

Comparing Locally Energy-Optimal Newton Algorithms to Steer Driftless Systems

Ignacy Duleba, Janusz Jakubiak and Michal Opalka

Abstract In this chapter two locally energy-optimal motion planning methods were compared. Both of them are based on the Newton algorithm (primarily exploited to solve an inverse kinematic task in robotic manipulators) adapted to motion planning of driftless nonholonomic systems. The first, continuous method employs a standard technique of optimization in the null-space of a Jacobian matrix. The matrix is derived from a linearization of the system along a current trajectory and the trajectory corresponds to a finite Fourier series representation of controls. The second, discrete one, admits only piecewise-constant controls modified at a finite number of sub-intervals of the time horizon. Some simulations performed on a model of the free-floating planar double pendulum placed atop of a base revealed that the continuous method is more reliable, although, when the goal configuration does not need to be reached very accurately, the discrete one can be useful and energy-effective. A classification of the discrete methods was also proposed. According to the classification the proposed discrete method is of the 1st order.

1 Introduction

Many models of contemporary robots (mobile platforms and robots, free-floating robots, under actuated manipulators), considered at a kinematic level, belong to a class of driftless nonholonomic systems described by the equations

$$\dot{\mathbf{q}} = \sum_{i=1}^m \mathbf{g}_i(\mathbf{q}) u_i \quad (1)$$

I. Duleba (✉) · J. Jakubiak · M. Opalka
Institute of Computer Engineering, Control and Robotics, Wrocław University
of Technology, Janiszewskiego Street 11/17 50-372 Wrocław, Poland
e-mail: ignacy.duleba@pwr.wroc.pl

where $\mathbf{q} = (q_1, \dots, q_n)^T$ is a configuration vector, $\mathbf{u} = (u_1, \dots, u_m)^T$ are control inputs, and $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_m$ are smooth vector fields called generators of the system (1). A characteristic feature of the models (1) is that the number of state variables n is larger than the number of controls m . A motion planning task is to find controls $\mathbf{u}(t) = (u_1(t), \dots, u_m(t))^T$ that steer the system (1) from a given initial configuration \mathbf{q}_0 to a final one \mathbf{q}_f . Despite the fact that the condition $m < n$ makes this task difficult, still there are many methods to solve it (Duleba 1998; LaValle 2006). The most effective were designed for special subclasses of driftless systems [flat (Rouchon et al. 1993), nilpotent (Lafferriere and Sussmann 1991), in a chain form (Murray and Sastry 1993)]. However, still there is a need for general-purpose methods, especially equipped with some additional features (optimality, collision avoidance). In this chapter one of the methods will be considered, namely the Newton algorithm. Originally, it was designed to solve inverse kinematic tasks in robot manipulators, then extended to driftless nonholonomic systems (Divelbiss and Wen 1993; Duleba and Sasiadek 2003) by appropriately redefined kinematics (Tchon and Jakubiak 2003). The main idea of the Newton algorithm applied to nonholonomic systems is quite simple: for given initial controls, define a trajectory initialized at \mathbf{q}_0 , then linearize the system (1) along the trajectory, and, finally, modify the controls to force the end-point of the trajectory move towards the goal point \mathbf{q}_f . Applying this procedure iteratively, the goal configuration is reached if only the system (1) is controllable and a broad enough class of admissible controls is assumed. The main issue in modifying controls is their range. In continuous methods the controls are modified globally, over the whole time horizon, while for discrete methods more selective modifications of controls are possible. Both of the approaches have got some advantages and disadvantages. Continuous controls are usually easier to generate with power suppliers, while discrete methods are able to adjust trajectories only at those segments where it is the most necessary.

The chapter is organized as follows. In Sect. 2 two methods to determine modifications of controls will be discussed, both aimed at a local energy minimization (locality means that the energy optimal decisions are worked out for each iteration separately). The first (continuous) method modifies the whole trajectory by changing slightly coefficients of a selected basis of controls (for example, a Fourier basis but other continuous bases, like: Chebyshev, Legendre, Hermite, are also possible) from one iteration to another. The second (discrete) method admits only modifications of controls in a few selected sub-intervals (in this chapter only one among possible sub-intervals will be chosen to modify). In Sect. 3 simulations of the two methods carried out on the model of a planar double pendulum placed atop of a free-floating base are provided. Section 4 concludes the chapter.

2 Methods to Minimize Energy of Motion with the Use of the Newton Algorithm

Mathematically, the Newton algorithm is formalized as follows: for current controls $\mathbf{u}(\cdot)$ (initial controls are assumed) system (1) is linearized to get the form

$$\delta \dot{\mathbf{q}} = \mathbf{A}(t)\delta \mathbf{q} + \mathbf{B}(t)\delta \mathbf{u}, \quad (2)$$

with

$$\mathbf{A}(t) = \frac{\partial(\mathbf{G}(\mathbf{q}(t))\mathbf{u}(t))}{\partial \mathbf{q}}, \quad \mathbf{B}(t) = \mathbf{G}(\mathbf{q}(t))$$

where $\delta \mathbf{w}$ introduces a small variation of quantity $\mathbf{w} \in \{\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}\}$ and $\mathbf{q}(\cdot)$ denotes a trajectory of the system (1) initialized at \mathbf{q}_0 generated with controls $\mathbf{u}(\cdot)$. The current end-point $\mathbf{q}(T)$ of the trajectory corresponding to controls $\mathbf{u}(\cdot)$ should be moved towards the goal \mathbf{q}_f . Therefore, controls $\mathbf{u}(\cdot)$ are modified by $\delta \mathbf{u}(\cdot)$ leading to the displacement of $\delta \mathbf{q}(T)$ given by

$$\delta \mathbf{q}(T) = \int_0^T \mathbf{\Phi}(T, s) \mathbf{B}(s) \delta \mathbf{u}(s) ds, \quad (\delta \mathbf{q}(0) = 0), \quad (3)$$

where $\mathbf{\Phi}(t, s)$ is the fundamental matrix (Gantmacher 1988) satisfying the equation

$$\frac{d}{dt} \mathbf{\Phi}(t, s) = \mathbf{A}(t) \mathbf{\Phi}(t, s), \text{ with initial condition } \mathbf{\Phi}(s, s) = \mathbf{I}_n$$

(\mathbf{I}_n stands for the $(n \times n)$ identity matrix).

The kinematics of system (1) is the mapping

$$k_{\mathbf{q}_0, T} : L_m^2[0, T] \rightarrow R^n, \quad k_{\mathbf{q}_0, T}(\mathbf{u}(\cdot)) = \varphi_{\mathbf{q}_0, T}(\mathbf{u}(\cdot)), \quad (4)$$

where \mathbf{q}_0 is an initial state, T is a fixed time horizon, and $L_m^2[0, T]$ is a Cartesian product of m ($\dim U = m$) spaces of square integrable functions of time, defined on the interval $[0, T]$. The flow function (system trajectory) $\varphi_{\mathbf{q}_0, t}(\mathbf{u}(\cdot))$ describes evolution of the state over the time period $[0, t]$. In this chapter, $\varphi_{\mathbf{q}_0, t}(\mathbf{u}(\cdot))$ is also interpreted as the trajectory state reached at t .

In Eq. (3) one can influence $\delta \mathbf{u}(\cdot)$ to get desirable $\delta \mathbf{q}(T) = \xi(\mathbf{q}_f - \mathbf{q}(T))$, where a (small) positive value ξ is used to preserve validity of the linearization (2). This equation has got a nice physical interpretation, a final $\delta \mathbf{q}(T)$ can be viewed as a superposition of small displacements caused by $\delta \mathbf{u}(s)$, $s \in [0, T]$ and transformed by the operator $\mathbf{\Phi}(T, s) \mathbf{B}(s)$ from the time point s to the point T .

In the following subsections, two methods to determine $\delta \mathbf{u}(\cdot)$ will be reported.

2.1 Optimization of Energy in Null Space of the Jacobian Matrix

First method, well known in robotic literature (Duleba and Sasiadek 2003) is the following: a functional orthogonal basis is selected and controls are described in the form of a series

$$u_k(t) = \sum_{j=1}^{N^k} \lambda_j^k \phi_j^k(t), \quad k = 1, \dots, m, \quad t \in [0, T], \quad (5)$$

where N^k is the number of items selected to express the k th control, $\phi^k(t)$ are some of the very first elements of the Fourier basis with its harmonics

$$(1, \cos(k\omega t), \sin(k\omega t))^T, \quad k = 1, \dots, \quad \text{where } \omega = 2\pi/T.$$

In this way, the infinite dimensional functional space $L_m^2[0, T]$ was replaced by a finite dimensional space of coefficients λ , $\dim(\lambda) = \sum_{k=1}^m N^k$. It is worth noticing that different controls can be spanned by different harmonics (for example $u_1(t)$ can be a linear combination of the basis elements 1 and $\sin(\omega t)$ while $u_2(t)$ can be spanned by 1 and $\cos(2\omega t)$). The total energy of controls (5) is given by the formula

$$f(\lambda) = \frac{T}{2} \left(\sum_{k=1}^m \left\{ 2(\lambda_1^k)^2 + \sum_{i=2}^{N^k} (\lambda_i^k)^2 \right\} \right), \quad (6)$$

where $\lambda^k = (\lambda_1^k, \lambda_2^k, \dots, \lambda_{N^k}^k)^T$, $k = 1, \dots, m$ is the vector of coefficients of the k th control (5) and λ gathers all the vectors. Substituting (5) into (3) we get

$$\delta \mathbf{q}(T) = \int_0^T \Phi(T, s) \mathbf{B}_\lambda(s) ds \cdot \delta \lambda = \mathbf{J}_{q_0, T}(\mathbf{u}(\cdot, \lambda)) \cdot \delta \lambda, \quad (7)$$

where $\mathbf{J}_{q_0, T}(\mathbf{u}(\cdot, \lambda)) = \mathbf{J}_{q_0, T}(\lambda)$ denotes the Jacobian matrix and

$$\mathbf{B}_\lambda(s) = [B_1 \phi_1^1, \dots, B_1 \phi_{N_1}^1, B_2 \phi_1^2, \dots, B_2 \phi_{N_2}^2, \dots, B_m \phi_1^m, \dots, B_m \phi_{N_m}^m],$$

where B_i , $i = 1, \dots, m$ are columns of the matrix \mathbf{B} . The search of controls is performed within the space of their coefficients $\Lambda \ni \lambda$ as the basis functions $\phi(t)$ are fixed. The vector λ is changed according to an iterative formula introducing the Newton algorithm for nonholonomic systems with the optimization in the null space of the Jacobian matrix (the last term in the following equation) (Nakamura 1991).

$$\begin{aligned} \lambda_{i+1} = & \lambda_i + \xi_1 \cdot \mathbf{J}_{\mathbf{q}_0, T}^\#(\lambda_i) (\mathbf{q}_f - \mathbf{k}_{\mathbf{q}_0, T}(\mathbf{u}(\cdot, \lambda_i))) \\ & - \xi_2 \left(\mathbf{I} - \mathbf{J}_{\mathbf{q}_0, T}^\#(\lambda_i) \mathbf{J}_{\mathbf{q}_0, T}(\lambda_i) \right) \frac{\partial f(\lambda)}{\partial \lambda}, \end{aligned} \quad (8)$$

where $\mathbf{J}^\# = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$ is the Moore–Penrose pseudo-inverse matrix inversion, \mathbf{q}_f is a goal state, ξ_i are given coefficients and the initial value λ_0 is uniquely determined by the initial value of \mathbf{u} .

The continuous version of the Newton algorithm for controllable systems (1) guarantees an existence, in each iteration, of such $\delta \mathbf{u}_i$ that moves $\mathbf{q}_i(T)$ towards \mathbf{q}_f . However, it changes controls in the whole interval $[0, T]$. It is possible that controls in some sub-intervals may disrupt effective approaching to the goal caused by the other pieces of controls. To avoid this drawback, a discrete way to modify controls will be proposed.

2.2 Optimization of Energy in One Sub-Interval of Controls

The other way to optimize energy of controls in one iteration progresses as follows: the interval $[0, T]$ is partitioned into N sub-intervals and in each sub-interval only constant-value controls are admissible. In order to simplify implementation, the intervals are assumed to be of the same length, although from a theoretical point of view their length should be correlated with model-dependent quantities $\mathbf{A}(s)$, $\mathbf{B}(s)$, $\Phi(T, s)$ on each sub-interval. In the s_i th sub-interval, $i \in \{1, \dots, N\}$, a hyperplane spanned by column-vectors of the fixed $(n \times m)$ matrix $\Phi(T, s_i)\mathbf{B}(s_i)\Delta s$, $\Delta s = T/N$, is defined. In this hyperplane one can move along by changing $\delta \mathbf{u}(s_i)$, aimed at getting desired $\delta \mathbf{q}(T)$. Unfortunately, $\delta \mathbf{q}(T)$ may not belong to the hyperplane as the hyperplane is m -dimensional and designed motion $\delta \mathbf{q}(T)$ lives in $n > m$ dimensional space. All what can be done is to project $\delta \mathbf{q}(T)$ into the hyper-plane, using the Gram-Schmidt orthogonalization procedure, to get decomposition $\delta \mathbf{q}(s_i) = \delta \mathbf{q}^\perp(s_i) + \delta \mathbf{q}^\parallel(s_i)$. The component $\delta \mathbf{q}^\parallel(s_i)$ placed on the hyperplane can be compensated by admissible $\delta \mathbf{u}(s_i)$, while the $\delta \mathbf{q}^\perp(s_i)$ component cannot.

Those $\delta \mathbf{u}(s_i)$ when added to the current $\mathbf{u}(s_i)$ in the interval change the energy on motion (by increasing or decreasing it). This interval which causes the motion towards the goal (this requirement guarantees convergence of the process) with the maximal energy decrease is selected and used to change $\mathbf{u}(\cdot)$ for the next iteration. Formally, the criterion function which admits modifications of controls in a single sub-interval is given by the expression

$$\begin{aligned} K(s_i) &= (\langle \mathbf{u}(s_i), \mathbf{u}(s_i) \rangle - \langle \mathbf{u}(s_i) + \delta \mathbf{u}(s_i), \mathbf{u}(s_i) + \mathbf{u}(s_i) \rangle) \cdot \left\| \frac{\delta \mathbf{q}^\parallel(s_i)}{\delta \mathbf{q}(s_i)} \right\|, K(s^*) \\ &= \max_{i=1, \dots, N} K(s_i), \end{aligned} \quad (9)$$

where $\langle \cdot, \cdot \rangle$ denotes inner product while $\|\cdot\|$ is the Euclidean metrics and s^\star denotes the optimal interval (really modified in the current iteration). The function $K(s_i)$ in (9) is a product of two terms: the first one measures positive or negative changes of energy due to $\delta \mathbf{u}(s_i)$ acting on the sub-interval corresponding to s_i , while the second one, taking a value from the range $[0, 1]$, evaluates how effective approaching to the goal is. In practical situations those sub-intervals with very small component $\|\delta \mathbf{q}(s_i)\|$ should be excluded from the maximization process in (9) not to slow down the convergence process. Obviously, the two components in Eq. (9) can be weighted to form other criterion functions.

3 Simulations

Continuous and discrete methods that locally minimize the energy of motion were compared on the model of a free-floating space robot depicted in Fig. 1 and characterized by the data $m_0 = 5$, $m_1 = m_2 = 1$, $l_1 = l_2 = 1$, $a = 1$, $b = 0.6$. Using principles of linear and angular momentum (Dubowsky and Papadopoulos 1993) laws (initial momenta were assumed to vanish) the following equations were derived

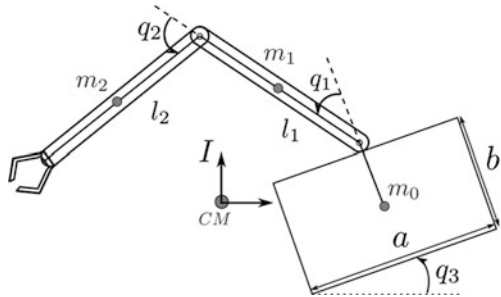
$$\begin{aligned} \dot{\mathbf{q}} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \\ A_1(q_1, q_2) \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 1 \\ A_2(q_1, q_2) \end{bmatrix} u_2 = g_1(\mathbf{q})u_1 + g_2(\mathbf{q})u_2 \\ &= G(\mathbf{q})\mathbf{u}, \end{aligned} \quad (10)$$

where $n = 3$, $m = 2$ and

$$\begin{aligned} A_1(q_1, q_2) &= -(76 + 135c_{q_1} + 33c_{q_2} + 45c_{q_1q_2})/A, \\ A_2(q_1, q_2) &= -(23 + 16.5c_{q_1} + 4.5c_{q_1q_2})/A \\ A &= 105.2 + 27c_{q_1} + 33c_{q_2} + 9c_{q_1q_2}, \quad \mathbf{u} = (\dot{q}_1, \dot{q}_2)^T. \end{aligned}$$

It is worth noticing that the linear momentum conservation law generates holonomic constraints while the angular momentum conservation law produces

Fig. 1 Free-floating 2D pendulum placed atop of a base



nonholonomic constraints. Holonomic constraints were incorporated into the model to reduce the natural configuration space $(q_1, q_2, q_3, x, y)^T$ into the final one $(q_1, q_2, q_3)^T$. Based on the Chow theorem (Chow 1939) that provides the small-time local controllability condition $\text{rank}(LA(\mathbf{G}))(\mathbf{q}) = n$, it was verified that the model (10) satisfies the condition (vector fields \mathbf{g}_1 and \mathbf{g}_2 and their descendants (generated with the Lie brackets) span the configuration space everywhere) so it is also controllable.

Two motion planning tasks were considered with the following common data: the time horizon $T = 1$, the number of sub-intervals $N = 50$, initial controls $u_1(t) = u_2(t) = \cos(\omega t)$, $\omega = 2\pi/T$, $t \in [0, T]$ (for the discrete method appropriately replaced with piecewise-constant values). For the continuous method both controls were searched within the family $\lambda_1^k + \lambda_2^k \cos(\omega t) + \lambda_3^k \sin(\omega t)$, $k = 1, 2$, containing constant term and the first harmonics.

Task 1 was designed to move the robot from $\mathbf{q}_0 = (-45^\circ, 90^\circ, 60^\circ)^T$ to $\mathbf{q}_f = (20^\circ, 15^\circ, 30^\circ)^T$ while Task 2 from $\mathbf{q}_0 = (0^\circ, 0^\circ, 0^\circ)^T$ to $\mathbf{q}_f = (-90^\circ, 60^\circ, 45^\circ)^T$. Resulting controls and trajectories for Task 1 were presented in Fig. 2 while for Task 2 in Fig. 3. In Fig. 4 a motion animation for Task 2 was provided. Numerical characteristics collected while solving the tasks were presented in Table 1.

From the simulations provided and those which were carried out but not reported here we can draw the following conclusions. The continuous method usually solves motion planning tasks with any accuracy assumed if only the number of parameters of controls is large enough comparing to the dimensionality of the configuration space (in our case the number of parameters was equal to $3 + 3 = 6$). However, for some tasks the energy expenditure on controls can be

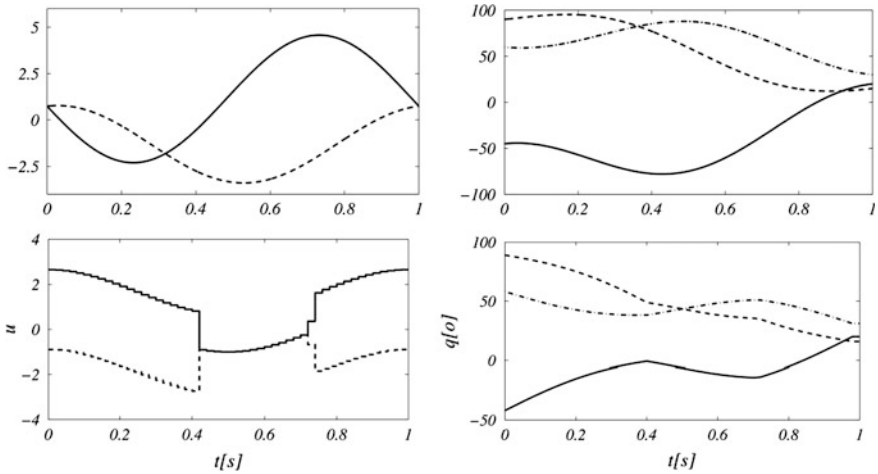


Fig. 2 Task 1: controls (left column, u_1/u_2 solid/slanted line) and trajectories (right column, $q_1/q_2/q_3$ —solid/slanted/dotted line) generated with the continuous method (first row) and discrete one (second row)

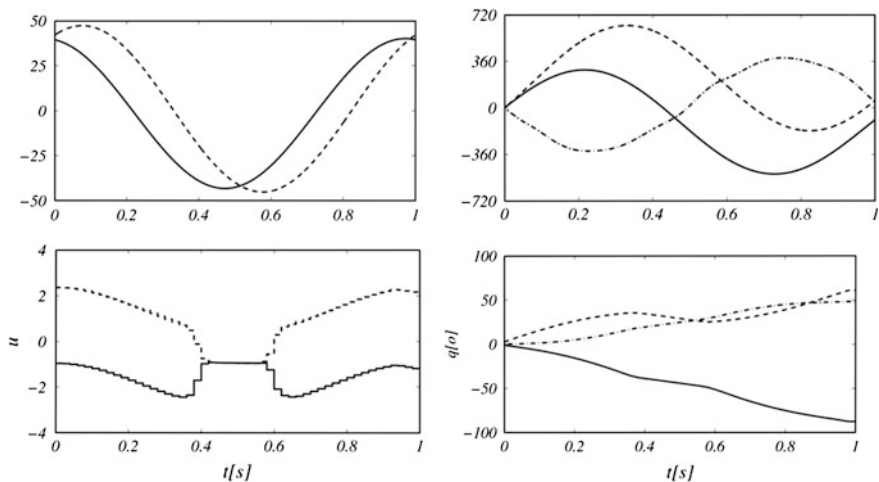


Fig. 3 Task 2: controls (*left column*, u_1/u_2 solid/slanted line) and trajectories (*right column*, $q_1/q_2/q_3$ —solid/slanted/dotted line) generated with the continuous method (*first row*) and discrete one (*second row*)

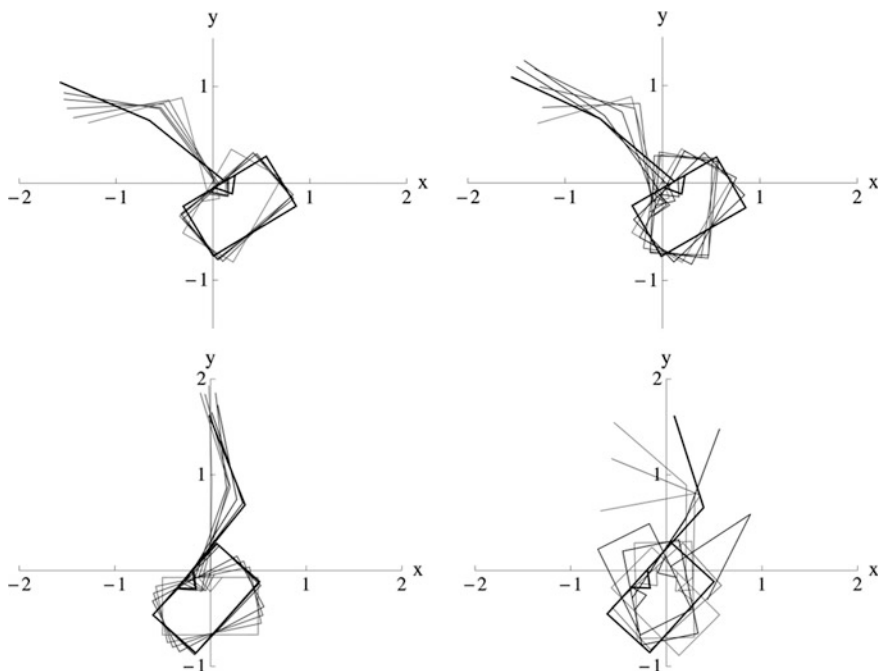


Fig. 4 Animation of motion: Task 1/2—first/second row; discrete/continuous method—left/right column

Table 1 Final error (err) in reaching the goal configuration, energy expenditure and the number of iteration to complete the computations, for Task 1 and Task 2 using the continuous and the discrete method

Task	Continuous			Discrete		
	Err($^\circ$)	Energy	Iter	Err($^\circ$)	Energy	Iter
1	8.9×10^{-9}	11.09	420	1.37	5.17	36
2	9.4×10^{-6}	1938.1	176	4.32	5.01	200

quite large (cf. Task 2). Also efficiency of solving the motion planning task can strongly depend on given initial controls.

The discrete method, admitting modification only one sub-interval of controls, does not allow to decrease the error to very small values. For some tasks (e.g., Task 1) the final error can be acceptable from a practical point of view but for the other ones (e.g., Task 2) not necessary. When the discrete method stops, all hyperplanes spanned by $\Phi(T, s_i)\mathbf{B}(s_i)$ (for each s_i , $i = 1, \dots, N$) are practically perpendicular to the $\delta\mathbf{q}(T)$, so controllable components $\delta\mathbf{q}^\parallel(s_i)$ are very small. Consequently, no modifications performed on a single sub-interval can cause approaching the end-point of the trajectory towards the goal.

The discrete method can be improved by admitting multi-interval modifications but it can drastically increase computational complexity of the method. In the limit, when N tends to infinity, both of the methods (discrete and continuous) are the same. But also more subtle, and probably not so computationally expensive, modifications of the discrete method are possible. To explain this statement some Lie algebraic terms are indispensable.

Generators of the system (1) can be considered as formal Lie monomials and generators of a free Lie algebra (Serre 1964). With each formal generator its degree is associated. More complex Lie monomials are obtained using Lie bracket operator. A degree of a Lie monomial counts how many formal generators are used to get the monomial. Consequently, generators are degree one Lie monomials. Using the Campbell-Baker-Hausdorff-Dynkin formula (Duleba and Sasiadek 2001), higher degree Lie monomials (vector fields) can be generated by switching on and off piecewise-constant controls. For example, to generate the second degree vector field $[\mathbf{g}_1, \mathbf{g}_2]$ over the small time period Δs the following controls can be applied: $u_1 = (+1, 0, -1, 0)$, $u_2 = (0, +1, 0, -1)$, and each interval lasts $\sqrt{\Delta s}$. From this example one can notice that motion along vector field $[\mathbf{g}_1, \mathbf{g}_2]$ is more energy consuming than a motion along generators $\mathbf{g}_1, \mathbf{g}_2$. A generalization of this observation for higher degree vector fields is also valid.

Now, the discrete method can be explained in Lie algebraic terms. The motions in each particular sub-interval characterized by s_i and coded by $\mathbf{B}(s_i)$ matrix can be viewed as a motion in linear spaces spanned by generators \mathbf{g}_i , $i = 1, \dots, m$ of the system (1) and shifted by the operator $\Phi(T, s_i)$. If they cannot help to solve the motion planning task, so, according to Chow's theorem (Chow 1939), they must be supported by higher degree vector fields $[\mathbf{g}_i, \mathbf{g}_j]$, $[\mathbf{g}_k, [\mathbf{g}_i, \mathbf{g}_j]]$, $i, j, k = 1, \dots, m$. As the system (1) is controllable, so there exist some finite degree vector fields that

force a motion of the end-point of a current trajectory towards the goal. With each discrete method its order can be associated. The order is inherited from a maximal degree vector field used to generate controls. According to this definition, the presented discrete method is of the first order. When more complex vector fields should be applied, an optimization procedure is used to determine at which time stamp s_i the generation of higher degree vector fields should be initialized. Scenarios of switching controls become even more cumbersome (for example, to generate third degree vector field 8 pieces of controls are used). Therefore, higher degree vector fields are to be generated only if lower degree vector fields fail to move $\mathbf{q}(T)$ towards \mathbf{q}_f .

Continuous and discrete methods should combine two contradictory factors: low computational complexity and a guaranteed convergence. By extending the number of parameters to change (number of harmonics in the continuous method or number of sub-intervals with varied controls) the chance of solving the task will increase but simultaneously computational complexity increases.

A singularity phenomenon is tightly bonded to an inverse kinematic task. For robot manipulators at singular configurations, the Jacobian matrix loses its maximal rank and the inverse task becomes ill-posed. For driftless nonholonomic systems (1) the problem of singularity is not so crucial. Firstly, because singular configurations cannot be computed analytically as the Jacobian matrix is based on the numerical data ($\mathbf{A}(s)$, $\mathbf{B}(s)$, $\Phi(T, s)$). Secondly, the Jacobian matrix can be extended easily by adding some more harmonics to continuous controls (or applying higher order discrete methods). In typical case, extra columns of the Jacobian matrix allow to avoid singularities. However, the aforementioned methods to deal with singularities will increase the computational complexity. In practical situations a compromise should be made between flexibility in controls' representation and computational costs. However, the costs should not be over-estimated as they are kinematic in nature (contrary to very extensive computations involving robot dynamics), usually low dimensional state spaces ($n \leq 6$) and some computations can be optimized (for example using the identity $\Phi(T, s_1) = \Phi(T, s) \cdot \Phi(s, s_1)$, $s < s_1 < T$) and parallelized.

4 Conclusions

In this chapter two methods of nonholonomic motion planning for driftless systems based on the Newton algorithm were compared. The first, continuous one uses the Fourier basis to express controls and utilizes optimization in the null-space of the Jacobian matrix. When the representation of controls is rich enough, it solves motion planning tasks for controllable systems with any selected accuracy. For some tasks, to preserve the convergence of the algorithm, quite energy-expensive trajectories were generated with this method. For a discrete method, admitting modifications of controls only in one sub-interval, it may happen that the motion planning task cannot be solved with any assumed accuracy. The discrete method

prefers energy-cheap trajectories. Some hints to improve convergence of the discrete method were also formulated. Both of the methods are local and strongly depend on initial controls assumed.

References

- Chow WL (1939) Über systeme von linearen partiellen Differentialgleichungen erster Ordnung. *Math Annalen* 117(1):98–105
- Divebiss AW, Wen J (1993) Nonholonomic path planning with inequality constraints. *IEEE Conf Decis Control* 3:2712–2717
- Dubowsky S, Papadopoulos E (1993) The kinematics, dynamics, and control of free-flying and free-floating space robotic system. *IEEE Trans Rob Autom* 9(5):531–542
- Duleba I (1998) Algorithms of motion planning for nonholonomic robots. Publishing House of Wrocław University of Technology, Wrocław
- Duleba I, Sasiadek J (2001) Calibration of control in steering nonholonomic systems. *Control Eng Pract* 9(2):217–225
- Duleba I, Sasiadek J (2003) Nonholonomic motion planning based on Newton algorithm with energy optimization. *IEEE Trans Control Sys Technol* 11(3):355–363
- Gantmacher FR (1988) Matrix theory. Nauka, Moscow
- Rouchon P, Fliess M, Levine J, Martin P (1993) Flatness, motion planning and trailer systems. *IEEE Conf Decis Control* 3:2700–2705
- Lafferriere G, Sussmann H (1991) Motion planning for controllable systems without drift. *IEEE Conf Robot Autom* 2:1148–1153
- LaValle SM (2006) Planning algorithms. Cambridge University Press, Cambridge
- Murray RM, Sastry SS (1993) Nonholonomic motion planning: steering using sinusoids. *IEEE Trans Autom Control* 38(5):700–716
- Nakamura Y (1991) Advanced robotics: redundancy and optimization. Addison Wesley, New York
- Serre J-P (1964) Lie algebras and lie groups. Lecture notes in Mathematics, Springer
- Tchon K, Jakubiak J (2003) Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of Jacobian inverse kinematics algorithms. *Int J Control* 26(14):1387–1419

Aerospace Robotics

Selected Papers from I Conference on Robotics in
Aeronautics and Astronautics

Sąsiadek, J. (Ed.)

2013, XII, 171 p. 129 illus., 102 illus. in color.,

Hardcover

ISBN: 978-3-642-34019-2