J Intell Robot Syst (2015) 77:445–456
DOI 10.1007/s10846-013-9880-0

Parametric and Non-parametric Jacobian Motion Planning for Non-holonomic Robotic Systems

Adam Ratajczak · Krzysztof Tchoń

Received: 13 January 2013 / Accepted: 10 June 2013 / Published online: 22 September 2013

Abstract This paper addresses computational aspects of the Jacobian motion planning algorithms for non-holonomic robotic systems. The motion planning problem is formulated in terms of a control problem in the control affine system representing the system's kinematics. Jacobian motion planning algorithms are derived by means of the continuation (homotopy) method applied to the inverse kinematics problem in the space of control functions. The solution of the motion planning problem is obtained as the limit solution of a functional differential equation involving the control function. Two methods of representing the control functions are studied: parametric and non-parametric. The parametric method parametrizes the control functions by truncated orthogonal series. The non-parametric method can manage without the parametrization. The functional differential equation can be solved using either the Euler method of integration or higher order methods. The paper focuses on the non-parametric Jacobian pseudo inverse motion planning algorithms incorporating a higher order integration method. Performance of this algorithm is illustrated by the numeric solution of a motion planning problem for the rolling ball kinematics.

Keywords Non-holonomic system · Motion planning · Jacobian algorithm · Continuation method · Computations · Rolling ball

1 Introduction

The motion planning problem of robotic systems can be regarded as a special instance of the inverse kinematics problem, and solved by means of the continuation or homotopy method [1]. The applicability of this method to the path or motion planning of non-holonomic robots was first observed in [2], and then developed in [3–5]. A continuation method-based motion planning algorithm for convex rolling surfaces has been presented in [6]. A variation of the continuation method is the endogenous configuration space approach, originally dedicated to mobile manipulators, that explores systematically the analogies between stationary manipulators and mobile robotic systems [7, 8]. The endogenous configuration space approach will be adopted as a leitmotif in this paper.

This work was supported by the Wrocław University of Technology under a statutory grant.

A. Ratajczak (✉) · K. Tchoń
Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, ul. Janiszewskiego 11/17, 50–372 Wrocław, Poland
e-mail: adam.ratajczak@pwr.wroc.pl

K. Tchoń
e-mail: krzysztof.tchon@pwr.wroc.pl

All motion planning algorithms devised by the continuation method rely on a functional differential equation whose trajectory in the limit yields the control function solving the problem. In the references mentioned above, this equation is first discretized in accordance with the Euler integration method, and then reduced to a finite-dimensional iterative process by the approximation of the control functions by truncated orthogonal series. A preference to the Euler method results primarily from a wide acceptance of the Newton method as a computational tool [9]. Also, a certain tradition of using the iterative learning control scheme in robotics, pioneered by Arimoto et al. [10], might have been meaningful; for a connection of the endogenous configuration space approach with the iterative learning control see [11]. Thus, the motion planning algorithm returns coefficients of the control function with respect to an orthogonal base in the control space. Such an algorithm will be called parametric. In the context of endogenous configuration space approach a comparative study of motion planning algorithms employing diverse orthogonal bases, such as trigonometric functions, Legendre, Gegenbauer and Tchebyshev polynomials as well as Haar functions, can be found in [12]. Complementarily, in [13] the use of Laguerre polynomials has been recommended. The parametric algorithms are conceptually simple and often computationally efficient. However, for the reason of using the Euler integration method, and being approximate and base-dependent, the parametric algorithm does not guarantee a convergence of the "true" solution of the motion planning problem. In order to improve the quality of convergence of parametric motion planning algorithms a variable step length has been introduced into the Euler method. In [4] the step length adjustment was based on the line searching strategy of the minimum of error norm. The Armijo step adjustment method and a more complex method resulting from the affine covariant Lipschitz condition [9] imposed on the Jacobian was examined in [14]. A further improvement of the numerical properties of motion planning algorithms could be obtained after replacing the Euler (first order) method by more advanced, higher order integration schemes along with the step size optimization.

Recently, in [15] it has been observed that, by a proper organization of computations, the functional differential equation underlying the motion planning algorithm could be solved without the introduction of any parametrization. A motion planning algorithm based on such an approach will be referred to as non-parametric. From the viewpoint of the control functions representation and the integration method the motion planning algorithms can be divided into four classes: (1) parametric and Euler, (2) parametric and higher order, (3) non-parametric and Euler and (4) non-parametric and higher order. A majority of reported research concentrates on the class (1). The work [15] makes a first step toward the class (3). The main contribution of this paper consists in developing a non-parametric Jacobian motion planning algorithm for non-holonomic systems, based on a higher order integration method (so belonging to the class (4)). For comparison, the parametric and higher order computations (of class (2)) will be studied. Also, the convergence of the higher order integration method vs. the Euler scheme will be examined.

The comparison will be accomplished by numeric computations using an example motion planning problem of the rolling ball. Although not very complex, this example has been chosen for the reason that, differently to the car or car-with-trailers non-holonomic systems, the rolling ball's kinematics are not differentially flat [16], thus harder to control. In summary, the novelty of this paper lies in providing a non-parametric solution to the motion planning problem for non-holonomic robotic systems by ingenious organization of computations and adaptation of available numeric tools in order to solve the basic functional differential equation.

The organization of this paper is the following. Section 2 presents basic concepts of endogenous configuration space approach, and a derivation of the Jacobian motion planning algorithms based on the continuation method. Section 3 is devoted to the organization of numeric computations within the Jacobian pseudo inverse motion planning

algorithm. Results of numeric computations are contained in Section 4. Section 5 concludes the paper.

2 Basic Concepts

In control theoretic terms the kinematics of a nonholonomic robotic system are represented by a driftless control system with output

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

where the control variable $u \in R^m$, the state variable $q \in R^n$, and the output (task space) variable $y \in R^r$. The task space variable is the one subject to motion planning. The functions and vector fields appearing in Eq. 1 need to be smooth (C^∞). Let $T > 0$ denote a control time horizon. The admissible control functions in Eq. 1 belong to the space $L_m^2[0, T]$ of Lebesgue square integrable functions defined on the interval $[0, T]$. The space $L_m^2[0, T]$ is a Hilbert space with inner product

$$\langle u_1(\cdot), u_2(\cdot)\rangle_{\mathrm{R}} = \int_0^T u_1^T(t)\mathrm{R}(t)u_2(t)dt, \tag{2}$$

specified by a matrix $\mathrm{R}(t) = \mathrm{R}^T(t) > 0$ imposing certain weights on components of the control. The inner product induces the norm $||u(\cdot)||_{\mathrm{R}} = \langle u(\cdot), u(\cdot)\rangle_{\mathrm{R}}^{1/2}$ of the control function $u(\cdot)$. Let $q(t) = \varphi_{q_0,t}(u(\cdot))$ denote the state trajectory of Eq. 1, initialized at q_0 and driven by $u(\cdot)$. We shall assume the existence of $q(t)$ for every time instant $t \in [0, T]$. Given an initial state q_0 of system (1) and the time horizon T, the motion planning problem is formulated in the following way: find a control $u(t)$ steering the system's output at T to a desired point y_d in the task space, so that $y(T) = y_d$.

2.1 Endogenous Configuration Space Approach

Our analysis of the motion planning problem will be based on the concept of the end-point map of system (1), defined as the value at T of the output function resulting from the application of a control function $u(\cdot)$,

$$K_{q_0,T}(u(\cdot)) = k(q(T)) = k(\varphi_{q_0,T}(u(\cdot))). \tag{3}$$

For bounded measurable control functions $u(\cdot) \in \mathcal{X} \subset L_m^2[0, T]$ the end-point map $K : \mathcal{X} \longrightarrow R^r$ is continuously differentiable (C^1), [17]. In the context of mobile robots or mobile manipulators the space $\mathcal{X}$ has been called the endogenous configurations space, [8]. The derivative of the end-point map is computed by means of the linear approximation to system (1)

$$\begin{aligned} \dot{\xi}(t) &= A(t)\xi(t) + B(t)v(t), \\ \eta(t) &= C(t)\xi(t), \quad \xi(0) = 0, \end{aligned} \tag{4}$$

along the control-trajectory pair $(u(t), q(t))$, where

$$\begin{aligned} A(t) &= \frac{\partial(G(q(t))u(t))}{\partial q}, \ B(t) = G(q(t)), \\ C(t) &= \frac{\partial k(q(t))}{\partial q}. \end{aligned} \tag{5}$$

Given the linear system (4), the derivative of the end-point map at $u(\cdot) \in \mathcal{X}$ is equal to

$$DK_{q_0,T}(u(\cdot))v(\cdot) = \eta(T) = C(T)\xi(T). \tag{6}$$

In compliance with the robotic terminology, the derivative (6) will be called the system's Jacobian,

$$DK_{q_0,T}(u(\cdot)) = J_{q_0,T}(u(\cdot)).$$

It follows that the computation of the Jacobian involves the integration of the differential equation (4) from 0 to T at zero initial condition. If $\Phi(t, w)$ denotes the transition matrix of Eq. 4, so that

$$\frac{\partial\Phi(t, w)}{\partial t} = A(t)\Phi(t, w), \quad \Phi(w, w) = I_n, \tag{7}$$

then this means that the Jacobian

$$J_{q_0,T}(u(\cdot)) : \mathcal{X} \longrightarrow R^r$$

can be expressed as

$$J_{q_0,T}(u(\cdot))v(\cdot) = C(T)\int_0^T \Phi(T, w)B(w)v(w)dw. \tag{8}$$

The Jacobian allows to distinguish between regular and singular controls (endogenous configurations) of system (1). A control $u(\cdot) \in \mathcal{X}$ will be called regular, if the Jacobian is a surjective map onto R^r, otherwise the control $u(\cdot)$ is referred to as singular. It can be shown that at regular controls the control affine system (1) is locally controllable.

The Jacobian (8) can be regarded as a linear map $J_{q_0,T}(u(\cdot)) : \mathcal{X} \longrightarrow R^r$. Using the inner product (2) in the endogenous configuration space as well as the Euclidean structure of the task space, and making the identifications $\mathcal{X}^* \cong \mathcal{X}$, $R^{r*} \cong R^r$ of dual spaces, the dual Jacobian map

$$J^*_{q_0,T}(u(\cdot)) : R^r \longrightarrow \mathcal{X}$$

can be defined in the following way

$$\left(J^*_{q_0,T}(u(\cdot))\eta\right)(t) = \mathrm{R}^{-1}(t)B^T(t)\Phi^T(T,t)C^T(T)\eta. \tag{9}$$

The composition of the Jacobian and its dual

$$\begin{aligned} &J_{q_0,T}(u(\cdot))J^*_{q_0,T}(u(\cdot)) \\ &= \mathcal{G}_{q_0,T}(u(\cdot)) \\ &= C(T)\int_0^T \Phi(T,w)B(w)\mathrm{R}^{-1}(w) \\ &\quad \times B^T(w)\Phi^T(T,w)dwC^T(T) \end{aligned} \tag{10}$$

yields the output controllability Gramian associated with the linear system (4). In the robotics context (10) is referred to as the mobility (for mobile robots) or dexterity (for mobile manipulators) matrix of the system (1), as it has a form and plays a role analogous to that of the manipulability matrix of a robotic manipulator [18]. In particular, at regular control functions this matrix has full rank r. The mobility matrix can be computed by solving the Lyapunov differential equation

$$\dot{M}(t) = B(t)\mathrm{R}^{-1}(t)B^T(t) + A(t)M(t) + M(t)A^T(t), \tag{11}$$

with zero initial condition $M(0) = 0$, and making the substitution $\mathcal{G}_{q_0,T}(u(\cdot)) = C(T)M(T)C^T(T)$.

2.2 Jacobian Motion Planning

In terms of the end-point map (3), the motion planning problem in system (1) means computing a control function $u_d(\cdot)$, such that $K_{q_0,T}(u_d(\cdot)) = y_d$. The motion planning problem can be solved using the continuation method that leads to a Jacobian motion planning algorithm. To sketch a derivation, we begin by picking an arbitrary control function $u_0(\cdot) \in \mathcal{X}$. If this function solves the problem, we are done. Otherwise, we define in $\mathcal{X}$ a differentiable (C^1) curve $u_\theta(\cdot)$ parametrized by $\theta \in R$, and compute the task space error

$$e(\theta) = K_{q_0,T}(u_\theta(\cdot)) - y_d \tag{12}$$

along this curve. The curve in the endogenous configuration space is required to be a lift of the error path

$$\frac{de(\theta)}{d\theta} = -\gamma e(\theta) \tag{13}$$

vanishing exponentially to 0 with a prescribed decay rate $\gamma > 0$. By differentiation of the error we get the Ważewski-Davidenko equation [19, 20]

$$J_{q_0,T}(u_\theta(\cdot))\frac{du_\theta(\cdot)}{d\theta} = -\gamma(K_{q_0,T}(u_\theta(\cdot)) - y_d). \tag{14}$$

Finally, choosing any right Jacobian inverse

$$J^\#_{q_0,T}(u(\cdot)) : R^r \longrightarrow \mathcal{X},$$

i.e. a map that satisfies $J_{q_0,T}(u(\cdot))J^\#_{q_0,T}(u(\cdot)) = I_r$, the Eq. 14 is transformed into a functional differential equation

$$\frac{du_\theta(\cdot)}{d\theta} = -\gamma J^\#_{q_0,T}(u_\theta(\cdot))(K_{q_0,T}(u_\theta(\cdot)) - y_d) \tag{15}$$

in $\mathcal{X}$. By passing to the limit

$$u_d(t) = \lim_{\theta \to +\infty} u_\theta(t),$$

we obtain a solution to the motion planning problem.

The dynamic system (15) defines any Jacobian motion planning algorithm for the control system (1). A frequently used right Jacobian inverse is the Jacobian pseudo inverse, [8], resulting from a constrained minimization of the squared norm of the control function $v(\cdot)$

$$\min_{\{v(\cdot)\in\mathcal{X} | J_{q_0,T}(u(\cdot))v(\cdot)=\eta\}} ||v(\cdot)||^2_{\mathrm{R}}, \tag{16}$$

and defined as

$$\begin{aligned} &\left(J^{\#P}_{q_0,T}(u(\cdot))\eta\right)(t) \\ &= \mathrm{R}^{-1}(t)B^T(t)\Phi^T(T,t)C^T(T)\mathcal{G}^{-1}_{q_0,T}(u(\cdot))\eta. \end{aligned} \tag{17}$$

The matrix $\mathcal{G}_{q_0,T}(u(\cdot))$ standing in Eq. 17 is just the mobility matrix (Eq. 10). By design, the Jacobian pseudo inverse motion planning algorithm is well defined in regular endogenous configurations.

3 Organization of Numeric Computations

3.1 Parametric Approach

Usually, in order to make the computations efficient, a finite-dimensional representation of control functions by the truncated orthogonal series is utilized. This approach will be referred to as parametric. Let us introduce a row matrix $P(t) = [\phi_0(t), \phi_1(t), \ldots, \phi_p(t)]$ whose entries are some basic functions defined on the interval $[0, T]$. Then, the control function of system (1) may be represented as

$$u_\lambda(t) = P_s(t)\lambda, \tag{18}$$

where

$$P_s(t) = \mathrm{diag}\{P(t), P(t), \ldots P(t)\}$$

is a block diagonal matrix built of m copies of $P(t)$, $s = m(p+1)$, and $\lambda \in R^s$ denotes a vector of control parameters. The basic functions will assumed orthogonal with respect to the inner product (2) in the sense that

$$\int_0^T P_s^T(t)\mathrm{R}(t)P_s(t)dt = I_s \tag{19}$$

or, equivalently, $||u_\lambda(\cdot)||^2_{\mathrm{R}} = \lambda^T\lambda$. Steered by Eq. 18, the control system (1) takes the form

$$\begin{cases} \dot{q} = G(q)u_\lambda(t) \\ y = k(q). \end{cases} \tag{20}$$

Let $q_\lambda(t) = \varphi_{q_0,t}(u_\lambda(\cdot))$ be the trajectory of Eq. 20. With some abuse of notation we shall denote the end point map of Eq. 20 by $K_{q_0,T}(\lambda) = k(q_\lambda(T))$ and the parametric Jacobian by $J_{q_0,T}(\lambda)$. For $\mu \in R^s$, we compute

$$\begin{aligned} J_{q_0,T}(\lambda)\mu &= \frac{d}{d\alpha}\bigg|_{\alpha=0} k(q_{\lambda+\alpha\mu}(T)) \\ &= C_\lambda(T)\int_0^T \Phi_\lambda(T,t)B_\lambda(t)P_s(t)dt\mu, \end{aligned} \tag{21}$$

where matrices $A_\lambda(t)$, $B_\lambda(t)$, $C_\lambda(t)$ are given by Eq. 5 along $(u_\lambda(t), q_\lambda(t))$, and

$$\frac{\partial\Phi_\lambda(t,w)}{\partial t} = A_\lambda(t)\Phi_\lambda(t,w), \quad \Phi_\lambda(w,w) = I_s.$$

The parametric Jacobian

$$J_{q_0,T}(\lambda) = C_\lambda(T)\int_0^T \Phi_\lambda(T,t)B_\lambda(t)P_s(t)dt \tag{22}$$

is a matrix of dimension $r \times s$. It is easily observed that $J_{q_0,T}(\lambda) = C_\lambda(T)J_\lambda(T)$, where $J_\lambda(t)$ is obtained by integration of the matrix differential equation

$$\dot{J}_\lambda(t) = A_\lambda(t)J_\lambda(t) + B_\lambda(t)P_s(t), \tag{23}$$

with initial condition $J_\lambda(0) = 0$.

The motion planning problem for system (20) consists in computing a vector of control coefficients $\lambda \in R^s$ such that $K_{q_0,T}(\lambda) = y_d$. The continuation argument, analogous to that used in Section 2.2, leads to a Ważewski-Davidenko equation

$$J_{q_0,T}(\lambda(\theta))\frac{d\lambda(\theta)}{d\theta} = -\gamma(K_{q_0,T}(\lambda(\theta)) - y_d), \tag{24}$$

that can be converted into an ordinary differential equation for the curve $\lambda(\theta)$

$$\frac{d\lambda(\theta)}{d\theta} = -\gamma J^{\#}_{q_0,T}(\lambda(\theta))(K_{q_0,T}(\lambda(\theta)) - y_d), \tag{25}$$

by means of a right inverse $J^{\#}_{q_0,T}(\lambda)$ of the parametric Jacobian. The existence of the right inverse requires that $s \geq r$. Given a trajectory $\lambda(\theta)$ of Eq. 25, the solution of the motion planning problem is computed as the limit $\lambda_d = \lim_{\theta\to+\infty}\lambda(\theta)$. The motion planning algorithm based on Eq. 25 will be referred to as parametric in contrast to the algorithm (Eq. 15) that will be called non-parametric.

In what follows as the right Jacobian inverse we shall use the Jacobian pseudo inverse

$$J^{\#P}_{q_0,T}(\lambda) = J^T_{q_0,T}(\lambda)\mathcal{G}^{-1}_{q_0,T}(\lambda), \tag{26}$$

where $\mathcal{G}_{q_0,T}(\lambda) = J_{q_0,T}(\lambda)J^T_{q_0,T}(\lambda)$ is a parametric mobility matrix of system (20). It is well known that the inverse (26) results from the constrained minimization of the squared Euclidean norm of the vector of control parameters

$$\min_{\{\mu\in R^s | J_{q_0,T}(\lambda)\mu=\eta\}} \mu^T\mu. \tag{27}$$

It would be instructive to examine the connection between the parametric and the non-parametric Jacobian pseudo inverse motion planning algorithms based on, respectively, the finite dimensional dynamic system (25) driven by Eq. 26 and the infinite dimensional dynamic system (15) driven by Eq. 17. To this aim, assume that $u_\theta(t) = P_s(t)\lambda(\theta)$, and differentiate both sides with respect to θ. By invoking Eq. 15 along with Eq. 17, we get

$$\begin{aligned}\frac{du_\theta(t)}{d\theta} &= P_s(t)\frac{d\lambda(\theta)}{d\theta}\\ &= -\gamma J^{\#P}_{q_0,T}(u_\theta(\cdot))(K_{q_0,T}(u_\theta(\cdot)) - y_d)\\ &= -\gamma \mathrm{R}^{-1}(t)B^T_\theta(t)\Phi^T_\theta(T,t)C^T_\theta(T)\mathcal{G}^{-1}_{q_0,T}(u_\theta(\cdot))\\ &\quad\times(K_{q_0,T}(u_\theta(\cdot)) - y_d).\end{aligned}$$

By multiplying the terms no 2 and 4 above from the left by $\mathrm{R}(t)$ and $P^T_s(t)$, then integrating from 0 to T, and using orthogonality (19) as well as the definition of the parametric Jacobian (22), we conclude that

$$\begin{aligned}&\frac{d\lambda(\theta)}{d\theta}\\ &= -\gamma J^T_{q_0,T}(\lambda(\theta))\mathcal{G}^{-1}_{q_0,T}(u_\theta(\cdot))(K_{q_0,T}(u_\theta(\cdot)) - y_d).\end{aligned}$$

Now, since by definition $K_{q_0,T}(u_\theta(\cdot)) = K_{q_0,T}(u_{\lambda(\theta)}(\cdot)) = K_{q_0,T}(\lambda(\theta))$, the above identity will convert into Eq. 25 for the Jacobian pseudo inverse (26), on condition that for $u_\theta(t) = u_{\lambda(\theta)}(t) = P_s(t)\lambda(\theta)$ the non-parametric and the parametric mobility matrices are identical, i.e.

$$\begin{aligned}&\mathcal{G}_{q_0,T}(u_\theta(\cdot))\\ &= C_\lambda(T)\int_0^T \Phi_\lambda(T,w)B_\lambda(w)\mathrm{R}^{-1}(w)\\ &\quad\times B^T_\lambda(w)\Phi^T_\lambda(T,w)dwC^T_\lambda(T)\\ &= J_{q_0,T}(\lambda)J^T_{q_0,T}(\lambda),\end{aligned}$$

where $\lambda = \lambda(\theta)$. Suppose that the number s of control parameters has been chosen in such a way that for a given $\eta \in R^r$

$$\mathrm{R}^{-1}(t)B^T_\lambda(t)\Phi^T_\lambda(T,t)C^T_\lambda(T)\eta = P_s(t)\nu,$$

where $\nu \in R^s$ depends on η. Using this assumption and orthogonality (19), we compute the quadratic form

$$\eta^T\mathcal{G}_{q_0,T}(u_\theta(\cdot))\eta = \nu^T\int_0^T P^T_s(t)\mathrm{R}(t)P_s(t)dt\nu = \nu^T\nu.$$

On the other hand, by definition of the parametric Jacobian (22),

$$\begin{aligned}J^T_{q_0,T}(\lambda)\eta &= \int_0^T P^T_s(t)B^T_\lambda(t)\Phi^T_\lambda(T,t)dtC^T_\lambda(T)\eta\\ &= \int_0^T P^T_s(t)\mathrm{R}(t)P_s(t)dt\nu = \nu,\end{aligned}$$

so

$$\eta^T J_{q_0,T}(\lambda)J^T_{q_0,T}(\lambda)\eta = \nu^T\nu.$$

Finally, a combination of these identities results in the desired identity

$$\mathcal{G}_{q_0,T}(u_\theta(\cdot)) = J_{q_0,T}(\lambda)J^T_{q_0,T}(\lambda).$$

We conclude that, if the number of parameters is sufficiently big then the parametric and the non-parametric Jacobian pseudo inverses coincide, so the parametric and the non-parametric Jacobian pseudo inverse motion planning algorithms yield the same solution. Obviously, this happens in the limit $s \to +\infty$.

3.2 Non-parametric Approach

An alternative approach to solving the functional differential equation (15) is called non-parametric. The non-parametric approach consists in solving this equation directly. This means that for each value of θ we solve simultaneously a system of differential equations composed of the kinematics equations (1), the transition matrix equations (7), and the Lyapunov differential equation (11). Having solved these equations, one can write directly the algorithm equation (15) with the Jacobian pseudo inverse (17) substituted into it.

3.3 Integration Methods

Besides choosing the parametric or the non-parametric approach, another important computational problem is the choice of the integration

method. As we have already said, traditionally the functional differential equation (15) has been solved by means of the Euler method with constant or variable step length. This approach is easily implementable and often provides sufficient accuracy of solution. In accordance with the Euler method, the θ variable is discretized with a step length h, so at the ith step $\theta_i = \theta_{i-1} + h$. Setting $u_i(t) = u_{\theta_i}(t)$, the Eq. 15 is transformed into the difference equation

$$u_{i+1}(t) - u_i(t) = -h\gamma \left(J^{\#}_{q_0,T}(u_i(\cdot))(K_{q_0,T}(u_i(\cdot)) - y_d) \right)(t), \quad (28)$$

The main advantage of the Euler method is its simplicity and modest computational demand. On the other hand, this is a fixed-step and a first order method whose accuracy considerably depends on the step length, and that is lacking control over the computation error.

Alternatively, the Eq. 15 can be solved by means of a higher order integration method. We have chosen the *MATLAB* built-in variable-step length method using the Dormand-Prince pair [21]. This method is a combination of the fourth and the fifth order Runge-Kutta methods RK4(5). In every step the algorithm computes the function derivative using the Runge-Kutta method of orders fourth and fifth. Next, these two results are used to assess the computation accuracy. As long as the accuracy is high enough, no additional corrections have to be made. However, when the accuracy drops below a pre-assumed level then the step size is decreased. On the other side, when the computation accuracy is very high then the same procedure can be enrolled to increase the step size and speed-up the computations. Advantages of using this approach will be revealed in Section 4. As mentioned in Introduction, we shall focus on the non-parametric and higher order Jacobian motion planning algorithms that will be compared with the parametric and higher order algorithm.

3.4 Computational Procedure

The computation of a solution to the motion planning problem by the Jacobian pseudo inverse algorithm (15) is essentially tantamount to solving for $u_\theta(t)$ the following set of differential-algebraic equations (DAE)

$$\begin{cases} \dfrac{dq_\theta(t)}{dt} = G(q_\theta(t))u_\theta(t), & (29) \\ \dfrac{d\Phi_\theta(T,t)}{dt} = -\Phi_\theta(T,t)A_\theta(t), & (30) \\ \dfrac{dM_\theta(t)}{dt} = B_\theta(t)\mathrm{R}^{-1}(t)B_\theta^T(t) + A_\theta(t)M_\theta(t) + M_\theta(t)A_\theta^T(t), & (31) \\ \dfrac{du_\theta(t)}{d\theta} = -\gamma \mathrm{R}^{-1}(t)B_\theta^T(t)\Phi_\theta^T(T,t)C_\theta^T(T) \times \left(C_\theta(T)M_\theta(T)C_\theta^T(T)\right)^{-1} e(\theta), & (32) \\ A_\theta(t) = \dfrac{\partial(G(q_\theta(t))u_\theta(t))}{\partial q}, \; B_\theta(t) = G(q_\theta(t)), \\ C_\theta(t) = \dfrac{\partial k(q_\theta(t))}{\partial q}, & (33) \\ e(\theta) = y_\theta(T) - y_d = k(q_\theta(T)) - y_d, & (34) \end{cases}$$

with boundary conditions $q_\theta(0) = q_0$, $\Phi_\theta(T, T) = I_n$, $M_\theta(0) = 0$, and a given initial control function $u_0(t)$. It should be noticed that in Eqs. 29–34 the subscript θ means a dependence on the trajectory $q_\theta(t)$ corresponding to the currently computed control function $u_\theta(t)$. The system 29–34 needs to be solved in the following way. Given a control $u_\theta(t)$, we first solve the first three differential equations, and find the functions $q_\theta(t)$ from Eq. 29, $\Phi_\theta(T, t)$ from Eq. 30, and $M_\theta(t)$ from Eq. 31 for $t \in [0, T]$. Having determined the trajectory $q_\theta(t)$, we compute matrices $B_\theta(t)$, $C_\theta(t)$ and the error $e(\theta)$ using the Eqs. 33 and 34 respectively. A substitution of previously computed data defines the right hand side of the differential equation 32, and allows us to accomplish one step of its integration, resulting in computing $u_{\theta+h_\theta}(t)$ for every $t \in [0, T]$. The newly computed control function is then substituted again into the system 29–34, and the computations are repeated. For a sufficiently large value of θ this procedure returns a solution $u_d(t)$ of the motion planning problem. The presented arrangement of the DAE system 29–34 allows us to use the differential equation

solver built-in *MATLAB*. As long as in the system 29–34 there exist two independent variables, namely t and θ, two solvers need to be used, one nested inside the other one. An outline of the computer code illustrating the proposed computational procedure may look as follows

```
READ θ_max, u_0(t), h_θ,init, h_t,init, T, q_0   % input values
h_θ = h_θ,init                                   %
θ = 0                                            % initialization
REPEAT
    h_t = h_t,init                               %
    t = 0                                        %
    q_θ(0) = q_0                                 % initialization
    Φ_θ(T, T) = I_n                              %
    M_θ(0) = 0                                   %
    REPEAT
        q_θ(t + h_t) =RK45[t,q_θ(t)]             % incrementing q
        Φ_θ(T, t + h_t) =RK45[t,Φ_θ(T, t)]        % incrementing Φ
        M_θ(t + h_t) =RK45[t,M_θ(t)]              % incrementing M
        h_t =OPTIMAL-STEP[t]                     % correcting step length h_t, if necessary
        t = t + h_t                              % taking next value of t
    UNTIL t ≥ T
    u_θ+h_θ(t) =RK45[θ, u_θ(t)]                  % incrementing u
    h_θ =OPTIMAL-STEP[θ]                         % correcting step length h_θ, if necessary
    θ = θ + h_θ                                  % taking next value of θ
UNTIL θ ≥ θ_max
```

where "RK45" stands for the method based on Runge-Kutta pair of order fourth and fifth (the Dormand-Prince pair), and "OPTIMAL-STEP" updates the optimal step length to guarantee the pre-assumed computation accuracy. As it was already mentioned, the change of the step size depends on the difference between the solution obtained from the fourth and fifth order methods. In the presented listing one can observe the nested structure of the differential equation solvers: the solver for the independent variable t is nested inside the solver for the independent variable θ.

4 Results of Numeric Computations

In this section the Jacobian pseudo inverse motion planning algorithm will be utilized to solve a motion planning problem for the rolling ball. We begin with the introduction of a model of the ball's kinematics, then state the problem and implement the non-parametric and the parametric algorithm. Two series of numeric computations have been accomplished. The first series shows the performance of the parametric vs. the non-parametric algorithm, both using the higher order integration method. The second one focuses on the convergence of the non-parametric algorithm with the Euler scheme vs. the higher order integration method.

4.1 Rolling Ball

A schematic view of the rolling ball is depicted in Fig. 1. The ball can roll freely on the XY plane. The ball's coordinates are chosen as $q = (x, y, \phi, \theta, \psi)^T \in R^5$, where the x and y denote the position of the contact point P expressed in the space coordinate frame $X_0Y_0Z_0$. Angles ϕ and θ (azimuth and elevation) define the position of the contact point in spherical coordinates with respect to the body coordinate frame $X_BY_BZ_B$. The angle θ describes the ball's rotation angle

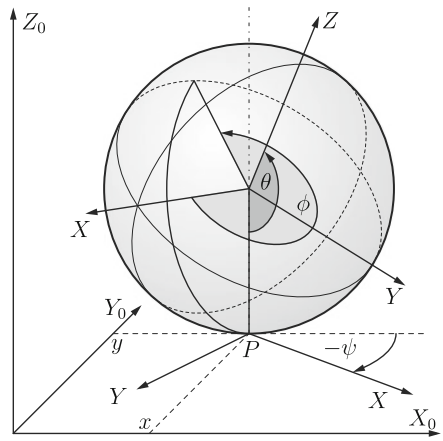

Fig. 1 Rolling ball

with respect to the space coordinate frame. There are two components of the control function $u = (u_1, u_2)^T \in R^2$, that can be interpreted as velocities of change of ϕ and θ. The coordinate representation of the rolling ball kinematics is the following, compare [22],

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{bmatrix} \sin\theta \sin\psi & \cos\psi \\ -\sin\theta \cos\psi & \sin\psi \\ 1 & 0 \\ 0 & 1 \\ -\cos\theta & 0 \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = G(q)u, \quad (35)$$

$$y = k(q) = (x, y, \psi), \quad (36)$$

where the output function reflects the assumption that the ball's position and orientation will be subject to motion planning. It is easily seen that the kinematics (35) are well defined when $0 < \theta < \pi$, so the ball cannot roll over its poles. A constrained motion planning algorithm that prevents the ball from rolling over the poles has been proposed in [23].

4.2 Simulation Results

For the kinematics (35) with output (36), the following motion planning problem will be addressed: Find a control $u(t)$ that drives the ball from the initial $q_0 = (0, 0, 0, \pi/4, 0)^T$ to the desired point $y_d = (1, 1, 0)^T$ in the task space over the time interval $[0, 2]$. This problem will be solved using the parametric and the non-parametric Jacobian pseudo inverse algorithms with higher order integration scheme. For both these algorithms we have chosen the error decay rate $\gamma = 4$, the matrix defining the inner product (2) $\mathrm{R}(t) = I_2$, the initial control $u_0(t) = (0.1, 0.2)^T$, and the stop condition $\|e(\theta)\| \leq 10^{-4}$. Although theoretically a solution of the problem is obtained when $\theta \to +\infty$, however we shall see that for this particular motion planning problem the desired accuracy is obtained for $\theta = \theta_{\max} = 3$. The parametric representation of the control function uses a truncated trigonometric series $\{1, \sin\left(\frac{2\pi}{T}t\right), \cos\left(\frac{2\pi}{T}t\right), \ldots, \sin\left(\frac{2k\pi}{T}t\right), \cos\left(\frac{2k\pi}{T}t\right)\}$ for $2k = p$. Numeric computations are made for $s = \{4, 6, 14, 22, 42, 62, 82, 102\}$. Their results are presented in Figs. 2, 3, 4 and 5.

It follows that the motion planning problem has been solved correctly. As can be seen from Fig. 2, the x and y coordinates variable have reached the desired point in the XY plane; also the orientation ψ angle at the end of the motion assumes the desired value. Figure 3 presents the control function $u_d(t)$ computed by the algorithms. In Figs. 2 and 3 the thin line shows the functions provided by the parametric algorithm, while the thick line refers to the non-parametric algorithm. To increase readability, the thin lines are marked with the number s of control coefficients. When the s number grows up, the parametric control function gets closer to the non-parametric one, as shown on the left hand side of Fig. 5, where $\varepsilon(s) = \|u(\cdot) - u_\lambda(\cdot)\|_{I_2}$ denotes the norm of the difference between these control functions for varied $\dim(\lambda) = s$. The method of solving the system (29)–(34) described in Section 3.4 allows to find the dependence of the control function $u_\theta(t)$ from both θ and t variables. This is presented in Fig. 4. One can see that the control function starts from the initial value $u_0(t)$ at $\theta = 0$, and reaches the solution $u_d(t)$ of the motion planning problem for $\theta = \theta_{\max}$. Finally, the right hand side of Fig. 5 refers to the convergence of the Jacobian pseudo inverse algorithms. It can be seen that the error convergence of the parametric and the non-parametric algorithm is the same, determined only

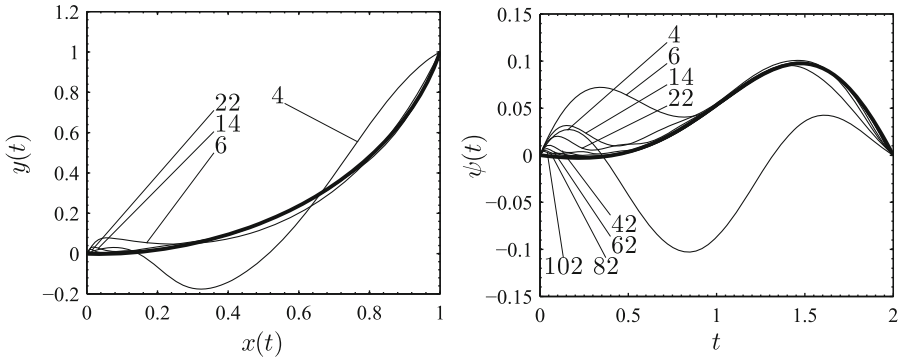

Fig. 2 Trace of contact point in XY plane (*left*) and trajectory of ball's orientation ψ (*right*)

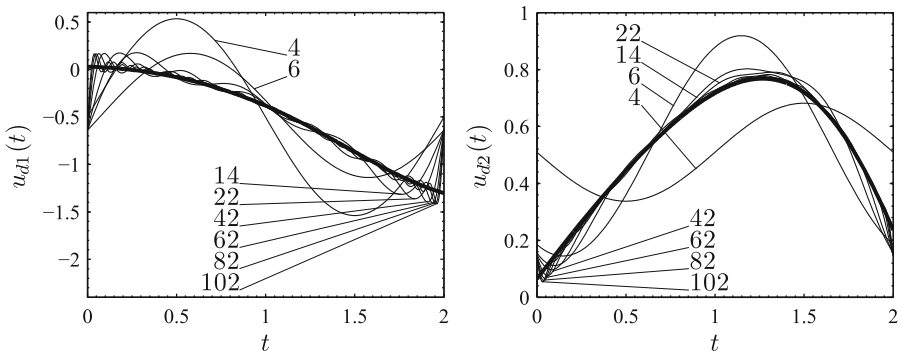

Fig. 3 Solution of motion planning problem: $u_{d1}(t)$ (*left*) and $u_{d2}(t)$ (*right*)

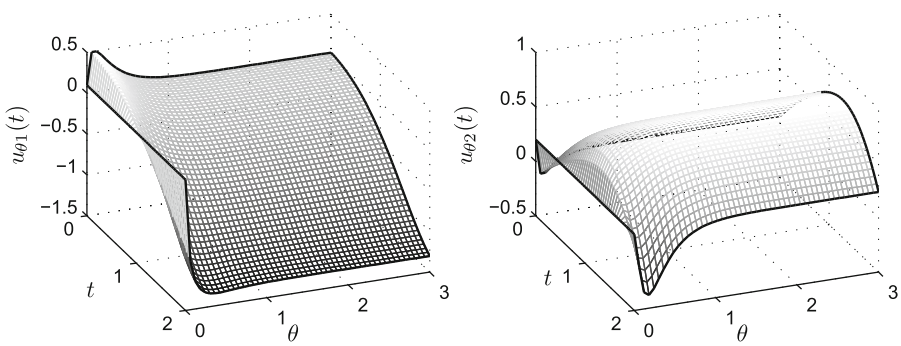

Fig. 4 Parametrized control function: $u_{\theta 1}(t)$ (*left*) and $u_{\theta 2}(t)$ (*right*)

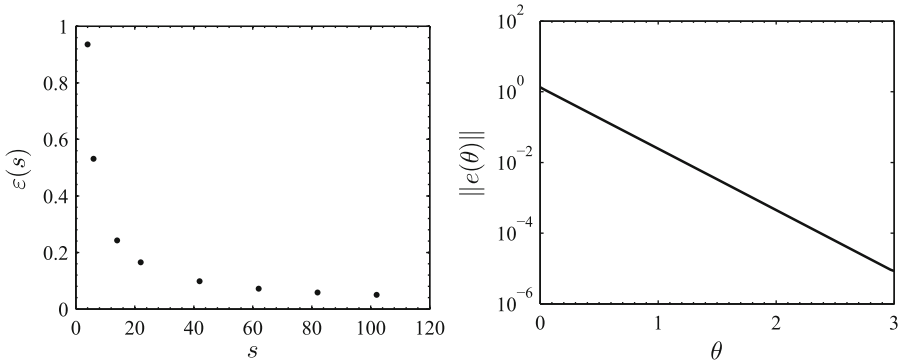

Fig. 5 Convergence of parametric to non-parametric (*left*) and algorithm convergence (*right*)

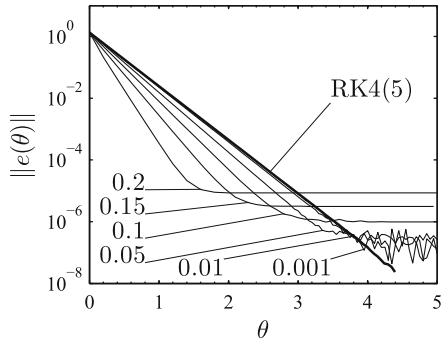

Fig. 6 Convergence of 1st order vs. higher order integration method

by the pre-assumed error decay rate γ. This is exactly what should be expected from Eq. 13.

To complement these results, the same motion planning problem will be solved by means of the non-parametric algorithm, first with the higher order integration method (Runge-Kutta—RK4(5)) and then by the first order, fixed step size method (the Euler scheme) with varied h_θ. The algorithm's convergence is summarized in Fig. 6, where the thick line refers to the higher order integration method while the thin lines present the convergence of the first order method with the varied step size. From the plot one can observe that, if the very high accuracy is not critical then the Euler scheme with relatively large step size is acceptable. As could be expected, decreasing the step size results in increasing the computation accuracy. Apparently, for the step size $h_\theta = 0.001$, convergence of the algorithm based on the Euler scheme is practically the same as that of RK4(5) method, and corresponds to pre-assumed error convergence ratio $\gamma = 4$ in Eq. 13. Computational properties of the examined integration methods can be assessed using the data collected in Table 1. One can observe that the Euler method with $h_\theta = 0.01$ needs about twice as much steps as the RK4(5) method. In the Euler scheme, the function standing on the right hand side of Eq. 32 is evaluated once for every step. In the higher order method this function has to be computed more than once in every step. The number of the function evaluations in Table 1 could be used to compare the computation effort for each integration method. As could be seen, this number for the first order method with step size $h_\theta = 0.001$ is almost four times bigger than in the case of the RK4(5) method.

5 Conclusion

This paper has been devoted to computational aspects of the Jacobian pseudo inverse motion planning algorithm of nonholonomic robotic systems. Special attention has been paid to the non-parametric representations of control functions. A thoughtful way of arranging the Eqs. 29–34 has been introduced, enabling to solve the basic functional differential equation of motion planning by means of a higher order (non-Euler) integration scheme.

Efficiency of this approach has been tested on the example motion planning problem of the rolling ball. Intuitively, it may be expected that the computations with the parametric algorithm are less time consuming. Nevertheless, the computation time depends on the number of the control coefficients s. The total computation time of the same problem achieved by the non-parametric version is comparable to the computation time of the parametric algorithm with $s = 30$. After an assignment of the computed control functions to the resulting rolling ball trajectories, one can observe that the trajectories provided by the non-parametric algorithm are smoother and more intuitive than those obtained from the parametric algorithm. It has been also confirmed by numerical computations that the increase of the number of control parameters s results in the parametric

Table 1 Comparison of examined integration methods

Algorithm	RK4(5)	Euler	Euler	Euler	Euler	Euler	Euler
Step size h_θ	–	0.2	0.15	0.1	0.05	0.01	0.001
No of steps	231	25	35	50	100	500	5000
No of function evaluation	1399	25	35	50	100	500	5000

control functions approaching the non-parametric ones.

Comparison of the integration methods has shown that, if the step size in the first order method is small enough then the obtained accuracy is similar as in the higher order method. However, the former method needs much more computational effort to produce the solution. Another advantage of the RK4(5) method is the control of computation accuracy by suitably adapting the step size. The numeric computations have also revealed a close to theoretic performance of the non-parametric, higher order Jacobian pseudo inverse motion planning algorithm. This contrasts with the performance observed, e.g. in [8].

Acknowledgements The authors are indebted to anonymous referees whose comments helped them to clarify some points and improved readability of this paper.

References

1. Tchoń, K.: Continuation method in robotics. In: Proc. 7th Conf. Computer Methods and Systems, pp. 17–24. Cracow, Poland (2007)
2. Sussmann, H.J.: A continuation method for nonholonomic path finding problems. In: Proc. 32nd IEEE CDC, pp. 2718–2723. San Antonio (1993)
3. Divelbiss, A.W., Wen, J.T.: A path space approach to nonholonomic motion planning in the presence of obstacles. IEEE Trans. Robot. Autom. **13**, 443–451 (1997)
4. Divelbiss, A.W., Seereeram, S., Wen, J.T.: Kinematic path planning for robots with holonomic and nonholonomic constraints. Essays on Mathematical Robotics, pp. 127–150. Springer-Verlag, New York (1998)
5. Chitour, Y.: A continuation method for motion-planning problems. ESAIM **12**, 139–168 (2006)
6. Alouges, F., Chitour, Y., Long, R.: A motion planning algorithm for the rolling-body problem. IEEE Trans. Robotics **26**, 827–836 (2010)
7. Tchoń, K., Muszyński, R.: Instantaneous kinematics and dexterity of mobile manipulators. In: Proc. 2000 IEEE Int. Conf. Robot. Automat., pp. 2493–2498. San Francisco (2000)
8. Tchoń, K., Jakubiak, J.: Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of Jacobian inverse kinematics algorithms. Int. J. Control **76**, 1387–1419 (2003)
9. Deuflhard, P.: Newton Methods for Nonlinear Problems. Springer-Verlag, Berlin (2004)
10. Arimoto, S., Kawamura, S., Miyazaki, F.: Bettering operation of robots by learning. J. Robot. Syst. **1**, 123–140 (1984)
11. Tchoń, K.: Iterative learning control and the singularity robust Jacobian inverse for mobile manipulators. Int. J. Control **83**, 2253–2260 (2010)
12. Jakubiak, J., Tchoń, K.: Fourier vs. non-Fourier band-limited Jacobian inverse kinematics algorithms for mobile manipulators. In: Proc. 10th IEEE Int. Conf. on Methods and Models in Automation and Robotics, pp. 1005–1010. Szczecin, Poland (2004)
13. Jung, A., Wen, J.T.: Nonlinear model predictive control for the Swing-Up of a rotary inverted pendulum. ASME J. Dyn. Syst. Meas. Control. **126**, 666–673 (2004)
14. Tchoń, K., Jakubiak, J.: Jacobian inverse kinematics algorithms with variable steplength for mobile manipulators. Advances in Robot Kinematics, pp. 465–472. Springer, Dordrecht (2006)
15. Ratajczak, A.: Motion planning of underactuated robotic systems. Ph.D. Dissertation, Wrocław University of Technology (2011)
16. Chelouch, A., Chitour, Y.: Motion planning of rolling surfaces. Forum Math. **15**, 727–258 (2003)
17. Sontag, E.D.: Mathematical Control Theory. Springer-Verlag, New York (1990)
18. Zadarnowska, K., Tchoń, K.: Kinematic dexterity of mobile manipulators: an endogenous configuration space approach. Robotica **21**, 521–530 (2003)
19. Ważewski, T.: Sur l'évaluation du domaine d'existence des fonctions implicites réelles ou complexes. Ann. Soc. Pol. Math. **20**, 81–120 (1947)
20. Davidenko, D.F.: On a new method of numerically integrating a system of nonlinear equations. Dokl. Akad. Nauk SSSR **88**, 601–603 (1953)
21. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**, 19–26 (1980)
22. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
23. Janiak, M., Tchoń, K.: Constrained motion planning of nonholonomic systems. Systems Control Lett. **60**, 625–631 (2011)