BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 73(2), 2025, Article number: e153427
DOI: 10.24425/bpasts.2025.153427

Nonholonomic motion planning with special restrictions on the end and via points of the control function

Joanna RATAJCZAK*

Department of Cybernetics and Robotics, Wrocław University of Science and Technology, Wrocław, Poland

Abstract. This paper introduces a new modification to the motion planning algorithm of nonholonomic robotic systems using the endogenous configuration space approach which allows imposing restrictions on control functions. The end and via points define the values which the control function should take in a predefined time, either at the beginning, the end or during the motion time horizon. Such a modification can be used to set the values of the control function, which usually are of velocity-like type, to be physically realizable. The constraints are introduced to the algorithm through the extension of the Jacobian. The efficiency of the presented method is shown with the computer simulation results for a nonholonomic space manipulator. A modified Jacobian motion planning algorithm is used for planning consisting of a sequence of two subtasks.

Keywords: motion planning; nonholonomic systems; constraints; continuous control function; space manipulators.

1. INTRODUCTION

The pure motion planning problem of a nonholonomic system is to determine the control function that acts on the system in such a way that it performs the desired motion. In many cases, especially practical ones, this approach is insufficient and additional tasks or constraints need to be added to successfully solve a given problem.

In this article, we present an algorithm that is able to solve the motion planning problem, namely the resulting control function leads the robotic system to the desired point, and in addition, the control function takes the desired values at specific time instants.

The motivation of this paper arises from practical requirements.

Usually, the local motion planning for the robot moving in the presence of obstacles is more effective than the solution returned by global planners. For local motion planners, the global motion may be composed as a sequence of movements, i.e., the final configuration from the previous planning becomes the initial configuration for the current one. In that case, the problem of control discontinuity on the transition segment can easily arise. This is very troublesome and not desirable in terms of practical applications. The proposed modified algorithm ensures that class $\mathbb{C}^0$ or $\mathbb{C}^1$ controls are obtained even when planning a sequence of movements.

On the other hand, very often, the initial values of the state vector, the velocities and also the accelerations of the nonholonomic robotic system are determined by the simulation scenario. Usually, the system composed of the robot together with the controller is defined throughout the ordinary differential equations (ODEs), which should be solved to obtain the solution of the planning problem. If one has the kinematics model of the nonholonomic robot that is defined as the first-order ODEs, then as initial values we can set only the state variables (e.g., that define the position/orientation of the robot), and it is impossible to set the initial velocities and accelerations. If the model is expanded by the dynamics, so the system is expressed as second-order ODEs, then the velocities at the beginning could be defined by the initial conditions. Nevertheless, it is still impossible to set the initial accelerations. If we assume that the controls in the kinematics model are velocity-like, and in the dynamics have the sense of acceleration then our idea to define in advance the end points of the control function will allow us to set the initial values of velocity or acceleration to meet the physical requirements.

It is worth mentioning that the proposed modification of the Jacobian motion planning algorithm is more general, and provides the possibility of setting prescribed values of control function at arbitrary points in time. So, in general, this algorithm allows us to set the control values at the beginning, the end, and any other specific point during the motion.

To sum up, the main contribution of this paper is a modification of the Jacobian motion planning algorithm based on the endogenous configuration space approach that introduces the restrictions into the resultant control function at specified time instants.

The algorithm is derived within the endogenous configuration space approach [1]. The modification is defined by the extension of the Jacobian [2] constituting a kind of the egalitarian two-task approach [3]. The endogenous configuration space was previously used to successfully solve both unconstrained [4]

*e-mail: joanna.ratajczak@pwr.edu.pl

Manuscript submitted 2024-07-23, revised 2024-11-20, initially accepted for publication 2025-01-05, published in March 2025.

and constrained [5, 6] motion planning problems, and could be even enrolled in trajectory reproduction task [7].

The introduced algorithm is originally dedicated to planning the motion of a nonholonomic robotic system where there is a need to constrain the control functions. To illustrate the efficiency of our proposition, the presented approach will be used to plan the motion of a free-floating space manipulator. Such a problem, namely the motion planning of a space manipulator was already studied in the literature. One can find some different approaches to unconstrained motion planning problem in [8–11], as well as to an approach with constraints [12–14]. Usually, the motion planning phase is followed by the control stage, some more information about the control methods for space manipulators can be found in e.g., [15–17]. Space manipulators controlled not only by joints but also by reaction wheels or thrusters have also become increasingly popular in recent research [18–20].

The remaining part of the paper is as follows. The problem statement and Jacobian motion planning algorithm are characterized in Section 2. In Section 3, the idea of the algorithm using the extended Jacobian is introduced. Section 4 describes a modification of the algorithm that introduces constraints on the control functions. Simulation results are included in Section 5. The paper is summarized in Section 6.

2. PROBLEM STATEMENT

We are dealing with a nonholonomic robotic system described by the control-affine system

$$\begin{cases} \dot{q} = f(q) + G(q)u = f(q) + \sum_{i=1}^{m} g_i(q)u_i, \\ y = k(q), \end{cases} \tag{1}$$

where $q \in \mathbb{R}^n$ is the vector of generalized coordinates, $u \in \mathbb{R}^m$ denotes the control variable, $f(q)$ is the drift vector, $G(q)$ is the control matrix, $y \in \mathbb{R}^r$ describes the task space vector and $k(q)$ is the output function. Let $T > 0$ denote a control time horizon. The control functions in system (1) will be chosen as Lebesgue square integrable functions $L^2_m[0,T]$ of time on the interval $[0,T]$. The space of such selected control functions will be called an endogenous configuration space $\mathscr{U} \ni u(\cdot)$ [21]. The trajectory of system (1), resulting from the initial state $q(0)$ and control $u(\cdot)$, is denoted by $q(t) = \varphi_{q_0,t}(u(\cdot))$, where $\varphi_{q_0,t}(u(\cdot))$ is a flow of the system (1), initialized at q_0 and driven by $u(\cdot)$.

The presented problem of motion planning with additional constraints on the control function relies on a combination of proper motion planning along with the ability to impose the prescribed values of the control function (and its derivatives) in a particular time moment, so in fact there are two subtasks.

The solution to such a defined problem, with two equal subtasks, will be a control function which drives the system (1) from an initial state q_0 to a desired value of output $y(T) = y_d$, at the end of time interval T. Moreover, the resultant control function should preserve the additional constraints:

$$u(t_k) = w_k, \quad \text{for } k = 1, 2, \ldots, c_w,$$

for the control function, and

$$\frac{\mathrm{d}u(t_k)}{\mathrm{d}t} = d_k, \quad \text{for } k = 1, 2, \ldots, c_d,$$

for its derivative. t_k denotes the time instance, w_k and d_k are constant values. So the whole main motion planning problem may be written symbolically as

$$\begin{cases} q_0 = q(0) \xrightarrow{u^*(\cdot)} y(T) = y_d, \\ u^*(t_k) = w_k = \text{const. } w_k \in \mathbb{R}^m, \\ \dfrac{\mathrm{d}u^*(t_k)}{\mathrm{d}t} = d_k = \text{const. } d_k \in \mathbb{R}^m, \end{cases} \tag{2}$$

where $u^*(\cdot)$ is a resultant control function. When the time instance t_k in (2) is $t_k = 0$ it refers to as initial point, when $t_k = T$ it is an end point, and any other value $0 < t_k < T$ is called the via point.

3. MOTION PLANNING ALGORITHM WITH EXTENSION FUNCTIONS

As we already mentioned, the above problem is composed of two equally significant subtasks, so the motion planning algorithm will be derived utilizing the extended Jacobian.

3.1. Preliminaries

So, let us define the end point map as

$$\mathscr{K}_{q_0,T}(u(\cdot)) = k(q(T)) = y(T) \tag{3}$$

which determines the output of system (1) at the time T. By differentiation (using Gâteaux derivative) the end point map (3) with respect to $u(\cdot)$, we arrive with the Jacobian of system (1) of the form

$$J_{q_0,T}(u(\cdot)) = \mathrm{D}\mathscr{K}_{q_0,T}(u(\cdot)) = \left.\frac{\mathrm{d}}{\mathrm{d}\vartheta}\right|_{\alpha=0} \mathscr{K}_{q_0,T}(u(\cdot) + \alpha v(\cdot)), \tag{4}$$

where $v(t)$ is a variation of $u(t)$ and $\vartheta \in \mathbb{R}$ is an independent variable orthogonal to t. The careful computation of the derivative equation (4), led us to a linear variational system associated with (1) of the form [21]

$$\begin{cases} \dot{\xi}(t) = A(t)\xi(t) + B(t)v(t), \\ \eta(t) = C(t)\xi(t). \end{cases} \tag{5}$$

The linear system (5) is actually a linear approximation to system (1) along the control-state pair $(u(t), q(t))$. It is well-known that the matrices are defined as $A(t) = \dfrac{\partial(f(q(t)) + G(q(t))u(t))}{\partial q}$, $B(t) = \dfrac{\partial(f(q(t)) + G(q(t))u(t))}{\partial u} = G(q(t))$, $C(t) = \dfrac{\partial k(q(t))}{\partial q}$.

The explicit form of the Jacobian (4), may be obtained as a solution of linear equation (5), following [22],

$$J_{q_0,T}(u(\cdot))v(\cdot) = \int_0^T \Phi(T,t)B(t)v(t)\,\mathrm{d}t, \tag{6}$$

where $\Phi(t,s)$ is a fundamental matrix of (5) and solves the partial differential equation $\dfrac{\partial\Phi(t,s)}{\partial t} = A(t)\Phi(t,s)$ with initial conditions $\Phi(s,s) = I_n$.

3.2. Algorithm derivation

To obtain the solution of the motion planning problem, the sought control function, we chose in the endogenous configuration space a smooth curve $u_\vartheta(\cdot)$ parameterized by $\vartheta \in \mathbb{R}$ passing an initial configuration (predefined initial controls) $u_{\vartheta=0}(\cdot)$. Along this curve, we propose the motion planning error as

$$e(\vartheta) = \mathscr{K}_{q_0,T}(u_\vartheta(\cdot)) - y_d\,, \tag{7}$$

which should decrease exponentially

$$\frac{\mathrm{d}e(\vartheta)}{\mathrm{d}\vartheta} = -\gamma e(\vartheta), \tag{8}$$

along ϑ with a decay rate $\gamma \in \mathbb{R}$. Combining (7) and (8) with (4) we arrive with Ważewski–Davidenko equation

$$J_{q_0,T}(u_\vartheta(\cdot))v_\vartheta(\cdot) = -\gamma e(\vartheta). \tag{9}$$

To solve (9) we may use any right Jacobian inverse. In addition, we may observe, that as long as the dimension of endogenous configuration space is greater than the dimension of the task space we may extend the Jacobian with a number of extension functions of the form, e.g., equality constraints,

$$f_i(u_\vartheta(\cdot)) = 0, \tag{10}$$

formulating the extended Jacobian

$$\mathbf{J}_{q_0,T}(u_\vartheta(\cdot))v_\vartheta(\cdot) = \begin{bmatrix} J_{q_0,T}(u_\vartheta(\cdot)) \\ \dfrac{\mathrm{d}f_1(u_\vartheta)}{\mathrm{d}\vartheta} \\ \vdots \\ \dfrac{\mathrm{d}f_i(u_\vartheta)}{\mathrm{d}\vartheta} \\ \vdots \end{bmatrix} v_\vartheta(\cdot). \tag{11}$$

It is worth mentioning that functions (10) may be treated as the augmenting kinematics functions. Additionally, to be able to solve (9) with the extended Jacobian, we need also to extend the error to the form

$$\mathbf{e}(\vartheta) = \begin{bmatrix} e(\vartheta) \\ 0 \\ \vdots \end{bmatrix}.$$

Because the dimension of the endogenous configuration space is greater than the dimension of the task space, to enrol the extended Jacobian equation (11) to solve (9) we need to introduce the pseudoinverse

$$\left(\mathbf{J}^{\#}_{q_0,T}(u_\vartheta(\cdot))\eta\right)(t) = \mathbf{J}^{*}_{q_0,T}(u_\vartheta(\cdot))\mathscr{G}_{q_0,T}(u_\vartheta(\cdot)), \tag{12}$$

where $\mathbf{J}^{*}_{q_0,T}(u_\vartheta(\cdot))$ is an adjoint Jacobian [21] and

$$\mathscr{G}_{q_0,T}(u_\vartheta(\cdot)) = \mathbf{J}_{q_0,T}(u_\vartheta(\cdot))\mathbf{J}^{*}_{q_0,T}(u_\vartheta(\cdot)) \tag{13}$$

is a Gram matrix. Using the inverse equation (12) in (9) we obtain as a solution a dynamical system

$$v_\vartheta(\cdot) = \frac{\mathrm{d}u_\vartheta(\cdot)}{\mathrm{d}\vartheta} = -\gamma\left(\mathbf{J}^{\#}_{q_0,T}(u_\vartheta(\cdot))\mathbf{e}(\vartheta)\right)(\cdot), \tag{14}$$

with an arbitrarily chosen initial condition $u_{\vartheta=0}(\cdot)$. The solution, namely the control function $u^*(\cdot)$ that drives the system (1) from initial configuration q_0 do the desired output value y_d in time $t \in [0,T]$, is the limit $u^*(\cdot) = \lim_{\vartheta\to\infty} u_\vartheta(\cdot)$ of the resultant trajectory of (14). Moreover, the obtained control function $u^*(\cdot)$ keeps all the extension functions $f_i(u_\vartheta(\cdot))$ values close to zero.

3.3. Finite-dimensional approach

The endogenous configuration space $\mathscr{U}$ is an infinite-dimensional function space. For practical reasons, to simplify the implementation aspects we shall employ a parametric representation of the control function. In this case, we assume that the control function is a finite-dimensional function defined by the truncated orthogonal series

$$u_i(\lambda,t) = \sum_{j=1}^{p_i} \lambda_{ij}\phi_{ij}(t) = P_{b_i}(t)\lambda_i, \qquad P_{b_i} = [\phi_{i1},\phi_{i2},\ldots,\phi_{ip_i}],$$

where p_i denotes the basis function number of particular control function u_i whose λ_i is a control parameter vector. This means that the whole control function $u(\lambda,t)$ of system (1) can be rewritten as

$$u(\lambda,t) = P(t)\lambda, \tag{15}$$

where $P(t) = \mathrm{diag}\{P_{b_1}(t), P_{b_2}(t), \ldots, P_{b_m}(t)\}$ is a diagonal matrix built of m vectors $P_{b_i}(t)$ and $\lambda \in \mathbb{R}^s$ denotes a collectible control parameters vector. So, the total number of control parameters is equal to $s = \sum_{i=1}^{m} p_i$. Such defined controls belong to a finite-dimensional endogenous configuration space $\widetilde{\mathscr{U}} = \mathbb{R}^s$. Following the line of reasoning, we introduce a finite-dimensional end point map as

$$\widetilde{\mathscr{K}}_{q_0,T}(\lambda) = \mathscr{K}_{q_0,T}(u(\lambda,\cdot)) = k(\varphi_{q_0,T}(u(\lambda,\cdot))). \tag{16}$$

Obviously, the parametric representation of the control functions, (15), induces also the parametric version of control function variations

$$v_\vartheta(\lambda,\cdot) = \frac{\mathrm{d}u_\vartheta(\lambda,\cdot)}{\mathrm{d}\vartheta} = \frac{\mathrm{d}P(t)\lambda_\vartheta}{\mathrm{d}\vartheta} = P(t)\frac{\mathrm{d}\lambda_\vartheta}{\mathrm{d}\vartheta} = P(t)\mu.$$

As it was in the infinite-dimensional case, the differentiation of the end point map (16) yields a finite-dimensional variational system

$$\begin{cases} \dot{\xi}(t) = A_\lambda(t)\xi(t) + B_\lambda(t)v(t), \\ \eta(t) = C_\lambda(t)\xi(t), \end{cases} \tag{17}$$

whose matrices are defined analogously to matrices from (5). The solution of equation (17) formulates the finite-dimensional Jacobian

$$\widetilde{J}_{q_0,T}(\lambda) = C_\lambda(T)\xi(T) = C_\lambda(T)\int_0^T \Phi_\lambda(t,s)B_\lambda(s)P(s)\,\mathrm{d}s. \tag{18}$$

Please observe, that in the finite-dimensional case, the Jacobian (18) is a linear operator acting between Euclidean spaces, so it is a matrix.

Having the parametric version of the end point map (16) and the Jacobian (18) we can define the motion planning problem which consists in finding a control function $u^* = u(\lambda^*, \cdot)$ satisfying

$$y(T) = \widetilde{\mathscr{K}}_{q_0,T}(\lambda^*) = y_d\,.$$

To solve this problem we proceed similarly to the infinite-dimensional case. We choose in $\widetilde{\mathscr{U}}$ a smooth curve $\lambda(\vartheta) \in \mathbb{R}^s$, parameterized by $\vartheta \in \mathbb{R}$, passing through certain initial control $\lambda_{\vartheta=0}$. Next, we define the motion planning error

$$\widetilde{e}(\vartheta) = \widetilde{\mathscr{K}}_{q_0,T}(\lambda(\vartheta)) - y_d\,,$$

and require it to decrease exponentially along this curve

$$\frac{\mathrm{d}\widetilde{e}(\vartheta)}{\mathrm{d}\vartheta} = -\gamma\widetilde{e}(\vartheta), \qquad \gamma > 0,$$

which leads us again to Ważewski–Davidenko equation

$$\widetilde{J}_{q_0,T}(\lambda_\vartheta)\mu_\vartheta = -\gamma\widetilde{e}(\vartheta).$$

Likewise the infinite-dimensional case, we introduce a set of extension functions $f_i(\lambda_\vartheta) = 0$, $i = 1,2,\ldots,k$. However, this time, the total number of them is limited by the difference $k = s - n$. Nevertheless, we define a priori the number s of control parameters in (15), so we can freely expand the parametrization as needed. This allows us to collect the finite-dimensional extended Jacobian (as a matrix)

$$\widetilde{\mathbf{J}}_{q_0,T}(\lambda_\vartheta)\mu_\vartheta = \begin{bmatrix} \widetilde{J}_{q_0,T}(\lambda_\vartheta) \\ \dfrac{\mathrm{d}f_1(\lambda_\vartheta)}{\mathrm{d}\vartheta} \\ \vdots \\ \dfrac{\mathrm{d}f_k(\lambda_\vartheta)}{\mathrm{d}\vartheta} \end{bmatrix}\mu_\vartheta \tag{19}$$

and the corresponding extended error $\widetilde{\mathbf{e}}(\vartheta) = (\widetilde{e}(\vartheta), 0, \ldots, 0)$.

Finally, enrolling the above derivation together with the Jacobian matrix pseudoinverse, we may constitute the finite-dimensional motion planning algorithm as a differential equation

$$\frac{\mathrm{d}\lambda_\vartheta}{\mathrm{d}\vartheta} = -\gamma\widetilde{\mathbf{J}}^{\#}_{q_0,T}(\lambda_\vartheta)\widetilde{\mathbf{e}}(\vartheta). \tag{20}$$

Then the sought control function which solves the motion planning problem and fulfills the constraints is equal to $u^*(t) = P(t)\lambda^*_\vartheta$, where λ^*_ϑ is obtained as $\lim_{\vartheta\to\infty}\lambda_\vartheta$ of a resultant trajectory of equation (20).

In fact we cannot fully control the evolution of the solution $u(t)$ along the independent variable $\vartheta \in \mathbb{R}$ of the Jacobian algorithm (20). For this reason, the obtained control is often impractical, e.g. its amplitude overshoots technical limits or the initial control $u(t=0)$ takes a nonzero value which may be hard to obtain in practical realizations. To overcome these disadvantages we will carefully define the extension functions.

4. MAINTAINING RESTRICTIONS

The approach presented in this paper tries to alleviate the above disadvantage yet along with solving the motion planning problem. The improved algorithm allows us to prescribe the beginning point, the end point and the via points in the resultant control function. As it was in the previous chapter, we again introduce the approach in the infinite-dimensional case and then propose the finite-dimensional implementation.

Let the control function constraints $u_\vartheta(t_k) = w_k = \text{const.}$, $t_k \in [0,T]$, $w_k \in \mathbb{R}^m$ and $k = 1,2,\ldots,c_w$, where c_w denotes the number of predefined control points. If $t_k = 0$ then we specify the beginning point of the control function, for $t_k = T$ we set the end point, for all other cases we have via points. Due to the construction of Jacobian algorithms, we cannot explicitly force the control function to pass through the points that we have specified. However, we can introduce the algorithm modification preserving

$$\frac{\mathrm{d}u_\vartheta(t_k)}{\mathrm{d}\vartheta} = v_\vartheta(t_k) = 0 \tag{21}$$

which together with a properly defined initial control function

$$u_{\vartheta=0}(t_k) = w_k, \quad \forall k \tag{22}$$

provides a resultant control function $u^*(t_k) = w_k$ with via point.

Additionally, we are interested not only in the control value itself but also in the derivative (slope, velocity) of the control function in point $u_\vartheta(t_k)$. So we want to fulfill also the condition $\dfrac{\mathrm{d}u_\vartheta(t_k)}{\mathrm{d}t} = d_k = \text{const.}$, $t_k \in [0,T]$, $d_k \in \mathbb{R}^m$ and $k = 1,2,\ldots,c_d$, where this time c_d denotes the number of predefined control first derivatives. Again, the modified algorithm that preserves

$$\frac{\mathrm{d}}{\mathrm{d}\vartheta}\frac{\mathrm{d}u_\vartheta(t_k)}{\mathrm{d}t} = \frac{\mathrm{d}v_\vartheta(t_k)}{\mathrm{d}t} = 0 \tag{23}$$

along with carefully selected initial condition

$$\frac{\mathrm{d}u_{\vartheta=0}(t_k)}{\mathrm{d}t} = d_k, \quad \forall k \tag{24}$$

will provide a resultant control function whose derivative $\mathrm{d}u^*(t_k)/\mathrm{d}t = d_k$. The functions (22) and (24) formulate algorithm constraints, namely the extension functions (10). And consequently, the extended Jacobian (11) may be constructed using their derivatives with respect to ϑ, equations (21) and (23).

4.1. Finite-dimensional approach

Introducing the parameterized control function according to (15), we may derive the explicit formulas for the extension functions. Next, we introduce the above-mentioned modification to algorithm (20) using the extended Jacobian technique. The control restrictions on its values and first derivatives take the finite-dimensional form

$$u_\vartheta(\lambda_\vartheta, t_k) = P(t_k)\lambda_\vartheta = w_k\,, \qquad \frac{\mathrm{d}u_\vartheta(\lambda_\vartheta, t_k)}{\mathrm{d}t} = \frac{\mathrm{d}P(t_k)}{\mathrm{d}t}\lambda_\vartheta = d_k\,.$$

The differentiation of them with respect to ϑ yields in the extension functions

$$P(t_k)\mu_\vartheta = 0, \qquad \frac{\mathrm{d}P(t_k)}{\mathrm{d}t}\mu_\vartheta = 0.$$

Now we are ready to formulate the extended Jacobian $\widetilde{\mathbf{J}}_{q_0,T}(\lambda_\vartheta)$ as

$$\widetilde{\mathbf{J}}_{q_0,T}(\lambda_\vartheta)\mu_\vartheta = \begin{bmatrix} \widetilde{J}_{q_0,T}(\lambda_\vartheta) \\ P(t_1) \\ \vdots \\ P(t_{c_w}) \\ \dfrac{\mathrm{d}P(t_1)}{\mathrm{d}t} \\ \vdots \\ \dfrac{\mathrm{d}P(t_{c_d})}{\mathrm{d}t} \end{bmatrix} \mu_\vartheta = \begin{bmatrix} \widetilde{J}_{q_0,T}(\lambda_\vartheta) \\ \Psi \end{bmatrix} \mu_\vartheta\,, \tag{25}$$

where Ψ collects all the extension functions, together with the extended error $\widetilde{\mathbf{e}}(\vartheta) = (\widetilde{e}(\vartheta), 0, \ldots, 0) \in \mathbb{R}^{r+mc_w+mc_d}$. This finally led us to the motion planning algorithm with the constraints on control function defined as

$$\frac{\mathrm{d}\lambda_\vartheta}{\mathrm{d}\vartheta} = -\gamma \widetilde{\mathbf{J}}^{\#}_{q_0,T}(\lambda_\vartheta)\widetilde{\mathbf{e}}(\vartheta) \tag{26}$$

where $\widetilde{\mathbf{J}}^{\#}_{q_0,T}(\lambda_\vartheta)$ is the right pseudoinverse of the Jacobian matrix (25). The Jacobian motion planning algorithm (26) returns as a limit $\lim_{\vartheta\to\infty}\lambda_\vartheta$ the parameterized control function which solves the motion planning problem and simultaneously takes desired values and derivatives at given points. The initial conditions for $\lambda_{\vartheta=0}$ should fulfill the following

$$\Psi\lambda_{\vartheta=0} = \begin{bmatrix} w_1 & \cdots & w_{c_w} & d_1 & \cdots & d_{c_d} \end{bmatrix}^T. \tag{27}$$

The above derivation of the algorithm can be easily expanded to even higher-order control derivatives to influence the smoothness of the control function.

5. SIMULATIONS RESULTS

To illustrate the performance of the motion planning algorithm we have chosen as a test bed the 2 DoF planar space manipulator mounted on the free-floating base, depicted in Fig. 1. The space robot has been inspired by the space manipulator designed in the Space Research Center of the Polish Academy of Sciences [23].

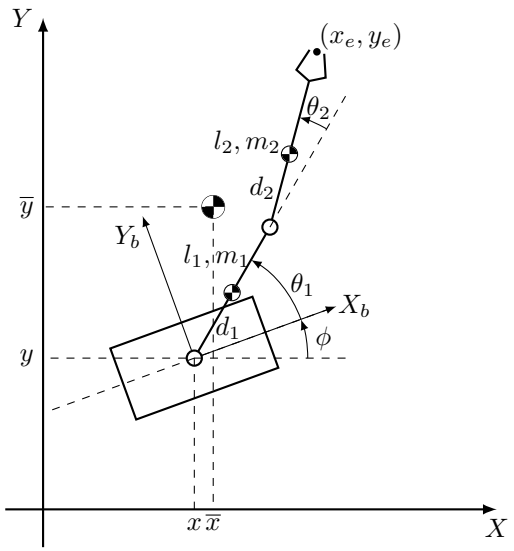

Fig. 1. 2DoF space manipulator

Note that the proposed method can be applied to the motion planning of any robotic system whose equations of motion can be represented by a control-affine system. It is possible to solve the motion planning problem for a spatial (3D) space manipulator, and the proposed model of a planar space manipulator has been chosen for simplicity.

To derive the dynamics of the space manipulator described by the generalized coordinates $\bar{q} = (\bar{x}, \bar{y}, \phi, \theta_1, \theta_2)^T$, we shall start with the Lagrangian

$$L(\bar{q}, \dot{\bar{q}}) = \frac{1}{2}\mathsf{A}\left(\dot{\bar{x}}^2 + \dot{\bar{y}}^2\right) + \frac{1}{2}\mathsf{I}\dot{\phi}^2 + \frac{1}{2}\mathsf{B}\left(\dot{\phi}+\dot{\theta}_1\right)^2 + \frac{1}{2}\mathsf{C}\left(\dot{\phi}+\dot{\theta}_{12}\right)^2 + \mathsf{D}\cos\theta_2\left(\dot{\phi}+\dot{\theta}_1\right)\left(\dot{\phi}+\dot{\theta}_{12}\right),$$

where $\bar{x}$, $\bar{y}$ are the barycentric coordinates of the manipulator [24], ϕ is the orientation of base, θ_1, θ_2 are manipulator joint angles, I is the moment of the inertia of the base and constant parameters A, B, C, D are as follows [25]

$$\mathsf{A} = M + m_{12}\,, \qquad \mathsf{B} = \frac{m_1 m_2 (l_1 - d_1)^2 + M(m_1 d_1^2 + m_2 l_1^2)}{M + m_{12}}\,,$$

$$\mathsf{C} = \frac{(M+m_1)m_2 d_2^2}{M+m_{12}}\,, \qquad \mathsf{D} = \frac{m_1 m_2 (l_1 - d_1) d_2 + M m_2 l_1 d_2}{M + m_{12}}\,.$$

Because the manipulator is moving in space we neglect the action of gravity.

From the Euler–Lagrange equations of motion of the space manipulator, it follows that the independence of the Lagrangian

of $\bar{x}$, $\bar{y}$ yields the conservation of the linear momenta

$$\mathsf{A}\dot{\bar{x}} = \text{const}, \qquad \mathsf{A}\dot{\bar{y}} = \text{const}.$$

Thus, the center of mass of the manipulator moves uniformly and rectilinearly in space, and the $\bar{x}$ and $\bar{y}$ changes are independent of the other coordinates.

The conservation of the angular momentum leads to the affine Pfaffian constraint

$$\mathscr{A}(q)\dot{q} = \mathsf{F}(\theta_2)\dot{\phi} + \mathsf{G}(\theta_2)\dot{\theta}_1 + \mathsf{H}(\theta_2)\dot{\theta}_2 = \mathsf{p}, \tag{28}$$

p denotes the constant conserved angular momentum, $\mathsf{F}(\theta_2) = \mathsf{I} + \mathsf{B} + \mathsf{C} + 2\mathsf{D}\cos\theta_2$, $\mathsf{G}(\theta_2) = \mathsf{B} + \mathsf{C} + 2\mathsf{D}\cos\theta_2$, $\mathsf{H}(\theta_2) = \mathsf{C} + \mathsf{D}\cos\theta_2$. Based on the Pfaffian constraint (28) the dynamics of the space manipulator in form of equation (1) can be obtained. The drift vector field is computed as $f(q) = \mathscr{A}^{\#}(a)\mathsf{p}$, where $\mathscr{A}^{\#}(q)$ is the right inverse of $\mathscr{A}(q)$, and the control matrix fulfills $\mathscr{A}(q)G(q) = 0$. Finally, the equation of motion of the space manipulator gets the following representation

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} = \begin{pmatrix} \dfrac{\mathsf{p}}{\mathsf{F}(\theta_2)} \\ 0 \\ 0 \end{pmatrix} + \begin{bmatrix} -\dfrac{\mathsf{G}(\theta_2)}{\mathsf{F}(\theta_2)} & -\dfrac{\mathsf{H}(\theta_2)}{\mathsf{F}(\theta_2)} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{29}$$

Since, as we mentioned earlier, the $\bar{x}$ and $\bar{y}$ changes independently of the other coordinates, we will focus on the model described by (29) with the coordinates $q = (\phi, \theta_1, \theta_2)^T$.

Now, for comparison purposes, we shall solve the motion planning problem for the space manipulator described in form of equation (29). The task is formulated as follows. Starting from the initial position $q_0 = q(t_0 = 0) = (\frac{\pi}{8}, -\frac{\pi}{6}, \frac{\pi}{6})$, move to the desired mid-position $y_1 = q(t_1 = T = 20) = \left(0, 0, \frac{\pi}{8}\right)$ and then to the final point $y_f = q(t_2 = 2T = 40) = \left(\frac{\pi}{8}, -\frac{\pi}{8}, \frac{\pi}{6}\right)$. It may be observed that in this case the output function $y = k(q) = q$, and the task itself can be viewed as a gluing together of two movements with a time horizon of $T = 20$ for each movement. To show the efficiency of the proposed approach, we shall solve this task in three scenarios. First, we apply the Jacobian motion planning algorithm without any restrictions on control functions. In the second scenario, we use the Jacobian motion planning algorithm to solve rest-to-rest motion which means that the control functions at the beginning and the end of the motion are equal to zero $u(0) = u(2T) = 0$. What is more, the control function should be continuous throughout the motion, so the control should be a function of class $\mathbb{C}^0$. Last but not least, the task should be solved with the assumption of the rest-to-rest motion, with $\left.\dfrac{\mathrm{d}u(t)}{\mathrm{d}t}\right|_{t=0} = 0.01$, and with the emphasis on the continuity of the first derivative, especially at the connection point $\left.\dfrac{\mathrm{d}u(t)}{\mathrm{d}t}\right|_{t=t_1}$. It is worth noting that satisfying the continuity condition for the first derivative of the control function gives a certain smoothness, namely the class $\mathbb{C}^1$.

In all cases the right pseudoinverse Jacobian is used, the algorithm error decay rate $\gamma = 0.02$, and the value of the conserved angular momentum $\mathsf{p} = 0$.

Because we have set the conditions that $u(0) = 0$ and $u(2T) = 0$, which means that we want the joint velocities to be equal to zero at time $t = 0$ and $t = 2T$, it is necessary to select an appropriate collection of orthogonal functions for representing the control function. In the case when only $u(0) = u(T)$, and therefore without any other restrictions on via points of control functions, the controls may be expressed in terms of a widely used Fourier series. In our case, we must ensure that also the constraints for via points are met. So, to obtain the continuity of the control function and its derivative, it is required to use a suitable collection of orthogonal functions. For this purpose, we have chosen the Legendre polynomials. The values of the vector λ_0 for the initial control functions $u_0 = 0$ are obtained as a solution of the equation (27) using Moore–Penrose's inverse.

The solution of the task for all three scenarios together with the resultant control functions are presented in Figs. 2–8. Also, Figs. 9–11 are included to better demonstrate the continuity of the derivative of the control functions. As can be seen in the figures, the motion planning problem is correctly solved for all three cases.

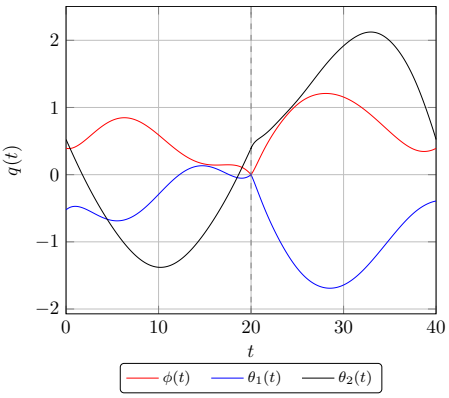

Fig. 2. The trajectories of $q(t)$ – the first scenario

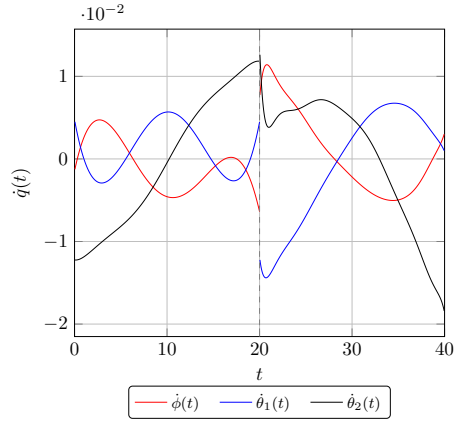

Fig. 3. The trajectories of $\dot{q}(t)$ – the first scenario

In Fig. 2, Fig. 4 and Fig. 9 it can be seen that although the task has been successfully solved, the trajectories $q(t)$ are continuous but at the sticking point $q(t = t_1 = T)$ they are not differentiable, as can also be seen in Fig. 3. Since no constraints have been

imposed on the control functions, the values of the controls at the beginning and end of the motion are not zero and, therefore, the velocities of the joint variables are not either. Moreover, neither the controls nor their derivatives are continuous, Fig. 4 and Fig. 9. As a rule, continuity is not maintained when this property does not need to be fulfilled. The simulation results show that the assumptions of the second scenario are met. The trajectories $q(t)$ are continuous and differentiable, Fig. 5, the resulting control functions are continuous, Fig. 6, but not differentiable, Fig. 10. So only the control functions of class $\mathbb{C}^0$ are reachable. In the third scenario, the trajectories $q(t)$ are continuous and

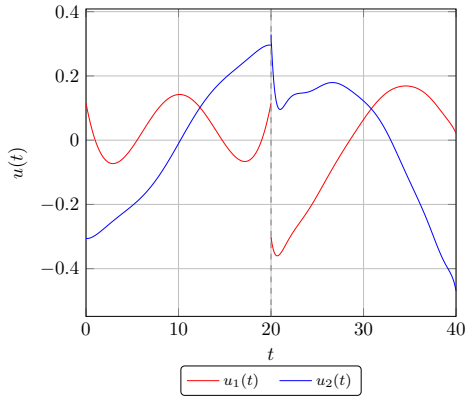

Fig. 4. The resultant control functions $u(t)$ – the first scenario

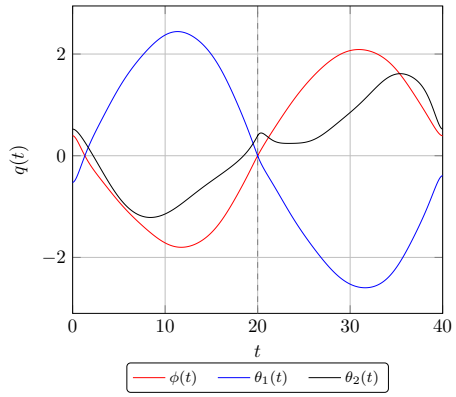

Fig. 5. The trajectories of $q(t)$ – the second scenario

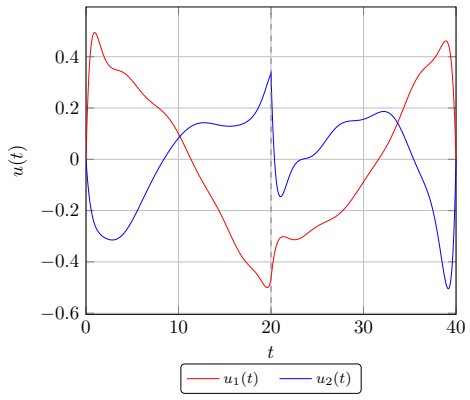

Fig. 6. The resultant control functions $u(t)$ – the second scenario

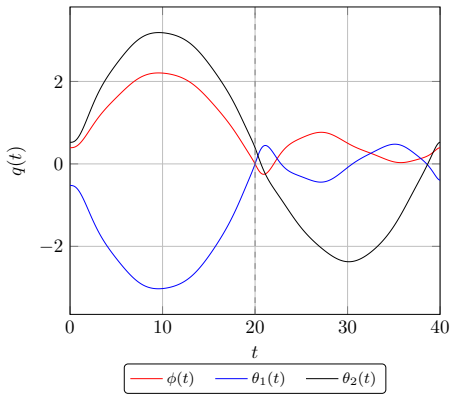

Fig. 7. The trajectories of $q(t)$ – the third scenario

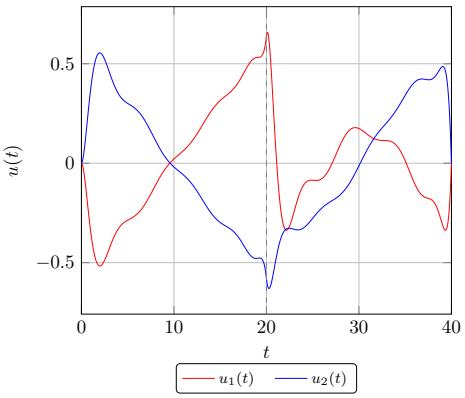

Fig. 8. The resultant control functions $u(t)$ – the third scenario

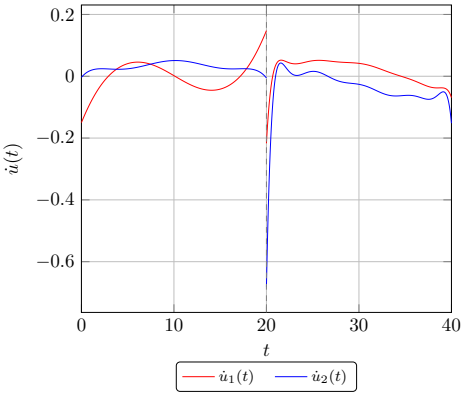

Fig. 9. The resultant derivative of the control functions $\dot{u}(t)$ – the first scenario

differentiable, and the continuously differentiable up to order 1 controls are guaranteed (the class $\mathbb{C}^1$), Fig. 8 and Fig. 11.

It is important to highlight that the last scenario presents a viable solution to a real problem. With the proposed method, we can define not only the initial, intermediate and final states but also the initial and final velocities and accelerations, all while using first-order ordinary differential equations (ODEs) of motion. Additionally, this approach ensures the continuity and differentiability of the resulting control functions.

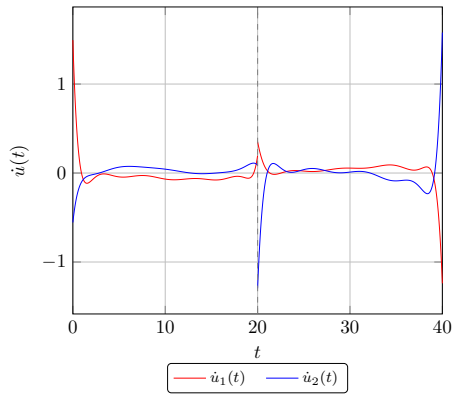

Fig. 10. The resultant derivative of the control functions $\dot{u}(t)$ – the second scenario

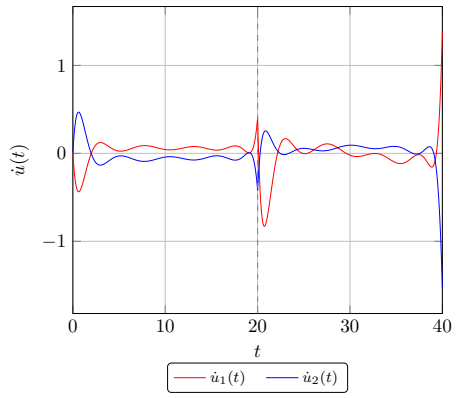

Fig. 11. The resultant derivative of the control functions $\dot{u}(t)$ – the third scenario

6. CONCLUSIONS

In this paper, a method for preserving specific restrictions on control functions for nonholonomic motion planning is presented. A proposed modification of the endogenous configuration space approach successfully solves the motion planning problem together with preserving the requirements for the prescribed end point (via point) in the control function. The efficiency of this approach is illustrated with simulation results for the space manipulator. This method can be used when, for some reason, the control function (or its derivative) must take on specific values at specific moments in time, for example, a motion planning task is to be solved in an environment full of obstacles or is divided into several subtasks. It is well-known that ensuring continuity of controls plays an important role in practical applications, and this method provides that.

On the other hand, when using the original (unconstrained) motion planning Jacobian algorithm based on the endogenous configuration space approach, the initial conditions for the control functions cannot be determined in advance. It makes it hard to compare motion planning solutions for nonholonomic robotic systems expressed in different forms (e.g., the original one and its normal form) or to compare the performance of various algorithms. The presented method is a helpful tool for dealing with such a drawback and makes the comparison more reliable.

REFERENCES

[1] K. Tchoń, "Endogenous configuration space approach: an intersection of robotics and control theory," in *Nonlinear systems: Techniques for dynamical analysis and control*, ser. Lecture Notes in Control and Information Sciences, N. van de Wouw, E. Lefeber, and I.L. Arteaga, Eds. Cham: Springer, 2017, vol. 470, pp. 209–234.

[2] K. Tchoń and J. Jakubiak, "Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of Jacobian inverse kinematics algorithms," *Int. J. Control*, vol. 26, no. 14, pp. 1387–1419, 2003.

[3] A. Ratajczak, "Egalitarian versus prioritarian approach in multiple task motion planning for nonholonomic systems," *Nonlinear Dyn.*, vol. 88, no. 3, pp. 1733–1747, 2017.

[4] K. Zadarnowska and K. Tchoń, "Modeling and motion planning of wheeled mobile robots subject to slipping," in *RoMoCo'15: 10th International Workshop on Robot Motion and Control*, Poznań, Poland, 2015, pp. 78–83.

[5] A. Ratajczak and K. Tchoń, "Multiple-task motion planning of non-holonomic systems with dynamics," *Mech. Sci.*, vol. 4, no. 1, pp. 153–156, 2013.

[6] M. Janiak and K. Tchoń, "Constrained motion planning of nonholonomic systems," *Syst. Control Lett.*, vol. 60, no. 8, pp. 625–631, 2011.

[7] A. Ratajczak, "Trajectory reproduction and trajectory tracking problem for the nonholonomic systems," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 64, no. 1, pp. 63–70, 2016.

[8] W. Xu, C. Li, X. Wang, Y. Liu, B. Liang, and Y. Xu, "Study on non-holonomic Cartesian path planning of a free-floating space robotic system," *Adv. Robot.*, vol. 23, no. 1-2, pp. 113–143, 2009.

[9] W. Xu, Y. Liu, B. Liang, Y. Xu, C. Li, and W. Qiang, "Non-holonomic path planning of a free-floating space robotic system using genetic algorithms," *Adv. Robot.*, vol. 22, no. 4, pp. 451–476, 2008.

[10] I. Tortopidis and E. Papadopoulos, "On point-to-point motion planning for underactuated space manipulator systems," *Robot. Autonom. Syst.*, vol. 55, no. 2, pp. 122 – 131, 2007.

[11] I. Dulęba and I. Karcz-Dulęba, "Sub-optimal motion planning of one-chained, two-input nonholonomic systems," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 71, no. 3, p. e145684, 2023.

[12] G. Misra and X. Bai, "Task-constrained trajectory planning of free-floating space-robotic systems using convex optimization," *J. Guid. Control Dyn.*, vol. 40, no. 11, pp. 2857–2870, 2017.

[13] I. Dulęba and M. Opałka, "Motion planning of strongly controllable stratified systems," *Robotica*, vol. 34, no. 10, p. 2223–2240, 2016.

[14] N. Wada, S. Tagami, and M. Saeki, "Path-following control of a mobile robot in the presence of actuator constraints," *Adv. Robot.*, vol. 21, no. 5–6, pp. 645–659, 2007.

[15] T. Rybus, K. Seweryn, and J. Sąsiadek, "Control system for free-floating space manipulator based on nonlinear model predictive control (nmpc)," *J. Intell. Robot. Syst.*, vol. 85, no. 3-4, pp. 491–509, 2017.

[16] W. Domski and A. Mazur, "Input-output decoupling for a 3d free-floating satellite with a 3r manipulator with state and input disturbances," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 67, no. 6, pp. 1031–1039, 2019.

[17] T. Rybus *et al.*, "Motion controller for the TITAN robotic manipulator dedicated for on-orbit servicing operations," in *17th*

Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA), 2023.

[18] C. Ogundipea and A. Ellery, "Bio-inspired adaptive control of robotic manipulators for space debris removal and on-orbit servicing," in *17th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2023.

[19] S. Asci and A. Nanjangud, "Contact dynamics and autonomous control during rendezvous and berthing maneuvers," in *17th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2023.

[20] M. D'Ambrosio, L. Capra, and M. Lavagna, "Deep reinforcement learning for reactive ios space manipulator operations," in *17th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2023.

[21] K. Tchoń, "Endogenous configuration space approach in robotics research," in *Automatic Control, Robotics, and Information Processing*, P. Kulczycki, J. Korbicz, and J. Kacprzyk, Eds. Cham: Springer International Publishing, 2021, pp. 425–454.

[22] E.D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems (2nd Ed.)*. Berlin, Heidelberg: Springer-Verlag, 1998.

[23] T. Rybus *et al.*, "Application of a planar air-bearing microgravity simulator for demonstration of operations required for an orbital capture with a manipulator," *Acta Astronaut.*, vol. 155, pp. 211–229, 2019.

[24] E. Papadopoulos, "On the dynamics and control of space manipulators," Ph.D. dissertation, MIT, 1990.

[25] K. Tchoń and J. Ratajczak, "General lagrangian jacobian motion planning algorithm for affine robotic systems with application to a space manipulator," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2017, pp. 909–914.