\ &\quad + \left[\frac{\partial F_{k-1}}{\partial U_{k-1}}\right]^T_{\left(X_{k-1}^i, U_{k-1}^i\right)} \Delta U_{k-1}^i + \mathrm{HOT}, \end{aligned} \tag{14}$$

where ΔX_k^i is the deviation in the state and ΔU_k^i is the deviation in the control input at time step *k* during iteration *i*. Moreover, the partial derivatives are defined such that $\left[\frac{\partial F_{k-1}}{\partial U_{k-1}}\right] \in \Re^{r \times n}$; similar definitions are followed in the rest of the paper. Under the assumption of small input deviations ($\Delta U_k^i \to dU_k^i$) and small state deviations ($\Delta X_k^i \to dX_k^i$), the higher order terms (HOT) in (14) can be neglected. Then, from the definition in (7), (14) can be written as

$$dX_k^i = \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]^T_{\left(X_{k-1}^i, U_{k-1}^i\right)} dX_{k-1}^i + \left[\frac{\partial F_{k-1}}{\partial U_{k-1}}\right]^T_{\left(X_{k-1}^i, U_{k-1}^i\right)} dU_{k-1}^i. \tag{15}$$

Further, the deviation in the state at the $(k-1)^{th}$ time step (that is, dX_{k-1}^i) can be expanded in terms of dX_{k-2}^i and dU_{k-2}^i; and so on. This expansion continues until the state deviation of the initial condition (i.e., dX_0^i). Eventually, one can write

$$dX_k^i = \left[A^k\right]^i dX_0^i + \left[B_0^k\right]^i dU_0^i + \cdots + \left[B_{k-1}^k\right]^i dU_{k-1}^i, \tag{16}$$

where $\left[A^k\right]^i \in \Re^{n\times n}$ is the sensitivity of state at the k^{th} time step (i.e. X_k) during the i^{th} iteration due to the deviation in initial state X_0. Also, $\left[B_j^k\right]^i \in \Re^{n\times r}$ is the *sensitivity matrix of the state* at the k^{th} time step due to the change in the control input at the j^{th} time step during the i^{th} iteration.

$$\left[A^k\right]^i = \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]^T_{\left(X^i_{k-1},U^i_{k-1}\right)} \cdots \left[\frac{\partial F_0}{\partial X_0}\right]^T_{\left(X^i_0,U^i_0\right)}, \tag{17}$$

$$\left[B_j^k\right]^i = \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]^T_{\left(X^i_{k-1},U^i_{k-1}\right)} \cdots \left[\frac{\partial F_{j+1}}{\partial X_{j+1}}\right]^T_{\left(X^i_{j+1},U^i_{j+1}\right)} \left[\frac{\partial F_j}{\partial U_j}\right]^T_{\left(X^i_j,U^i_j\right)}. \tag{18}$$

However, since it is assumed that the initial condition is known, there is no error in the initial state (i.e., $dX_0^i = 0$). Hence, the state deviation at the k^{th} step reduces to

$$dX_k^i = \sum_{j=0}^{k-1} \left[B_j^k\right]^i dU_j^i. \tag{19}$$

Thus, (19) gives the state sensitivity at the k^{th} time step due to changes in control input at all time steps prior to it. Next, the expression for the *output* deviation at the k^{th} time step at the iteration can be written as

$$dY_k^i = \left[\frac{\partial h_k}{\partial X_k}\right]^T_{\left(X^i_k,U^i_k\right)} dX_k^i + \left[\frac{\partial h_k}{\partial U_k}\right]^T_{\left(X^i_k,U^i_k\right)} dU_k^i. \tag{20}$$

Substituting the expression for dX_k^i from (19) in the expression of dY_k^i from (20) gives

$$\begin{aligned} dY_k^i &= \left[\frac{\partial h_k}{\partial X_k}\right]^T_{\left(X^i_k,U^i_k\right)} \sum_{j=0}^{k-1} \left[B_j^k\right]^i dU_j^i + \left[\frac{\partial h_k}{\partial U_k}\right]^T_{\left(X^i_k,U^i_k\right)} dU_k^i, \\ &= \sum_{j=0}^{k} \left[S_j^k\right]^i dU_j^i, \end{aligned} \tag{21}$$

where $\left[S_j^k\right]^i \in \Re^{m\times r}$, appropriately defined for $j < k$ and $j = k$, is the *sensitivity matrix of the output* at k^{th} time step due to the changes in the control input prior to time *k*.

Note that (21) is also valid for the output deviation at the final time step, which can be written as

$$dY_N^i = \sum_{j=0}^{N} \left[S_j^N\right]^i dU_j^i. \tag{22}$$

Since input changes at future time steps do not change the state vector at the current time step, $\left[B_j^k\right]^i$ is set to zero for all $j \geq k,\ \forall i$. Next, to reduce the computational requirements of the algorithm, $\left[B_j^k\right]^i$ is recursively computed as

$$\left.\begin{aligned} \left[\Phi_k^k\right]^i &= I_{n\times n}, \\ \left[\Phi_j^k\right]^i &= \left[\Phi_{j+1}^k\right]^i \left[\frac{\partial F_j}{\partial X_j}\right]^T_{\left(X^i_j,U^i_j\right)}, \\ \left[B_j^k\right]^i &= \left[\Phi_{j+1}^k\right]^i \left[\frac{\partial F_j}{\partial U_j}\right]^T_{\left(X^i_j,U^i_j\right)}, \\ \left[S_j^k\right]^i &= \left[\frac{\partial h_k}{\partial X_k}\right]^T_{\left(X^i_k,U^i_k\right)} \left[B_j^k\right]^i, \end{aligned}\right\} \quad \forall j < k$$

$$\begin{aligned} \left[S_j^k\right]^i &= \left[\frac{\partial h_k}{\partial U_k}\right]^T_{\left(X^i_k,U^i_k\right)}, && j = k \\ \left[B_j^k\right]^i &= [0]_{n\times m}, && \forall j \geq k \\ \left[S_j^k\right]^i &= [0]_{p\times m}, && \forall j > k \end{aligned} \tag{23}$$

where $\left[\Phi_j^k\right]^i$ is the sensitivity matrix of the state at the k^{th} time step with respect to change in state at the j^{th} time step.

Control Computation in the Unconstrained Case

In this case, following the principle of unconstrained optimization, from (12), (11) and (22), the augmented cost function is constructed as

$$\tilde{J}^i = \sum_{k=0}^{N} L_k^{i+1} + \left(\lambda^i\right)^T \left(\sum_{j=0}^{N} \left[S_j^N\right]^i dU_j^i + \Delta Y_N^{*i}\right), \tag{24}$$

where $\lambda^i \in \Re^m$ is the Lagrange multiplier and $L_k^{i+1} \triangleq L_k(X_k^{i+1}, U_k^{i+1}) \geq 0,\ \forall X_k, U_k$. Next, the necessary conditions for optimality can be written as

$$\frac{\partial \tilde{J}^i}{\partial \lambda} = 0, \tag{25}$$

$$\frac{\partial \tilde{J}^i}{\partial \left(dU_j^i\right)} = 0,\ \forall j \leq N \tag{26}$$

It is straightforward to see that (25) leads to

$$\sum_{j=0}^{N} \left[S_j^N\right]^i dU_j^i + \Delta Y_N^{*i} = 0. \tag{27}$$

However, to simplify (26), one can notice that

$$\frac{\partial \tilde{J}^i}{\partial \left(dU_j^i\right)} = \sum_{k=0}^{N} \frac{\partial L_k\left(X_k^{i+1}, U_k^{i+1}\right)}{\partial \left(dU_j^i\right)} + \left(\left[S_j^N\right]^i\right)^T \lambda^i. \tag{28}$$

The next step is to obtain simplified expressions corresponding to each term in the right hand side of (28). The first term involves evaluation of partial derivative of function $L_k(X_k^{i+1}, U_k^{i+1})$ with respect to dU_j^i. But $L_k(\cdot)$ is a generic function of both the state and control, and its

partial derivative cannot directly be written in explicit form. At this point, a simplification is introduced for evaluation of the partial derivative of $L_k(\cdot)$ with respect to dU_j^i. First, the function $L_k(X_k^{i+1}, U_k^{i+1})$ is expanded about the point (X_k, U_k) using a Taylor series expansion. Then, the partial derivative of $L_k(X_k^{i+1}, U_k^{i+1})$ with respect to dU_j^i is evaluated by differentiating the Taylor series expansion with respect to dU_j^i.

The Taylor series expansion of $L_k(\cdot)$, up to the second order term, reads as

$$\begin{aligned} L_k\left(X_k^{i+1}, U_k^{i+1}\right) = L_k\left(X_k^i, U_k^i\right) &+ \left(\left[\partial_{X_k} L_k\right]_k^i\right)^T dX_k^i \\ &+ \left(\left[\partial_{U_k} L_k\right]_k^i\right)^T dU_k^i \\ &+ \frac{1}{2}\left(dX_k^i\right)^T \left[\partial_{X_k U_k} L_k\right]_k^i dU_k^i \\ &+ \frac{1}{2}\left(dU_k^i\right)^T \left[\partial_{U_k X_k} L_k\right]_k^i dX_k^i \\ &+ \frac{1}{2}\left(dX_k^i\right)^T \left[\partial_{X_k}^2 L_k\right]_k^i dX_k^i \\ &+ \frac{1}{2}\left(dU_k^i\right)^T \left[\partial_{U_k}^2 L_k\right]_k^i dU_k^i, \end{aligned} \tag{29}$$

where $\left[\partial_{X_k} L_k\right]_k^i \triangleq \left[\frac{\partial L_k}{\partial X_k}\right]_{(X_k^i, U_k^i)}$, $\left[\partial_{U_k} L_k\right]_k^i \triangleq \left[\frac{\partial L_k}{\partial U_k}\right]_{(X_k^i, U_k^i)}$, $\left[\partial_{X_k}^2 L_k\right]_k^i \triangleq \left[\frac{\partial^2 L_k}{\partial X_k^2}\right]_{(X_k^i, U_k^i)}$, $\left[\partial_{U_k}^2 L_k\right]_k^i \triangleq \left[\frac{\partial^2 L_k}{\partial U_k^2}\right]_{(X_k^i, U_k^i)}$, $\left[\partial_{X_k U_k} L_k\right]_k^i \triangleq \left[\frac{\partial^2 L_k}{\partial X_k U_k}\right]_{(X_k^i, U_k^i)}$, and $\left[\partial_{U_k X_k} L_k\right]_k^i \triangleq \left[\frac{\partial^2 L_k}{\partial U_k X_k}\right]_{(X_k^i, U_k^i)}$.

Equation (29) comprises several terms, some of which are purely dependent on state X_k^i and control U_k^i, while some are dependent on dX_k^i (deviation in the state) and dU_k^i (deviation in control). But X_k^i and U_k^i are the state history and control history, respectively, at k^{th} time step and are constant quantities during the iteration process. Hence, the terms solely dependent on X_k^i and U_k^i are constants in (29), while dX_k^i and dU_k^i are variables and have to be handled accordingly. The partial derivative of $L_k\left(X_k^{i+1}, U_k^{i+1}\right)$ with respect to dU_j^i can thus be written as

$$\begin{aligned} \frac{\partial L_k\left(X_k^{i+1}, U_k^{i+1}\right)}{\partial\left(dU_j^i\right)} = &\left[\frac{\partial\left(dX_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left[\partial_{X_k} L_k\right]_k^i \\ &+ \left[\frac{\partial\left(dU_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left[\partial_{U_k} L_k\right]_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dX_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left[\partial_{X_k U_k} L_k\right]_k^i dU_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dU_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left(\left[\partial_{X_k U_k} L_k\right]_k^i\right)^T dX_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dX_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left[\partial_{X_k}^2 L_k\right]_k^i dX_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dX_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left(\left[\partial_{X_k}^2 L_k\right]_k^i\right)^T dX_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dU_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left[\partial_{U_k X_k} L_k\right]_k^i dX_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dX_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left(\left[\partial_{U_k X_k} L_k\right]_k^i\right)^T dU_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dU_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left[\partial_{U_k}^2 L_k\right]_k^i dU_k^i \\ &+ \frac{1}{2}\left[\frac{\partial\left(dU_k^i\right)}{\partial\left(dU_j^i\right)}\right]\left(\left[\partial_{U_k}^2 L_k\right]_k^i\right)^T dU_k^i. \end{aligned} \tag{30}$$

It is apparent that $\frac{\partial\left(dX_k^i\right)}{\partial\left(dU_j^i\right)}$ is the sensitivity of the state at the k^{th} time step due to change in control at the j^{th} time step. Hence, from (19), this is nothing but $\left(\left[B_j^k\right]^i\right)^T$. Moreover, this can also be obtained by taking the partial derivative of both sides of (19) with respect to dU_j^i. On the other hand, $\frac{\partial\left(dU_k^i\right)}{\partial\left(dU_j^i\right)}$ is the change in control input at the k^{th} time step due to change in control at the j^{th} time step, but control inputs at every time instant are decision variables and are is independent of the control input at any other time step. Hence, this derivative can be represented by a Kronecker delta function δ_{jk}. Using this definition of partial derivatives, (30) can be written as

$$\begin{aligned} &\frac{\partial L_k\left(X_k^{i+1}, U_k^{i+1}\right)}{\partial\left(dU_j^i\right)} = \left(\left[B_j^k\right]^i\right)^T \left[\partial_{X_k} L_k\right]_k^i + \delta_{jk}\left[\partial_{U_k} L_k\right]_k^i \\ &+ \frac{1}{2}\delta_{jk}\left\{\left[\partial_{U_k X_k} L_k\right]_k^i + \left(\left[\partial_{X_k U_k} L_k\right]_k^i\right)^T\right\}\sum_{l=0}^{k-1}\left[B_l^k\right]^i dU_l^i \\ &+ \frac{1}{2}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k}^2 L_k\right]_k^i + \left(\left[\partial_{X_k}^2 L_k\right]_k^i\right)^T\right\}\sum_{l=0}^{k-1}\left[B_l^k\right]^i dU_l^i \\ &+ \frac{1}{2}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k U_k} L_k\right]_k^i + \left(\left[\partial_{U_k X_k} L_k\right]_k^i\right)^T\right\} dU_k^i \\ &+ \frac{1}{2}\delta_{jk}\left\{\left[\partial_{U_k}^2 L_k\right]_k^i + \left(\left[\partial_{U_k}^2 L_k\right]_k^i\right)^T\right\} dU_k^i. \end{aligned} \tag{31}$$

The first term of (28) is the summation of partial derivatives of $L_k(\cdot)$ with respect to control deviation dU_j^i. Using the summation operator on both sides of (31) gives

$$\begin{aligned}\sum_{k=0}^{N}\frac{\partial L_k\left(X_k^{i+1},U_k^{i+1}\right)}{\partial\left(dU_j^i\right)}&=\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left[\partial_{X_k}L_k\right]_k^i+\left[\partial_{U_j}L_j\right]_j^i\\&+\frac{1}{2}\left\{\left[\partial_{U_jX_j}L_j\right]_j^i+\left(\left[\partial_{X_jU_j}L_j\right]_j^i\right)^T\right\}\sum_{l=0}^{j-1}\left[B_l^j\right]^i dU_l^i\\&+\frac{1}{2}\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k}^2L_k\right]_k^i+\left(\left[\partial_{X_k}^2L_k\right]_k^i\right)^T\right\}\sum_{l=0}^{k-1}\left[B_l^k\right]^i dU_l^i\\&+\frac{1}{2}\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_kU_k}L_k\right]_k^i+\left(\left[\partial_{U_kX_k}L_k\right]_k^i\right)^T\right\}dU_k^i\\&+\frac{1}{2}\left\{\left[\partial_{U_j}^2L_j\right]_j^i+\left(\left[\partial_{U_j}^2L_j\right]_j^i\right)^T\right\}dU_j^i.\end{aligned}\tag{32}$$

Using the fact that $\left[B_l^k\right]^i=0\ \forall l\geq k$, it can be written that $\sum_{l=0}^{k-1}\left[B_l^k\right]^i dU_l^i=\sum_{l=0}^{N}\left[B_l^k\right]^i dU_l^i$. This is used for the algebraic simplification of the third term in (32) to obtain the following:

$$\begin{aligned}&\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k}^2L_k\right]_k^i+\left(\left[\partial_{X_k}^2L_k\right]_k^i\right)^T\right\}\sum_{l=0}^{k-1}\left[B_l^k\right]^i dU_l^i\\&=\sum_{l=0}^{N}\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k}^2L_k\right]_k^i+\left(\left[\partial_{X_k}^2L_k\right]_k^i\right)^T\right\}\left[B_l^k\right]^i dU_l^i.\end{aligned}\tag{33}$$

Using (33), (32) can be simplified as

$$\begin{aligned}\sum_{k=0}^{N}\frac{\partial L_k\left(X_k^{i+1},U_k^{i+1}\right)}{\partial\left(dU_j^i\right)}&=\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left[\partial_{X_k}L_k\right]_k^i+\left[\partial_{U_j}L_j\right]_j^i\\&+\frac{1}{2}\left\{\left[\partial_{U_jX_j}L_j\right]_j^i+\left(\left[\partial_{X_jU_j}L_j\right]_j^i\right)^T\right\}\sum_{l=0}^{j-1}\left[B_l^j\right]^i dU_l^i\\&+\frac{1}{2}\sum_{l=0}^{N}\sum_{k=\max(l+1,j+1)}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k}^2L_k\right]_k^i\right.\\&\left.+\left(\left[\partial_{X_k}^2L_k\right]_k^i\right)^T\right\}\left[B_l^k\right]^i dU_l^i+\frac{1}{2}\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_kU_k}L_k\right]_k^i\right.\\&\left.+\left(\left[\partial_{U_kX_k}L_k\right]_k^i\right)^T\right\}dU_k^i+\frac{1}{2}\left\{\left[\partial_{U_j}^2L_j\right]_j^i+\left(\left[\partial_{U_j}^2L_j\right]_j^i\right)^T\right\}dU_j^i.\end{aligned}\tag{34}$$

Using (34) in (28) and equating it to zero leads to the second necessary condition of optimality, the detailed expression of which is omitted here for brevity.

Next, one can observe here that the optimality conditions (26) and (27) are essentially linear equations, and hence, can be written in compact form as

$$\begin{bmatrix}A_{UU}&A_{U\lambda}\\A_{\lambda U}&A_{\lambda\lambda}\end{bmatrix}\begin{bmatrix}\delta U^i\\\lambda^i\end{bmatrix}+\begin{bmatrix}b_U\\b_\lambda\end{bmatrix}=0,\tag{35}$$

where $\delta U^i\triangleq\begin{bmatrix}\left(dU_0^i\right)^T&\dots&\left(dU_N^i\right)^T\end{bmatrix}^T$, $A_{\lambda\lambda}=0$, and other variables are defined as

$$A_{UU}\triangleq\begin{bmatrix}a_{UU_{00}}&a_{UU_{01}}&\cdots&a_{UU_{0N}}\\a_{UU_{10}}&a_{UU_{11}}&\cdots&a_{UU_{1N}}\\\vdots&\vdots&\ddots&\vdots\\a_{UU_{N0}}&a_{UU_{N1}}&\cdots&a_{UU_{NN}}\end{bmatrix},\tag{36}$$

$$A_{U\lambda}\triangleq\begin{bmatrix}a_{U\lambda_0}\\a_{U\lambda_1}\\\vdots\\a_{U\lambda_N}\end{bmatrix},\qquad A_{\lambda U}=A_{U\lambda}^T,\tag{37}$$

$$\delta U^i=\begin{bmatrix}dU_0^i\\dU_1^i\\\vdots\\dU_N^i\end{bmatrix},\qquad b_U=\begin{bmatrix}b_{U_0}\\b_{U_1}\\\vdots\\b_{U_N}\end{bmatrix},\qquad b_\lambda=\Delta Y_N^{*^i},\tag{38}$$

$$\begin{aligned}a_{UU_{jl}}&\triangleq\frac{1}{2}\delta_{jl}\left\{\left[\partial_{U_l}^2L_l\right]_l^i+\left(\left[\partial_{U_l}^2L_l\right]_l^i\right)^T\right\}\\&+\frac{1}{2}\sum_{k=0}^{N}\left(\left[B_j^k\right]^i\right)^T\left\{\left[\partial_{X_k}^2L_k\right]_k^i+\left(\left[\partial_{X_k}^2L_k\right]_k^i\right)^T\right\}\left[B_l^k\right]^i+\\&\frac{1}{2}\left(\left[B_j^l\right]^i\right)^T\left\{\left[\partial_{X_lU_l}L_l\right]_l^i+\left(\left[\partial_{U_lX_l}L_l\right]_l^i\right)^T\right\}\\&+\frac{1}{2}\left\{\left(\left[\partial_{X_jU_j}L_j\right]_j^i\right)^T+\left[\partial_{U_jX_j}L_j\right]_j^i\right\}\left[B_l^j\right]^i,\end{aligned}\tag{39}$$

$$a_{U\lambda_j}\triangleq\left(\left[S_j^N\right]^i\right)^T,\tag{40}$$

$$b_{U_j}\triangleq\sum_{l=0}^{N}\left\{\left(\left[B_j^l\right]^i\right)^T\left[\partial_{X_l}L_l\right]_l^i+\delta_{jl}\left[\partial_{U_l}L_l\right]_l^i\right\},\tag{41}$$

$$b_\lambda\triangleq\Delta Y_N^*.\tag{42}$$

Solving (35) gives the optimal control deviations and Lagrange multiplier (λ) during the i^{th} iteration. The addition of these optimal control deviations to the previous control history gives the updated control history, namely,

$$U_j^{i+1}=U_j^i+dU_j^i,\qquad\forall\ j\in\{0,1,\dots,N\}.\tag{43}$$

In this manner, the control input is updated from the control history in an iteration cycle.

Control Computation in the Constrained Case

In general, the constraints can be written as in (13), containing l_k inequality constraints in terms of the state and control vector at the k^{th} time step. Now, the state and control histories need to be updated in such a manner that they satisfy the applicable constraints. The constrained nonlinear optimal control problem contained in (8), (11), (12) and (13) can be solved efficiently using algorithms for solving the *approximate quadratic programming problem with linear constraints*; such that the solution converges to the solution of the original problem.

The first step is to convert the nonlinear cost function into an approximate quadratic cost using the Taylor series

approximation as in (29), which can be simplified using (19) as

$$\begin{aligned} L_k\left(X_k^{i+1},U_k^{i+1}\right) &= L_k\left(X_k^i,U_k^i\right)+\left(\left[\partial_{X_k}L_k\right]_k^i\right)^T\sum_{j=0}^{k-1}\left[B_j^k\right]^i dU_j^i \\ &+\left(\left[\partial_{U_k}L_k\right]_k^i\right)^T dU_k^i \\ &+\frac{1}{2}\left(\sum_{j=0}^{k-1}\left[B_j^k\right]^i dU_j^i\right)^T\left[\partial_{X_kU_k}L_k\right]_k^i dU_k^i \\ &+\frac{1}{2}\left(dU_k^i\right)^T\left[\partial_{U_kX_k}L_k\right]_k^i\sum_{j=0}^{k-1}\left[B_j^k\right]^i dU_j^i \\ &+\frac{1}{2}\left(\sum_{j=0}^{k-1}\left[B_j^k\right]^i dU_j^i\right)^T\left[\partial^2_{X_k}L_k\right]_k^i\sum_{l=0}^{k-1}\left[B_l^k\right]^i dU_l^i \\ &+\frac{1}{2}\left(dU_k^i\right)^T\left[\partial^2_{U_k}L_k\right]_k^i dU_k^i. \end{aligned} \tag{44}$$

Using (44), and carrying out the necessary algebra, the cost function (12) can be written in a quadratic form as

$$\begin{aligned} J^i=&\frac{1}{2}\left(\delta U^i\right)^T\left\{\left(\left[B\right]^i\right)^T\left(L_{XX}^i\right)\left[B\right]^i\right. \\ &\left.+\left(\left[B\right]^i\right)^T\left(L_{XU}^i\right)+\left(L_{UX}^i\right)\left[B\right]^i+\left(L_{UU}^i\right)\right\}\delta U^i \\ &+\left(\left(L_X^i\right)^T[B]^i+\left(L_U^i\right)^T\right)\delta U^i+\sum_{k=0}^{N}L_k\left(X_k^i,U_k^i\right), \end{aligned} \tag{45}$$

where

$$\delta U^i\triangleq\begin{bmatrix} dU_0^i\\ dU_1^i\\ \vdots\\ dU_N^i\end{bmatrix},\quad \left(\left[B^k\right]^i\right)^T\triangleq\begin{bmatrix}\left(\left[B_0^k\right]^i\right)^T\\ \left(\left[B_1^k\right]^i\right)^T\\ \vdots\\ \left(\left[B_N^k\right]^i\right)^T\end{bmatrix}, \tag{46}$$

$$\left[B\right]^i\triangleq\begin{bmatrix}\left[B^0\right]^i\\ \vdots\\ \left[B^N\right]^i\end{bmatrix}=\begin{bmatrix}\left[B_0^0\right]^i & \cdots & \left[B_N^0\right]^i\\ \vdots & \ddots & \vdots\\ \left[B_0^N\right]^i & \cdots & \left[B_N^N\right]^i\end{bmatrix}, \tag{47}$$

$$L_X^i\triangleq\begin{bmatrix}\left[\partial_{X_0}L_0\right]_0^i\\ \vdots\\ \left[\partial_{X_N}L_N\right]_N^i\end{bmatrix},\quad L_U^i\triangleq\begin{bmatrix}\left[\partial_{U_0}L_0\right]_0^i\\ \vdots\\ \left[\partial_{U_N}L_N\right]_N^i\end{bmatrix}, \tag{48}$$

$$L_{XX}^i\triangleq\begin{bmatrix}\left[\partial^2_{X_0}L_0\right]_0^i & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \left[\partial^2_{X_N}L_N\right]_N^i\end{bmatrix}, \tag{49}$$

$$L_{UU}^i\triangleq\begin{bmatrix}\left[\partial^2_{U_0}L_0\right]_0^i & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \left[\partial^2_{U_N}L_N\right]_N^i\end{bmatrix}, \tag{50}$$

$$L_{UX}^i\triangleq\begin{bmatrix}\left[\partial_{U_0X_0}L_0\right]_0^i & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \left[\partial_{U_NX_N}L_N\right]_N^i\end{bmatrix}, \tag{51}$$

$$L_{XU}^i\triangleq\begin{bmatrix}\left[\partial_{X_0U_0}L_0\right]_0^i & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \left[\partial_{X_NU_N}L_N\right]_N^i\end{bmatrix}. \tag{52}$$

The equality constraint at the final time step (11) can be simplified by using the expression for the output deviation at the final time step (i.e., dY_N^i) from (22), resulting in

$$\sum_{j=0}^{k}\left[S_j^k\right]^i dU_j^i+\Delta Y_N^{*^i}=0. \tag{53}$$

Using the notation of δU^i from (46) and defining a new final output sensitivity matrix, the control constraint in (53) can be written as

$$\left[S^N\right]^i\delta U^i+\Delta Y_N^{*^i}=0, \tag{54}$$

where the output sensitivity matrix at the final time step is defined as

$$\left[S^N\right]^i\triangleq\begin{bmatrix}\left(\left[S_0^N\right]^i\right)^T & \cdots & \left(\left[S_N^N\right]^i\right)^T\end{bmatrix}^T. \tag{55}$$

Next, the nonlinear path constraints in (13) are simplified to linearized constraints on the state and control deviations using a Taylor series approximation and are written as

$$\begin{aligned} &\left[\frac{\partial g_k^{l_k}}{\partial X_k}\right]^T_{\left(X_k^i,U_k^i\right)}\sum_{j=0}^{k-1}\left[B_j^k\right]^i dU_j^i \\ &+\left[\frac{\partial g_k^{l_k}}{\partial U_k}\right]^T_{\left(X_k^i,U_k^i\right)}dU_k^i\le -g_k^{l_k}\left(X_k^i,U_k^i\right). \end{aligned} \tag{56}$$

Equation (56) can be written in the compact form as

$$B_C^i\delta U^i\le g^i \tag{57}$$

where the elements of $B_C^i=\left[b_{C_{kj}}^i\right]$ and g^i are defined as

$$b_{C_{kj}}^i\triangleq\begin{bmatrix}\left[\frac{\partial g_k^0}{\partial X_k}\right]^T_{\left(X_k^i,U_k^i\right)}\left[B_j^k\right]^i+\delta_{jk}\left[\frac{\partial g_k^0}{\partial U_k}\right]^T_{\left(X_k^i,U_k^i\right)}\\ \vdots\\ \left[\frac{\partial g_k^{l_k}}{\partial X_k}\right]^T_{\left(X_k^i,U_k^i\right)}\left[B_j^k\right]^i+\delta_{jk}\left[\frac{\partial g_k^{l_k}}{\partial U_k}\right]^T_{\left(X_k^i,U_k^i\right)}\end{bmatrix}, \tag{58}$$

$$g^i \triangleq \begin{bmatrix} -g_k^0\left(X_k^i, U_k^i\right) \\ \vdots \\ -g_k^{l_k}\left(X_k^i, U_k^i\right) \end{bmatrix}. \tag{59}$$

Equations (45), (54) and (57) form a quadratic programming (QP) problem, which it can be solved using standard QP solvers to obtain the control update vector δU^i. In the present paper, *Hildreth's quadratic programming algorithm* (Luenberger 1969; Wismer and Chattergy 1978) has been used to solve the posed QP problem because it is known to be computationally efficient (Wang 2009). Hence, it is suitable for real-time applications.

Convergence Analysis of C-MPSP

Only the general form of the algorithm, i.e., the constrained case discussed in Sect. 2.4 is considered here as the unconstrained case is a special case of the the constrained case. An existing convergence result related to a general iterative algorithm is first presented. The C-MPSP algorithm is then related to this result to establish its convergence behavior. It is shown that the sequence of iterations leads to a locally optimal solution if the initial iterate is sufficiently close to that solution.

[1]Moreover, an analysis of the convergence rate is also carried out, which leads to the conclusion that the iteration process goes to the optimal solution rapidly in a sense as specified below.

Existing Convergence Result for a General Algorithm

The result in this section shows *Q-linear convergence* of a general iterative algorithm, which is defined as follows.

Definition 1 (*Q-linear convergence (Nocedal and Wright 2006)*) Let $\{U^i\}_{i\in\mathbb{N}}$ be a sequence converging to U^*. The convergence is said to be Q-linear if there exists a constant $0 < r < 1$ such that

$$\lim_{i\to\infty} \frac{\left\|U^{i+1} - U^*\right\|}{\left\|U^i - U^*\right\|} < r. \tag{60}$$

To present the existing results, we consider a general nonlinear program (NLP) defined below.

$$\min_{U\in\Re^q} J(U) \text{ subject to } Y_N(U) = Y_N^*, \text{ and } g(U) \le \mathbf{0}, \tag{61}$$

where the functions $J : \Re^q \to \Re$, $Y_N : \Re^q \to \Re^p$, $g : \Re^q \to \Re^l$. Using a vector of slack variables $Z \in \Re^l$ to handle the nonpositive constraint on $g(U)$, we arrive at

$$\begin{aligned} &\min_{U\in\Re^q, Z\in\Re^l} J(U) \text{ subject to } \quad Y_N(U) = Y_N^*, \\ &g(U) + Z = \mathbf{0}, \quad \text{and} \quad Z \ge \mathbf{0}. \end{aligned} \tag{62}$$

Suppose that the NLP is solved using a general iterative procedure based on its extended Lagrange function (Boggs and Tolle 2000), defined as

$$\mathcal{L}(U, Z, \lambda, \rho) = J(U) + Y_N(U)^{\mathrm{T}}\lambda + [g(U) + Z]^{\mathrm{T}}\rho \in \Re, \tag{63}$$

where $\lambda \in \Re^p$ and $\rho \in \Re^l$ are the Lagrange multipliers corresponding to the two constraints in (62). In every iteration, the algorithm updates the variables of optimization U and Z as well as the Lagrange multipliers λ and ρ. Let the estimate of variables U, Z, λ, and ρ in the ith iteration be denoted as given U^i, Z^i, λ^i, and ρ^i, respectively. In the $(i+1)$th iteration, the variables are updated as follows:

$$\begin{bmatrix} U^{i+1} \\ Z^{i+1} \\ \lambda^{i+1} \\ \rho^{i+1} \end{bmatrix} = \begin{bmatrix} U^i \\ Z^i \\ \lambda^i \\ \rho^i \end{bmatrix} + \alpha^i d^i, \tag{64}$$

where $\alpha^i \in \Re$ is a scalar and $d^i \in \Re^{q+2l+p}$ denotes the direction in which current estimates deviate from the estimates in the previous iteration. Let (U^*, Z^*) be a feasible solution of the NLP and (λ^*, ρ^*) be the corresponding optimal Lagrange multiplier vectors. The convergence result establishes conditions under which the iterates converge to a feasible solution. To state the result, the following definitions are needed:

- Perturbation vectors w^i and p^i, for $i = 1, 2, \ldots$:

[1] The requirement that the initial guess be close to the optimal solution is typical of theoretical guarantees for any iterative algorithm. In our observation from a number of simulations under various initial conditions and control history guesses, we found that the algorithm is relatively insensitive to it, which means it converges to the desired solution even if the initial guess was far off. The reader is referred to the simulation results section for more details. We also note that, if one uses an interior point method (which has fast convergence properties) in the optimization process, the requirement is to start with a 'feasible' solution satisfying all constraints. However, in practice, this is a difficult task. To avoid this problem, one can start with sequential quadratic programming, which does not require initialization at a feasible point. Once all the constraints are satisfied, thus making the solution feasible, one can switch over to the interior point method to speed up the process.

$$w^i \triangleq \begin{bmatrix} U^i - U^* \\ Z^i - Z^* \\ \lambda^i - \lambda^* \\ \rho^i - \rho^* \end{bmatrix} \in \Re^{q+2l+p}, \tag{65}$$

$$p^i \triangleq \begin{bmatrix} \nabla^2 J(U^i) & 0 & \nabla Y_N(U^i) & \nabla g(U^i) \\ \nabla Y_N(U^i)^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \nabla g(U^i)^{\mathrm{T}} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D^i_\rho & \mathbf{0} & D^i_Z \end{bmatrix} d^i + \begin{bmatrix} \nabla J(U^i) \\ Y_N(U^i) \\ g(U^i) + Z^i \\ D^i_\rho Z^i \end{bmatrix} \in \Re^{q+21+p} \tag{66}$$

where $D^i_\rho \in \Re^{l\times l}$ and $D^i_Z \in \Re^{l\times l}$ are diagonal matrices with ρ^i and Z^i along the diagonal, respectively.

- $\mathcal{A} \triangleq \{j : g_j(U^*) = 0\}$, the set of active inequality constraints where $g_j(U^*) \in \Re$ is the j^{th} component of $g(U^*)$.
- $G(U^*) \triangleq \left[\nabla Y_N(U^*) \;\; \nabla g_{\mathcal{A}}(U^*)\right] \in \Re^q \times \Re^{p+|\mathcal{A}|}$, a matrix whose columns are the gradients of the equality and active inequality ($g_j = 0$) constraints at U^*.
- $\nabla^2\mathcal{L}^i$ are the Hessian of the extended Lagrangian $\mathcal{L}$ in (63) with respect to U, evaluated at $(U^i, Z^i, \lambda^i, \rho^i)$.

The stage is now set to state the convergence result for the above iterative algorithm, which is as follows:

Theorem 1 (Boggs and Tolle (2000), Theorem 4.1) *Suppose that the following conditions hold:*

C1 *All of the functions in the NLP have Lipschitz continuous second derivatives.*

C2 *The feasible solution satisfies*

a. *For* $j \in \mathcal{A}$, $\rho^*_j > 0$.
b. *The matrix* $G(U^*)$ *has full column rank.*
c. *For all* $y \in \Re^q$ *such that* $y \neq \mathbf{0}$ *and* $G(U^*)^{\mathrm{T}} y = \mathbf{0}$, $y^{\mathrm{T}}\nabla^2\mathcal{L}^* y > 0$.

C3 $\lim_{i\to\infty} \alpha^i = 1$.

C4 *The perturbation* p^i *satisfies* $\left\|p^i\right\| = o\left(\left\|w^i\right\| + \left\|d^i\right\|\right)$.

C5 *The sequence* $\left\{\nabla^2 J\left(U^i\right)\right\}_{i\in\mathbb{N}}$ *satisfies*

a. *For each i, the matrix* $\nabla^2 J\left(U^i\right)$ *satisfies the condition:* $y \neq \mathbf{0}$ *and* $G(U^i)^{\mathrm{T}} y = \mathbf{0}$ *implies* $y^{\mathrm{T}}\nabla^2 J\left(U^i\right) y > 0$.
b. *There exist constants* η_1 *and* η_2 *such that*

$$\left\|\nabla^2 J\left(U^{i+1}\right) - \nabla^2\mathcal{L}^i\right\| \leq \eta_2\sigma^i + \left(1 + \eta_1\sigma^i\right)\left\|\nabla^2 J\left(U^i\right) - \nabla^2\mathcal{L}^i\right\|, \tag{67}$$

where $\sigma^i = \mathcal{O}\left(\left\|w^{i+1}\right\| + \left\|w^i\right\|\right)$.

Then, there exists $\epsilon > 0$ *such that if* $\left\|w^0\right\| < \epsilon$, *and* $\left\|\nabla^2 J\left(U^0\right) - \nabla^2\mathcal{L}^*\right\| < \epsilon$, *the sequence* $\left\{w^i\right\}_{i\in\mathbb{N}}$ *converges to* $\mathbf{0}$, *and* $\left\{(U^i, Z^i)\right\}_{i\in\mathbb{N}}$ *converges Q-linearly to* (U^*, Z^*). *Here,* $\nabla^2\mathcal{L}^*$ *is the Hessian of the extended Lagrangian* $\mathcal{L}$ *in (63) with respect to U, evaluated at* $(U^*, Z^*, \lambda^*, \rho^*)$.

Next, the above theorem is applied to the C-MPSP algorithm proposed in this paper. To this end, first, the algorithm needs to be rewritten slightly differently, as explained in the following subsection.

Matrix Notation

This section reformulates the C-MPSP algorithm as the iterative algorithm described in Theorem 1. From (1) and (2), it is clear that

$$\begin{aligned} Y_N &= h_N(X_N, U_N) = h_N\left(F_{N-1}(X_{N-1}, U_{N-1}), U_N\right) \\ &= h_N\left(F_{N-1}\left(\ldots F_1(X_0, U_0), \ldots, U_{N-1}\right), U_N\right) \end{aligned} \tag{68}$$

Since it is assumed that the knowledge of the initial state X_0 as well as the system dynamics is available, Y_N is a function of the control history $\{U_k\}_{k=0}^N$ alone. Let the set of unknowns be denoted by U:

$$U \triangleq \left[U_0^{\mathrm{T}} \;\; U_1^{\mathrm{T}} \;\; \ldots \;\; U_N^{\mathrm{T}}\right]^{\mathrm{T}} \in \Re^q \tag{69}$$

where $q \triangleq (N+1)m$. Then, $Y_N : \Re^q \to \Re^p$ is a function of U. Similarly, it can also be shown that the cost function J is a function of U alone.

Next, a new function $g : \Re^q \to \Re^l$ with $l = \sum_{k=0}^N l_k$ is defined to denote the inequality constraints associated with the problem as defined in (13). Thus, the constraints can be denoted as $g(U) \leq \mathbf{0}$, where $(\sum_{i=0}^{k-1} l_i + j)^{th}$ entry of g is g_k^j, for $j = 0, 1, \ldots, l_k$ and $k = 1, 2, \ldots, N-1$. Following the above notation, the problem of finding the optimal control inputs as an optimization problem that solves for the unknown vector U is now rewritten. For doing this, U^i is defined as the iterate values (set of control inputs) computed in the $_i{}^{th}$ iteration of the C-MPSP algorithm:

$$U^i \triangleq \left[U_0^{i\mathrm{T}} \;\; U_1^{i\mathrm{T}} \;\; \ldots \;\; U_N^{i\mathrm{T}}\right]^{\mathrm{T}} \in \Re^q. \tag{70}$$

Then, from (7), one gets the following:

$$\delta U^i = U^{i+1} - U^i, \tag{71}$$

$$\Delta Y_N^i = Y_N(U^{i+1}) - Y_N(U^i) \tag{72}$$

The algorithm's convergence can be established by characterizing the convergence of the sequence $\{U^i\}_{i\in\mathbb{N}}$. So, the first step towards proving the convergence is to rewrite the C-MPSP algorithm in terms of U as given by the following lemma. Here, for brevity, the operations $\frac{\partial}{\partial U}$ and $\frac{\partial^2}{\partial U^2}$ are denoted by ∇ and ∇^2, respectively.

Lemma 1 *The quadratic subproblem solved by the C-MPSP algorithm in the* i^{th} *iteration, which is described by (45), (54) and (57), can be written as follows:*

$$\begin{aligned}
&\min_{U\in\Re^q, Z\in\Re^l} \frac{1}{2}\left(U-U^i\right)^{\mathrm{T}}\nabla^2 J(U^i)\left(U-U^i\right)\\
&\quad + \nabla J(U^i)^{\mathrm{T}}\left(U-U^i\right) + J(U^i)\\
&\text{subject to}\\
&\quad Z + g(U^i) + \nabla g(U^i)^{\mathrm{T}}\left(U-U^i\right) = \mathbf{0},\\
&\quad Y_N^{*^i} + \nabla Y_N(U^i)^{\mathrm{T}}\left(U-U^i\right) = \mathbf{0}, \quad \text{and} \quad Z \geq 0.
\end{aligned} \tag{73}$$

Proof From (18) and (46), $\left[B^k\right]^i = \nabla X_k(U^i)$. Therefore,

$$\begin{aligned}
\left([B]^i\right)^T \left(L_{XX}^i\right)[B]^i &= \sum_{k=1}^{N}\left(\left[B^k\right]^i\right)^T\left[\partial_{X_k}^2 L_k\right]_k^i\left[B^k\right]^i\\
&= \sum_{k=1}^{N}\left(\nabla X_k(U^i)\right)^{\mathrm{T}}\left[\partial_{X_k}^2 L_k\right]_k^i \nabla X_k(U^i)
\end{aligned} \tag{74}$$

where (74) follows from (47) and (49). Similarly, from (48), (50)-(52), one can show that

$$\left(L_X^i\right)^T[B]^i = \partial_{X_k} L_k\big|_{X_k=X_k^i, U_k=U_k^i} \tag{75}$$

$$\left(L_U^i\right)^T = \partial_{U_k} L_k \tag{76}$$

$$\left([B]^i\right)^T L_{XU}^i = \sum_{k=1}^{N}\nabla X_k(U^i)\left(\partial_{X_k U_k} L_k\right) \tag{77}$$

$$L_{UX}^i[B]^i = \sum_{k=1}^{N}\left(\partial_{U_k X_k} L_k\right)\nabla X_k(U^i) \tag{78}$$

$$L_{UU}^i = \sum_{k=1}^{N}\left(\partial_{U_k}^2 L_k\right) \tag{79}$$

Substituting (74)–(79) in (45), and using (3), it can be deduced that (45) is equivalent to

$$\begin{aligned}
J^i = &\frac{1}{2}\left(U^{i+1}-U^i\right)^{\mathrm{T}}\nabla^2 J(U^i)\left(U^{i+1}-U^i\right)\\
&+ \nabla J(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right) + J(U^i)
\end{aligned} \tag{80}$$

Further, using (23), (58) and (59), it is clear that (54) and (57) are equivalent to

$$g(U^i) + \nabla g(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right) \leq \mathbf{0} \tag{81}$$

$$Y_N(U^i) - Y_N^* + \nabla Y_N(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right) = 0. \tag{82}$$

Finally, a slack variable $Z \in \Re^l$ is introduced to rewrite (81) as

$$g(U^i) + \nabla g(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right) + Z = \mathbf{0}, \quad Z \geq \mathbf{0} \tag{83}$$

Thus, the optimization problem in (73) is obtained which is equivalent to the optimization problem described by (45), (54) and (57). Hence, the proof is complete. □

With this, the convergence result for the proposed C-MPSP algorithm can be stated, which is discussed next.

Convergence of C-MPSP

To prove the convergence, the definitions introduced in Sect. 3.1 are used. It is also assumed that the quadratic subproblem solved by the C-MPSP algorithm in each iteration is solved exactly. Further, to connect the C-MPSP algorithm to Theorem 2, the following is defined:

$$d^i = \delta U^i = \begin{bmatrix} U^{i+1}-U^i \\ Z^{i+1}-Z^i \\ \lambda^{i+1}-\lambda^i \\ \rho^{i+1}-\rho^i \end{bmatrix}, \text{ and let } \alpha_i = 1 \tag{84}$$

Thus, the algorithm update given in (71) is equivalent to algorithm update (64) given in Sect. 3.1. Therefore, the following result holds for every iteration:

Lemma 2 *Suppose that (73) is solved exactly in every iteration. The feasible solution* (U^i, Z^i) *is such that a regular point for the constraints, i.e., the matrix whose columns are the gradients of the equality and active inequality constraints of (73) at* U^i*, has full column rank. Then,* (U^i, Z^i) *satisfies* $p^i = 0$ *for some vector* $(\lambda^i \in \Re^q, \rho^* \in \Re^l)$*, where* p^i *is defined in (66).*

Proof It is known that a feasible solution to an optimization problem always satisfies the first-order necessary condition for optimality, i.e., the derivative of the Lagrange function vanishes at the solution. Hence, there exists a sequence $\left\{\lambda^i \in \Re^q, \rho^* \in \Re^l\right\}_{i\in\mathbb{N}}$ such that the following holds:

$$
\begin{aligned}
0 = \frac{\partial}{\partial U}\Big\{ & \frac{1}{2}\left(U-U^i\right)^{\mathrm{T}} \nabla^2 J(U^i)\left(U-U^i\right) \\
& + \nabla J(U^i)^{\mathrm{T}}\left(U-U^i\right) + J(U^i) \\
& + \left[Z + g(U^i) + \nabla g(U^i)^{\mathrm{T}}\left(U-U^i\right)\right]^{\mathrm{T}}\left(\rho^{i+1}-\rho^i\right) \\
& + \left[Y_N^{*^i} + \nabla Y_N(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right)\right]^{\mathrm{T}}\left(\lambda^{i+1}-\lambda^i\right)\Big\}_{U=U^{i+1}} \\
= {} & \nabla^2 J(U^i)\left(U^{i+1}-U^i\right) \\
& + \nabla J(U^i) + \nabla Y_N(U^i)\left(\lambda^{i+1}-\lambda^i\right) + \nabla g(U^i)\left(\rho^{i+1}-\rho^i\right)
\end{aligned} \tag{85}
$$

Note that the Lagrange multipliers are $\lambda^{i+1}-\lambda^i \in \Re^q$ and $\rho^{i+1}-\rho^i \in \Re^l$. Similarly, due to the constraint $Z \geq 0$, the first-order optimality also guarantees the following Boggs and Tolle (2000):

$$
D_\rho^i Z^{i+1} + D_Z^i\left(\rho^{i+1}-\rho^i\right) = \mathbf{0} \tag{86}
$$

Further, since the solution is feasible, it satisfies

$$
Y_N^{*^i} + \nabla Y_N(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right) = \mathbf{0} \tag{87}
$$

$$
Z^{i+1} + g(U^i) + \nabla g(U^i)^{\mathrm{T}}\left(U^{i+1}-U^i\right) = \mathbf{0} \tag{88}
$$

Substituting (85), (86), (87) and (88) into the definition of p^i in (66), the desired result is obtained. □

Using the above lemmas, the following result gives the desired convergence result for the C-MPSP algorithm.

Theorem 2 *Suppose that the assumptions of Lemma 2 hold, and the following conditions are satisfied:*

(i) *The functions J, Y_N and g have Lipschitz continuous second derivatives.*
(ii) *The Hessian $\nabla^2 J$ of function J is positive definite for all values of U.*
(iii) *The feasible solution satisfies the following:*
  (a) *For $j \in \mathcal{A}$, $\rho_j^* > 0$.*
  (b) *The matrix $G(U^*)$ has full column rank.*
  (c) *For all $y \in \Re^q$ such that $y \neq \mathbf{0}$ and $G(U^*)^{\mathrm{T}} y = \mathbf{0}$, $y^{\mathrm{T}} \nabla^2 \mathcal{L}^* y > 0$.*

Then, there exists an $\epsilon > 0$ such that if $\left\|w^0\right\| < \epsilon$ and $\left\|\nabla^2 J\left(U^0\right) - \nabla^2 \mathcal{L}^*\right\| < \epsilon$, the sequence $\left\{(U^i)\right\}_{i\in\mathbb{N}}$ converges Q-linearly to U^*. Here, $\nabla^2 \mathcal{L}^*$ is the Hessian of the extended Lagrangian $\mathcal{L}$ in (63) with respect to U, evaluated at $(U^*, Z^*, \lambda^*, \rho^*)$.

Proof Lemma 1 shows that the C-MPSP algorithm can be rewritten such that Theorem 1 holds. Thus, to prove the result, it suffices to show that Conditions C1 to C5 of Theorem 1 are satisfied by the C-MPSP algorithm, which is verified next.

Conditions C1 and C2 are satisfied since they are equivalent to Conditions (i) and (ii) of Theorem 2, respectively. Next, Condition C3 holds because of (84). Also, Lemma 2 implies that $\left\|p^i\right\| = 0$ and thus, Condition C4 holds. Further, Condition C5a holds due to (iii) of Theorem 2. Finally, Condition C5b also holds since

$$
\begin{aligned}
& \left\|\nabla^2 J\left(U^{i+1}\right) - \nabla^2 \mathcal{L}^i\right\| \\
& \leq \left\|\nabla^2 J\left(U^{i+1}\right) - \nabla^2 J\left(U^i\right)\right\| + \left\|\nabla^2 J\left(U^i\right) - \nabla^2 \mathcal{L}^i\right\| \\
& \leq \tau\left\|U^{i+1}-U^i\right\| + \left\|\nabla^2 J\left(U^i\right) - \nabla^2 \mathcal{L}^i\right\|
\end{aligned} \tag{89}
$$

where τ is the Lipschitz constant of $\nabla^2 J$ as assumed in (i). Hence, the proof is complete. □

An important implication of the above result is as follows. Q-linear convergence with rate of convergence r (as given in Definition 1) implies that there exists a constant $\gamma > 0$ and $i^* \in \mathbb{N}$ such that for all $i > i^*$,

$$
\lim_{i\to\infty}\left\|U^i - U^*\right\| \leq \gamma r^i. \tag{90}
$$

Thus, Theorem 2 shows that to achieve convergence within an error ζ, (i.e., $\left\|U^{i+1}-U^*\right\| \leq \zeta$), our algorithm requires at most $\frac{\log \epsilon/\gamma}{\log r}$ iterations. In other words, the error $\left\|U^{i+1}-U^*\right\|$ decreases exponentially in the number of iterations ensuring a faster convergence. Our numerical experiments also corroborate these observations (see Sect. 5.6). The next section, Sec. 4, summarizes the C-MPSP algorithm and its implementation steps.

Implementation steps of C-MPSP

The procedure to implement the C-MPSP is as follows:

1. Start the iterations with a guess control history U_k^0, $\forall k = 0, 1, \ldots, N$; and set the iteration index to zero ($i = 0$). Set all the tuning parameters of the cost function $L_k, \forall k = 0, 1, \ldots N$ to convenient values.
2. Use the known initial condition X_0 and the control input $\left\{U_0^i, U_1^i, \ldots, U_{N-1}^i\right\}$ to propagate the system dynamics (8) and obtain the predicted state trajectory X_k^i, $\forall k = 0, 1, \ldots, N$, and the output at the final time step Y_N^i. Use the desired output at the final time step Y_N^* to calculate $\Delta Y_N^* = Y_N^i - Y_N^*$.
3. Terminate the algorithm if $i \geq 1$ and the output error is smaller than the user-defined tolerance values $(\varepsilon_Y, \varepsilon_U)$, i.e., $\| \Delta Y_N^*\|_2 / \| Y_N^*\|_2 < \varepsilon_Y$ and the control history has converged, i.e., $\sum_{k=0}^{N} \| U_k^i - U_k^{i-1}\|_2 \Big/ \sum_{k=0}^{N} \| U_k^{i-1}\|_2 < \varepsilon_U$.

If these conditions are met for $i \geq 1$, then use U_k^i as the suboptimal control history. Otherwise continue through Steps 4 to 7.

4. Using X_k^i and U_k^i (for all $k = 0, 1, \ldots, N$), calculate the sensitivity matrices given by (23), the Jacobian matrices $[\partial h_N/\partial X_N]_{(X_N^i, U_N^i)}$, $[\partial h_N/\partial U_N]_{(X_N^i, U_N^i)}$, $[\partial G_k/\partial X_k]_{(X_k^i)}$, and the Hessian matrix $\left[\partial^2 G_k/\partial X_k^2\right]_{(X_k^i)}$.
5. Using sensitivity and Jacobian matrices in Step 4, compute matrices $A_{UU}, A_{U\lambda}, A_{\lambda U}, A_{\lambda\lambda}$ using Eqs. (36)–(37). Thus, we have the coefficient matrix of the linear system of equations in (35). The vectors b_U, and b_λ of the linear system (35) are obtained using (38).
6. Solve the linear system of equation obtained in Step 5 (using (35)) to obtain the optimal control deviations $dU_k^i, \forall k = 0, 1, \ldots N$.
7. Compute the updated control U_k^{i+1} using (43). Increase the iteration index *i* and return to Step 2.

It may be noted that $R_k, \forall k = 0, 1, \ldots, N$ is the control weightage matrix and it has to be given by the designer; otherwise a default value of $R_k = I_{r\times r}$ can be used. This remains a constant during the iteration process, hence the superscript *i* has not used on matrix R_k.

The guidance algorithm has to run at each time step to obtain the converged optimal solution. For the very first time, the guess history is generated using any of the available methods (the guess control history can be chosen arbitrarily). Thereafter, the guess history is obtained from the previous time step's converged history. In this manner, the proposed MPSP based guidance algorithm eventually becomes independent of the initial guess.

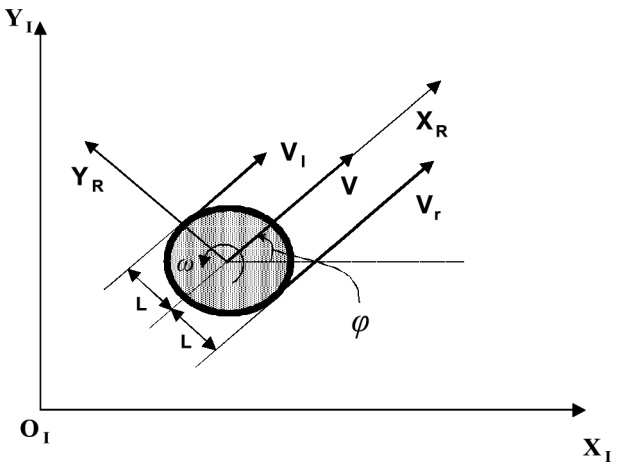

Fig. 1 Differential wheel drive two-wheel mobile robot

A Demonstrative Example

The performance of the proposed C-MPSP algorithm is demonstrated by solving a guidance problem for a two-wheeled differential drive mobile robot. The robot must pass through a pre-specified desired final position at a given final time, without violating the constraints on state and control inputs on the way.

Plant Model and Output Equation

The kinematic model of a two-wheel differential drive mobile robot, as shown in Fig. 1, can be written as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \left(\frac{\omega_r+\omega_l}{2}\right) r\cos\phi \\ \left(\frac{\omega_r+\omega_l}{2}\right) r\sin\phi \\ \left(\frac{\omega_r-\omega_l}{2L}\right) r \end{bmatrix}, \tag{91}$$

where x and y are positions of the centre of the mobile robot in the fixed frame of reference $X_I O_I Y_I$, ϕ is the orientation angle of the vehicle with respect to the x-axis of the fixed frame of reference, ω_r is the angular velocity of the right wheel, and ω_l is the angular velocity of the left wheel. Also, r is the wheel radius and L is the wheel hinge's distance from the mobile robot's center. The selected values of the mobile robot parameters are $r = 35mm$ and $L = 55mm$. From (91), the state and control vectors are defined as $X \triangleq [x\ y\ \phi]^T$ and $U \triangleq [\omega_r\ \omega_l]^T$. It is required that, starting from a feasible point (x_0, y_0) at $t = 0s$, where $x_0 = 0$ and $y_0 \in (-11, -9)$, the robot needs to pass through the desired point $(x_f^*, y_f^*) = (10, 0)$ at the desired final time $t_f = 10s$. Defining the output vector at k^{th} time step as $Y_k = h_k(X_k, U_k) \triangleq [x(t_k)\ y(t_k)]^T = [x_k\ y_k]^T$ and at the final time step N as $Y_N^* \triangleq [x_f^*\ y_f^*]^T$, it is required that $Y_N = Y_N^*$.

Path Constraint

Without loss of generality, the path's shape is considered to be an annular space between arcs of two concentric circles. Accordingly, the constraint on the states at k^{th} time step due to the path can be written as

$$r_i^2 \leq (x_k - x_{c_p})^2 + (y_k - y_{c_p})^2 \leq r_0^2\ , \tag{92}$$

for $k = 0, 1, \ldots, N$, where (x_{c_p}, y_{c_p}) is the center of the circle making the annular path, r_i is the radius of the inner circle, and r_o is the radius of the outer circle of the path. In the present simulation, the values of the parameters of the annular path are $(x_{c_p}, y_{c_p}) = (0, 0)$, $r_i = 9m$ and $r_o = 11m$. All these constraints can be seen in Fig. 2.

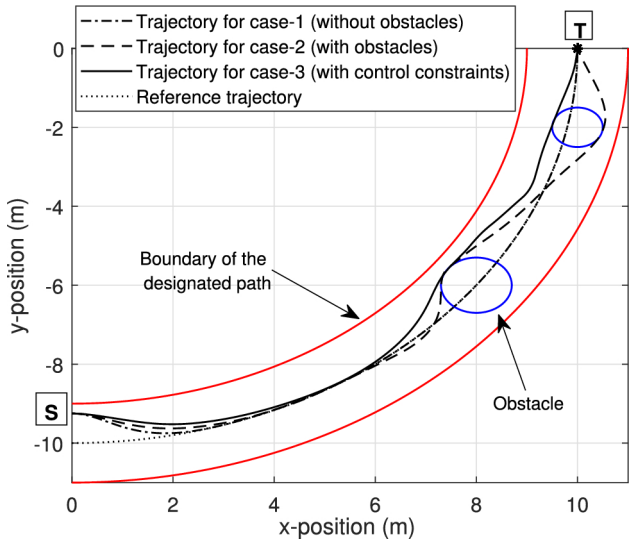

Fig. 2 Mobile robot trajectory for three different cases

Modeling of Obstacles

Without loss of generality, static circular obstacles of varying radii are considered on the path. These lead to additional constraints on the states, which can be written as

$$(x_k - x_{c_i})^2 + (y_k - y_{c_i})^2 \geq r_{o_i}^2 \,, \tag{93}$$

where (x_{c_i}, y_{c_i}) is the center of the i^{th} obstacle, r_{o_i} is the radius of the i^{th} obstacle. In the simulation, two obstacles are considered, and the values of the parameters of these obstacles are chosen as $(x_{c_1}, y_{c_1}) = (8, -6)$, $r_1 = 0.7m$ and $(x_{c_2}, y_{c_2}) = (10, -2)$, $r_2 = 0.5m$. These obstacles are shown in Fig. 2.

Control Input Constraints

To make the simulation more realistic, constraints on the control input are also considered as

$$|\omega_{r_k}| \leq \omega_{max} \quad \text{and} \quad |\omega_{l_k}| \leq \omega_{max} \,, \tag{94}$$

where ω_{r_k} and ω_{l_k} are the wheel rotation rate of the right and left wheels at k^{th} time step, and ω_{max} is the maximum allowed wheel rotation rate (set to $51 rad/s$ here).

Cost Function

With the motivation that the robot remains in the middle of the designated path as much as possible, and the updated control history does not drift away from the existing value, the cost function at the i^{th} iteration is constructed as

$$J^i = \frac{1}{2}\sum_{k=0}^{N} q_k \left(r_k^{i+1} - r_p\right)^2 + \frac{1}{2}\sum_{k=0}^{N} (U_k^{i+1} - U_k^i)^T R_k (U_k^{i+1} - U_k^i), \tag{95}$$

where $r_k^{i+1} = \sqrt{\left(x_k^{i+1} - x_{c_p}\right)^2 + \left(y_k^{i+1} - y_{c_p}\right)^2}$ is the distance of the mobile robot from the center of the circle of the annular path and $r_p = \frac{1}{2}(r_i + r_o)$ is the radius of the middle of the path. A tuning parameter q_k is included in the cost function with the usual implication, i.e., a higher q_k ensures that the mobile robot stays in the middle of the path, while a lower q_k gives the mobile robot more freedom to move farther from the middle of the path. In the simulation, q_k is set to 15. The weighting matrix on the control vector is $R_k = I_2$.

Simulation Results

Three cases are considered in the increasing order of complexity: (i) *Case 1: path constraint without obstacles*, (ii) *Case 2: path constraint with obstacles*, and (iii) *Case 3: path and control input constraints*. In Case 1, the trajectory constraints as given in (4) are obtained from (92). For Case 2, the trajectory constraints are obtained from (92) and (93), and similarly, for Case 3, they are obtained from (92), (93) and (94). Note that all these three cases are subject to same system dynamics in (91) and employ the same cost function given by (95). The initial state of the mobile robot is taken as $x(0) = 0m$, $y(0) = -9.25m$, and $\phi(0) = 0^\circ$.

To compute a meaningful guess history of control to start the C-MPSP algorithm, it is assumed that the mobile robot passes on the central line of the circular annular space with known turn radius $r_p = 10m$, with a known constant velocity $v = \left(\frac{\omega_r + \omega_l}{2}\right) r$, and the path is obstacle-free. Under these assumptions, it can easily be shown that $\dot{\phi} = v/r_p$. Hence, assuming constant v and $\dot{\phi}$, the required constant values of the ω_r and ω_l can be computed from the system dynamics (91), which are denoted as ω_r^g and ω_l^g to avoid confusion. Assuming $v = 1.5708 m/s$ (computed with the aim of traveling the quarter-circle distance in $10s$), it turns out that $\omega_r^g = 45.13 rad/s$ and $\omega_l^g = 44.63 rad/s$, which serve as the initial guess control histories at $t = 0s$, and these are used to initialize the algorithm. Beyond the first iteration, the guess control histories are obtained from the updated control history during the previous iteration.

Figure 2 shows the trajectories for the three cases. First, one can observe that in all three cases, the mobile robot can pass the desired final position. In all three cases, the robot can also move toward the center of the path from its initial condition, as expected. It then follows the reference trajectory for the rest of the duration for Case 1, when there is no obstacle present on the path. In the presence of the obstacles, i.e., for Cases 2 and 3, the robot reshapes its trajectory to avoid the obstacles on the path, thereby satisfying the

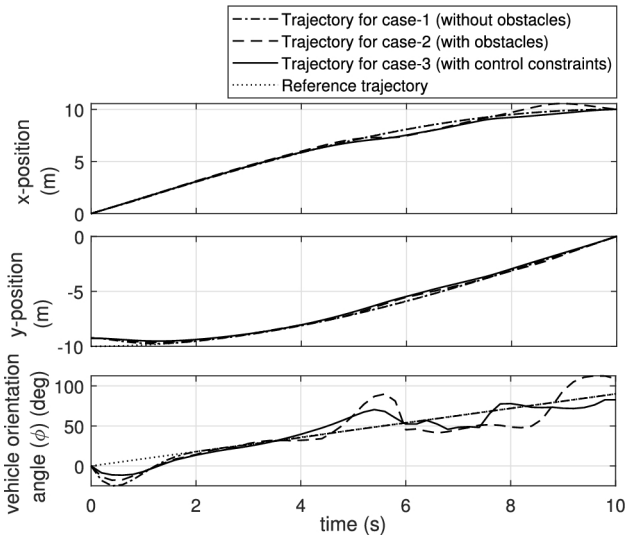

Fig. 3 Mobile robot position and vehicle orientation

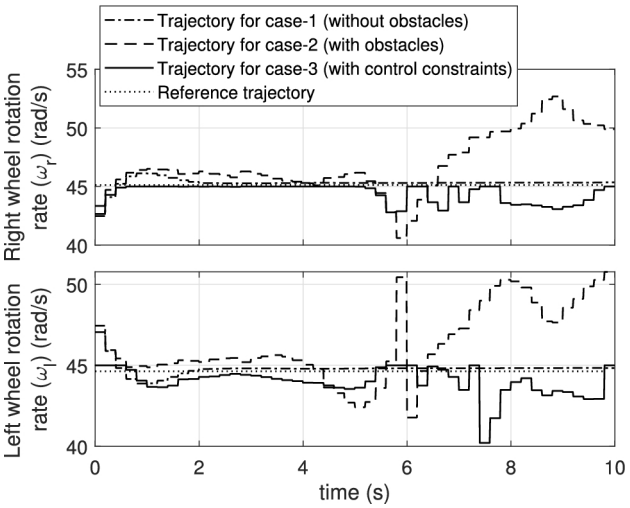

Fig. 4 Wheel rotation rate of mobile robot

imposed path constraint. It can be observed that the trajectory for Case 2 returns to the middle of the path more often and more quickly as compared to Case 3. This is due to the imposed control constraint, which restricts the turning rate of the robot in Case 3.

From the first two sub-plots of Fig. 3, it can be seen that mobile robot reaches the desired final position at the stipulated time interval of 10*sec* with very low error. This is due to the fact that the final position constraints in both the coordinated have been taken as hard constraints. It can also be seen in this figure that the mobile robot does not reach the final position with same ϕ for all cases. This is due to the presence of the second obstacle on the path for Case 2 and Case 3, and the destination happens to be too close to it.

Figure 4 shows the wheel rotation rates. It can be seen that wheel rotation rates of the right and left wheels for Case 1 are nearly constant after 2*s*; this is because by this time, the mobile robot reaches the reference trajectory, and a constant wheel rotation rate is required to maintain it along rest of the reference trajectory. The wheel rotation rates for Case 2 reach the maximum rate of 52.7*rad/s* for the right wheel and 50.4*rad/s* for the left wheel. However, in Case 3, the wheel rotation rate is limited to 45*rad/s*. It can be seen that the right wheel rotation rate is nearly fully saturated all the time (except for $8-10s$). In contrast, the left wheel gets into saturation two times, first during the initial phase, and second when it encounters the first obstacle. It is worth mentioning here that the control commands are computed at every 0.2 *s* interval following the zero-order hold philosophy.

Algorithm Convergence

The algorithm's convergence is illustrated here to validate Sect. 2. First, it is required to see whether the conditions of Sect. 2 hold good in this case, which is done as follows:

- From the definition of the output variable, as well as (94) and (95), it is obvious that the functions $J, Y_N,$ and g satisfy Condition (i).
- Since $\nabla^2 J = R_k, \quad R_k$ has been chosen to be positive definite to ensure that Condition (ii) holds, which is compatible with the control minimization requirement.
- From experiments, it is observed that none of the inequality constraints in (94) holds with equality (i.e., $g_j(U^*) \neq 0, \forall j$) and therefore, $\mathcal{A}$ is an empty set. Consequently, $G(U^*) = \nabla h_N(U^*) = \left(\left[S^N\right]_{U^*}\right)^T \in \Re^{2\times 2}$, where S^N is the final output sensitivity matrix. Since U^* is unknown, $G(\cdot)$ is evaluated at the value to which the algorithm iterates converge and it is observed that its rank 2 (i.e., full row rank). Thus, Conditions (iii)(a) and (iii)(b) are satisfied. Finally, since $\text{rank}\{G(U^*)\} = 2$, there exists no vector y such that $G(U^*)^T y = 0$. Therefore, Condition (iii) also holds, thereby ensuring Q-linear convergence, i.e. exponential convergence of control history with number of iterations.

Pictorially, the Q-linear convergence behavior of the algorithm for five perturbed guess control histories in the constraint-free case is shown in Fig. 5, where U^* is the final converged solution. The different (ω_r^g, ω_l^g) used here are (30, 29.5), (35, 34.5), (40, 39.5), (43, 42.5), (45, 44.5) respectively (all in *rad/s*). One can see that each of it shows exponential convergence as expected. Figure 5 also shows that a crude initial guess (with higher $||\Delta Y_N||$) may take a couple of more iterations to converge and vice versa, which

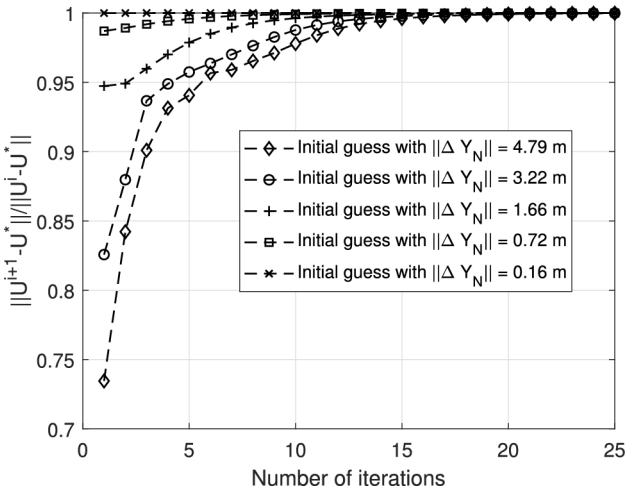

Fig. 5 Simulation convergence for Case-1

Table 1 Initial guess control history for convergence analysis

Sl.	Guess control		Colour
	ω_r (rad/s)	ω_l (rad/s)	
1.	45	44.5	Shown in "black colour" in Figs. 6 and 7
2.	45	45	Shown in "blue colour" in Figs. 6 and 7
3.	18	15	Shown in "magenta colour" in Figs. 6 and 7
4.	44.5	44.5	Shown in "green colour" in Figs. 6 and 7

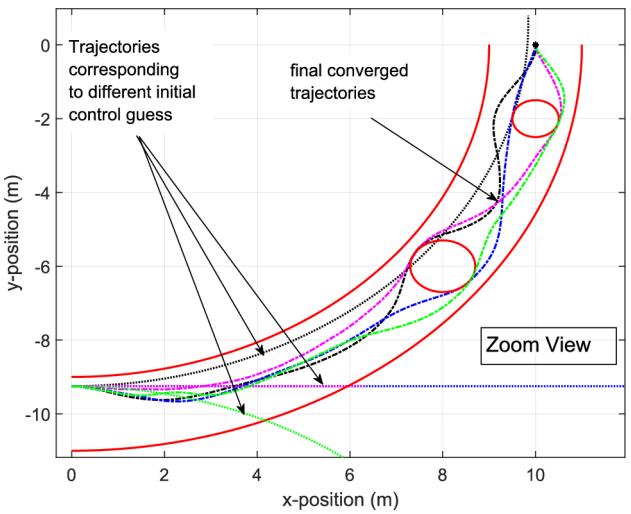

Fig. 6 Mobile robot trajectories corresponding to guess control given in Table 1

is intuitive. To address this and minimize the risk of non-convergence, it is always recommended to start with a good initial control guess history wherever possible.

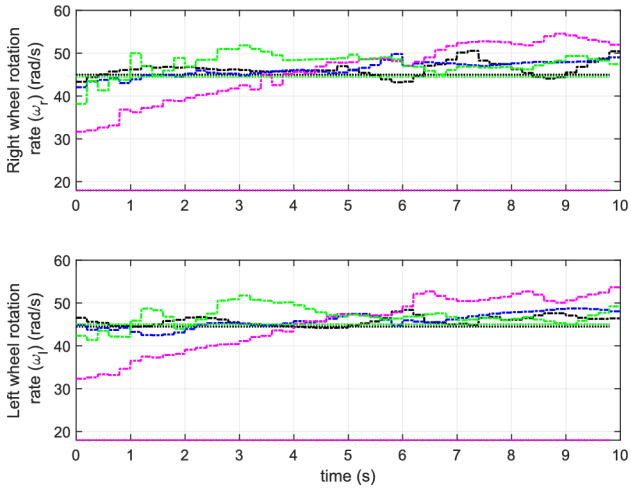

Fig. 7 Mobile robot wheel rotation rates corresponding to guess control given in Table 1

Algorithm Convergence with Different Initial Guess Control Histories

Generally, a good initial guess control history close to the optimal solution is required to provide theoretical guarantees of convergence for any iterative algorithm of optimal control problems. However, in practice, it is not critical to initialize the guess control history very accurately. Moreover, the algorithm's initialisation is highly problem-dependent and a general rule cannot be given for initialization. In this section, the convergence of the C-MPSP algorithm for different initial guess control histories is included. For this, four different initial guess controls have been given to the algorithm, these guess control histories are summarised in Table 1.

It can be seen that in all the cases the algorithm converges and achieves its objectives as shown in Fig. 6. Moreover, it can be seen in Fig. 6 that the final trajectory of the mobile robot is very different in all four cases. This can be explained from Fig. 7, where wheel rotation rates for all four cases are different. The C-MPSP algorithm has been implemented in an iteration unfolding manner (meaning, that maximum number of iterations has been limited to two in each time step). This makes the evolution of the trajectory dependent on the initial guess control; the same has been observed in Fig. 7. Where, it is easy to see that, even though the mobile robot meets all the constraints along the trajectory and the end point constraints; the trajectories with different initial guess control (as given in Table 1) end up in totally different trajectories.

Table 2 Convergence analysis table for different initial conditions

Sl	Initial condition			Guess control		Colour
	x (m)	y (m)	ϕ (deg)	ω_r (rad/s)	ω_l (rad/s)	
1	0	9.25	60	36	36	Shown in "black colour" in Figs. 8 and 9
2	0	9.25	0	36	36	Shown in "blue colour" in Figs. 6 and 7
3	0	9.25	– 60	36	36	Shown in "green colour" in Figs. 6 and 7

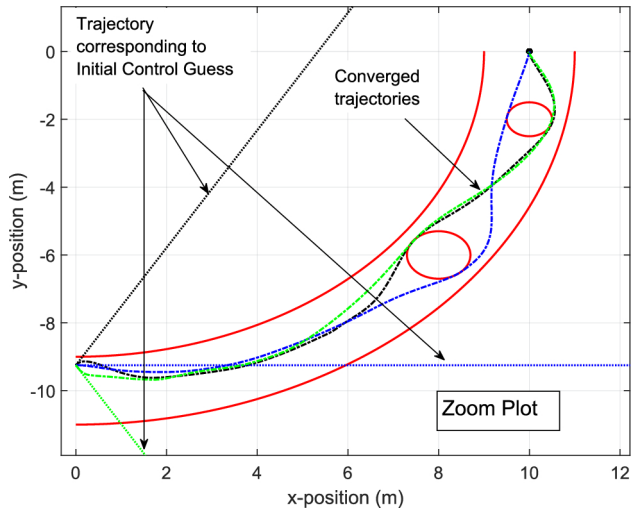

Fig. 8 Mobile robot trajectories corresponding to cases given in Table 2

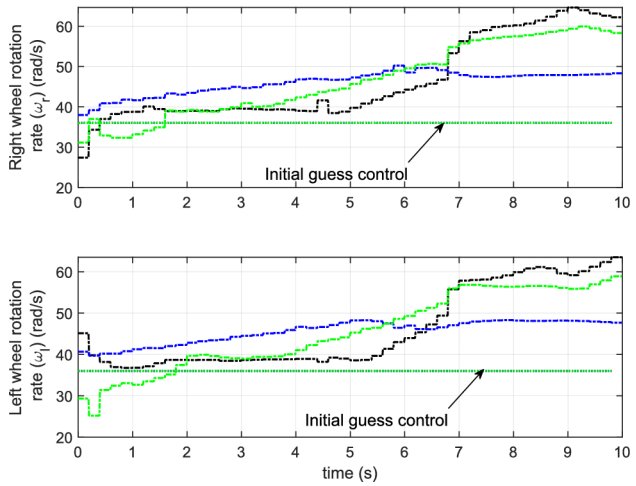

Fig. 9 Mobile robot wheel rotation rates corresponding to cases given in Table 2

Algorithm Convergence for Different Initial States

The convergence of the C-MPSP algorithm not only depends on the initial guess control history but is strongly dependent on the state trajectory. However, the entire state trajectory depends on the initial guess control and the initial states of the system. Because of this reason, the convergence analysis is not complete unless variations in the initial states are also included. Table 2 considers the cases where initial state and initial guess control have been summarised for simulation.

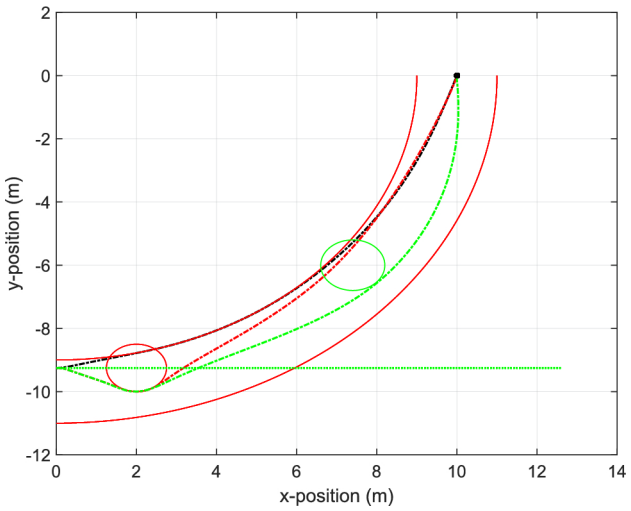

Fig. 10 Mobile robot trajectory with 'zero obstacles' (shown in black colour), 'one obstacle' (shown in red colour), and 'three obstacles' (shown in green colour) (color figure online)

In this simulation, three cases have been considered as given in Table 2. It can be seen in Fig. 8 that the mobile robot can meet all the desired objectives and reach the required final point while satisfying all the path constraints. Moreover, the trajectory followed for each case is again different even though the initial guess control is the same for all three cases. This is because the entire trajectory evolution is strongly associated with the initial states of the system (Fig. 9).

Algorithm Convergence with More Number of Obstacles

One of the aspects of algorithm convergence is the introduction of more path constraints. In the present simulation, this has been achieved by increasing the number of obstacles in the feasible path of the mobile robot. The simulation results in this section has been divided in two sets; the first set (*Simulation results set-1*) shows the results with fewer (upto 2) obstacles, while the second set (*Simulation results set-2*) shows the results with more obstacles (3, 4 and 5). Moreover, the obstacles in *Simulation results set-1* are sparsely placed, while the obstacles in *Simulation results set-2* are placed to densely pack the entire path. The segregation of simulation results has been done to keep the simulation results more legible.

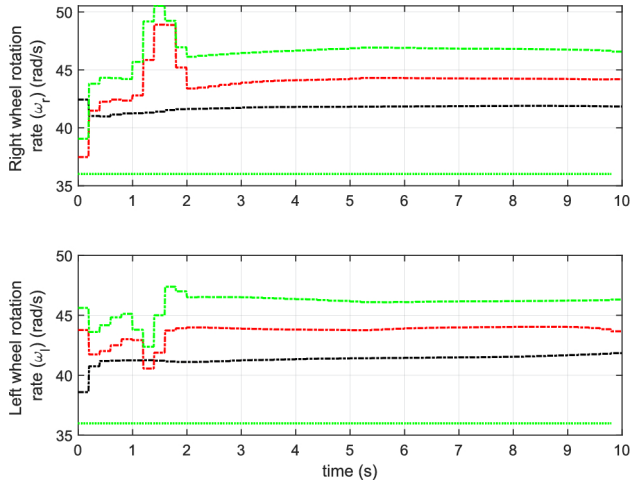

Fig. 11 Mobile robot wheel rotation rate with 'zero obstacles' (shown in black colour), 'one obstacle' (shown in red colour), and 'three obstacles' (shown in green colour) (color figure online)

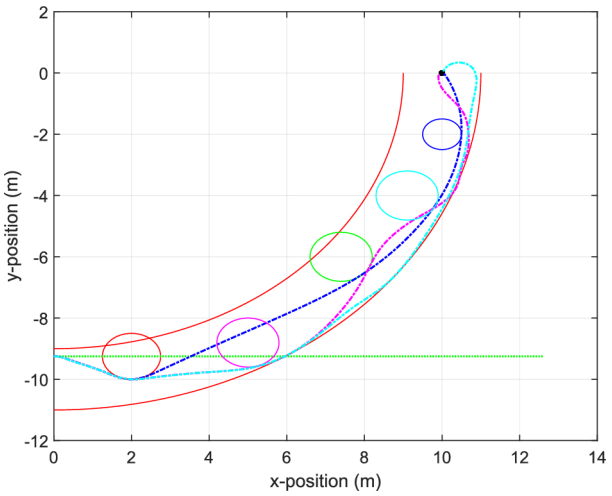

Fig. 12 Mobile robot trajectory with 'three obstacles' (shown in blue colour), 'four obstacle' (shown in magenta colour), and 'five obstacles' (shown in cyan colour) (color figure online)

Simulation results set-1: Figure 10 shows the trajectory of the mobile robot when the number of obstacles is 'zero' (trajectory plots in black colour), 'one' (trajectory plots in red colour), and 'two' (trajectory plots in green colour). The initial guess control history for these three cases is same, and it is shown as a dotted green colour straight line in Fig. 11. Moreover, the mobile robot trajectory corresponding to the initial guess control is a straight line, and it is shown as dotted green straight line in Fig. 10.

Simulation results set-2: Figure 12 shows the trajectory of the mobile robot when the number of obstacles is 'three' (trajectory plots in blue colour), 'four' (trajectory plots in magenta colour), and 'five' (trajectory plots in cyan colour). The initial guess control is same for all these three cases,

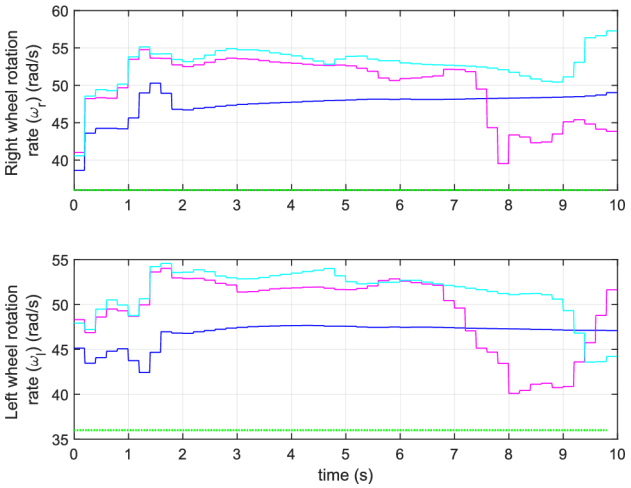

Fig. 13 Mobile robot wheel rotation rate with 'three obstacles' (shown in blue colour), 'four obstacle' (shown in magenta colour), and 'five obstacles' (shown in cyan colour) (color figure online)

and it is shown as green dotted straight lines in Fig. 13. Moreover, the trajectory of the mobile robot corresponding to initial guess control is also shown as a dotted straight line in Fig. 12. It can be seen in Fig. 12, that when the number of obstacles are more, and they are closely spaced; the C-MPSP algorithm gives an admissible trajectory (satisfying all the constraints). Moreover, when the obstacles are up to three, and they are spaced sparsely; the C-MPSP gives an intuitive trajectory. However, for the condition of densely packed obstacles (five obstacle case); most part of the trajectory merges with the boundary of the path. The reason for this is that the feasible domain is now pretty narrow, and hence the performance of the C-MPSP algorithm started deteriorating.

Computational Efficiency

An important feature of the C-MPSP algorithm is its computational efficiency. This is because of several good features of the MPSP such as converting the optimal control problem to a static optimization problem with recursive computation of sensitivity matrices [see (23)], which makes the algorithm computationally very efficient. This has been observed in the simulations, where, with a $\Delta t = 0.2s$ the run-time of the algorithm is about $0.15s$ on Matlab-2018a in windows-7 professional (64-bit) environment with Intel(R) Core(TM) i3-3220 CPU@3.30GHz and RAM of 4.00 GB. With a dedicated processor and code written in a low-level language (such as embedded C), the computational time is expected to be significantly lower. Thus, one can use the C-MPSP algorithm to efficiently solve complex optimal control problems online without difficulty.

Conclusion

In this paper, a new optimal control law named *Comprehensive Model Predictive Static Programming (C-MPSP)* has been proposed. This optimal control law solves a constrained optimal control problem efficiently by converting it to a quadratic programming problem by successive linearization of the state dynamics along the predicted state trajectory. The Q-linear convergence of the algorithm to a local optimum was rigorously established. The algorithm has been implemented on a differentially driven two-wheel mobile robot. It has been shown using numerical results that it can achieve the terminal objective in the presence of state and control constraints.

Funding No funds, grants, or other supports were received for conducting this study.

Declarations

Conflict of interest There is no Conflict of interest.

References

Allgöwer F, Zheng A (2012) Nonlinear model predictive control. Progress in systems and control theory. Birkhäuser, Basel

Balakrishnan SN, Biega V (1996) Adaptive-critic based neural networks for aircraft optimal control. J Guid Control Dyn 19(4):893–898. https://doi.org/10.2514/3.21715

Ben-Asher JZ (2010) Optimal control theory with aerospace applications. American Institute of Aeronautics and Astronautics, Reston

Betts JT (2001) Practical Methods for Optimal Control Using Nonlinear Programming. Advances in design and control, pp 61–125. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. Chap. 3 and 4

Boggs PT, Tolle JW (2000) Sequential quadratic programming for large-scale nonlinear optimization. J Comput Appl Math 124(1–2):123–137

Bryson JAE, Ho Y-C (1975) Applied optimal control: optimization, estimation and control. Hemisphere Publishing Corporation, New York, pp 128–211 (Chap. 4 and 6)

Chen H, Allgöwer F (1998) A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. Automatica 34(10):1205–1217. https://doi.org/10.1016/S0005-1098(98)00073-9

Dwivedi PN, Bhattacharya A, Padhi R (2011) Suboptimal midcourse guidance of interceptors for high-speed targets with alignment angle constraint. J Guid Control Dyn 34(3):860–877. https://doi.org/10.2514/1.50821

Fahroo F, Ross IM (2002) Direct trajectory optimization by a chebyshev pseudospectral method. J Guid Control Dyn 25(1):160–166. https://doi.org/10.2514/2.4862

Ghazaei Ardakani MM, Olofsson B, Robertsson A, Johansson R (2019) Model predictive control for real-time point-to-point trajectory generation. IEEE Trans Autom Sci Eng 16(2):972–983. https://doi.org/10.1109/TASE.2018.2882764

Gong Q, Fahroo F, Ross IM (2008) Spectral algorithm for pseudospectral methods in optimal control. J Guid Control Dyn 31(3):460–471. https://doi.org/10.2514/1.32908

Hager WW, Pardalos PM (2013) Optimal control: theory, algorithms, and applications, vol 15. Springer, New York

Halbe O, Raja RG, Padhi R (2014) Robust reentry guidance of a reusable launch vehicle using model predictive static programming. J Guid Control Dyn 37(1):134–148. https://doi.org/10.2514/1.61615

Hong H, Maity A, Holzapfel F, Tang S (2019) Model predictive convex programming for constrained vehicle guidance. IEEE Trans Aerosp Electron Syst 55(5):2487–2500. https://doi.org/10.1109/TAES.2018.2890375

Hull DG (2013) Optimal control theory for applications. Springer, New York

Kirk DE (1970) Optimal control theory: an introduction. Prentice Hall, Englewood Cliffs, pp 329–413 (**Chap. 6**)

Kumar P, Anoohya BB, Padhi R (2018) Model predictive static programming for optimal command tracking: a fast model predictive control paradigm. ASME J Dyn Syst Measure Control 141(2):021014–02101412. https://doi.org/10.1115/1.4041356

Larson RE, Casti JL (1982) Principles of dynamic programming: advanced theory and applications. Control and systems theory. M. Dekker, New York

Longuski JM, Guzmán JJ, Prussing JE (2014) Optimal control with aerospace applications. Springer, New York

Luenberger DG (1969) Optimization by vector space methods. Wiley, New York

Maity A, Padhi R, Mallaram S, Rao GM, Manickavasagam M (2016) A robust and high precision optimal explicit guidance scheme for solid motor propelled launch vehicles with thrust and drag uncertainty. Int J Syst Sci 47(13):3078–3097. https://doi.org/10.1080/00207721.2015.1088100

Mayne DQ, Rawlings JB, Rao CV, Scokaert POM (2000) Constrained model predictive control: stability and optimality. Automatica 36(6):789–814. https://doi.org/10.1016/S0005-1098(99)00214-9

Mondal S, Padhi R (2018) Angle-constrained terminal guidance using quasi-spectral model predictive static programming. J Guid Control Dyn 41(3):783–791. https://doi.org/10.2514/1.G002893

Morrison DD, Riley JD, Zancanaro JF (1962) Multiple shooting method for two-point boundary value problems. Commun ACM 5(12):613–614. https://doi.org/10.1145/355580.369128

Naidu DS (2003) Optimal control systems. CRC Press, Florida, pp 101–187 (**Chap. 3 and 4**)

Nocedal J, Wright S (2006) Numerical optimization. Springer, New York

Oza HB, Padhi R (2012) Impact-angle-constrained suboptimal model predictive static programming guidance of air-to-ground missiles. J Guid Control Dyn 35(1):153–164. https://doi.org/10.2514/1.53647

Padhi R, Kothari M (2009) Model predictive static programming: a computationally efficient technique for suboptimal control design. Int J Innov Comput Inf Control 5(2):399–411

Padhi R, Banerjee A, Mathavaraj S, Srianish V (2024) Computational guidance using model predictive static programming for challenging space missions: an introductory tutorial with example scenarios. IEEE Control Syst Mag 44(2):55–80. https://doi.org/10.1109/MCS.2024.3358624

Prakash R, Behera L, Mohan S, Jagannathan S (2022) Dual-loop optimal control of a robot manipulator and its application in warehouse automation. IEEE Trans Autom Sci Eng 19(1):262–279. https://doi.org/10.1109/TASE.2020.3027394

Rawlings JB, Angeli D, Bates CN (2012) Fundamentals of economic model predictive control. In: 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pp 3851–3861

Ross IM (2015) A primer on Pontryagin's principle in optimal control, 2nd edn. Collegiate Publishers, San Francisco

Roux JD, Padhi R, Craig IK, (2014) Optimal control of grinding mill circuit using model predictive static programming: a new

nonlinear mpc paradigm. J Process Control 24(12):29–40. https://doi.org/10.1016/j.jprocont.2014.10.007

Sachan K, Padhi R (2019) Waypoint constrained multi-phase optimal guidance of spacecraft for soft lunar landing. Unmanned Syst 07(02):83–104. https://doi.org/10.1142/S230138501950002X

Sage AP (1968) Optimum systems control. Networks series. Prentice-Hall, Englewood Cliffs

Sakode CM, Padhi R (2014) Computationally efficient suboptimal control design for impulsive systems based on model predictive static programming. IFAC Proc 47(1):41–46. https://doi.org/10.3182/20140313-3-IN-3024.00172. 3rd International Conference on Advances in Control and Optimization of Dynamical Systems (2014))

Wang L (2009) Model predictive control system design and implementation using MATLAB®. Springer, London

Wang Y, Boyd S (2010) Fast model predictive control using online optimization. IEEE Trans Control Syst Technol 18(2):267–278

Wismer DA, Chattergy R (1978) Introduction to nonlinear optimization: a problem solving approach, vol 1. North Holland, New York

Yan Z, Kreidieh AR, Vinitsky E, Bayen AM, Wu C (2023) Unified automatic control of vehicular systems with reinforcement learning. IEEE Trans Autom Sci Eng 20(2):789–804. https://doi.org/10.1109/TASE.2022.3168621

Zheng Y, Li Q, Li S (2022) Stability guaranteed model predictive control with adaptive lyapunov constraint. IEEE Trans Autom Sci Eng. https://doi.org/10.1109/TASE.2022.3222182