Journal of Intelligent & Robotic Systems (2019) 93:669–686
https://doi.org/10.1007/s10846-018-0871-z

Closing the Loop – Predictive Lifted Newton Trajectory Tracking Algorithm

Mariusz Janiak[1] · Łukasz Chojnacki[1]

Received: 30 October 2017 / Accepted: 11 May 2018 / Published online: 23 May 2018

Abstract
This paper introduces a predictive closed-loop trajectory tracking algorithm for nonlinear control systems that combines the Model Predictive Control (MPC) approach with the task priority Lifted Newton method. The optimal control problem within MPC is replaced by the open-loop trajectory tracking problem formulated as a constrained motion planning problem. Constraints reflect the distance in the task space between the system current output and the desired trajectory. The original constrained motion planning problem is replaced by an unconstrained one addressed in an extended control system representation, and solved with the task priority version of the Lifted Newton method. All other steps of the MPC scheme remains unchanged. Performance of the closed-loop predictive Lifted Newton trajectory tracking algorithm has been demonstrated with series of computer simulations for the kinematic car type platform.

Keywords Predictive scheme · Trajectory tracking · Lifted Newton · Task priority

1 Introduction

Typically, in the perspective of robotics, the trajectory tracking problem is considered as a feedback motion control task [21, 23, 24, 36]. The trajectory tracking problem regarded as a planning problem, has been studied in [20] where the continuation method has been used for design of the so-called reproduction equation, and then plan a trajectory of the rolling ball in the task space. In [30], the Endogenous Configuration Space Approach (ECSA) has been applied to a continuous inverse problem, where the author has defined an instantaneous kinematics of a nonholonomic system equipped with a Jacobian inverse algorithm. The ECSA framework has been also used in [17] to formulate the trajectory tracking problem as an open-loop planning problem. The original motion planning through waypoints problem [18], has been expanded with the additional task responsible for dragging the tracking error to zero between wayponits. Formally, the trajectory tracking problem has been defined as a constrained motion planning problem [19], whereas constraints reflect the instantaneous distance between the system output and the desired trajectory. Constraints have been incorporated into the system through extending the system by extra state variables. As a result, the trajectory tracking problem has been made equivalent to an unconstrained motion planning problem formulated in the extended system, and solved with the task priority version of the Lifted Newton method [1, 16]. In this context, the task priority Lifted Newton algorithm can be regarded as "multiple shooting" version of a ECSA Jacobian algorithm [31].

Due to lack of feedback in the approach presented in [17] , the evolution of the system's actual state is not considered in the solution of the tracking problem. As a consequence, in the presence of disturbances, model mismatch and uncertainties the quality of the resultant trajectory tracking may deteriorate dramatically. This makes the open-loop algorithm not suitable for a real-life scenarios. In order to "close the feedback loop", the well known Model Predictive Control (MPC) scheme [4, 11, 26, 27] can be applied. In general, MPC refers to a class of iterative control algorithms that make use of the explicit process model to predict the future response of a system over a certain prediction

✉ Mariusz Janiak
mariusz.janiak@pwr.edu.pl

Łukasz Chojnacki
lukasz.chojnacki@pwr.edu.pl

1 Department of Cybernetics and Robotics, Faculty of Electronics, Wrocław University of Science and Technology, Z. Janiszewskiego 11/17, 50-320 Wrocław, Poland

horizon. At each iteration, based on the most recent measurements, the MPC attempts to solve a finite horizon open-loop optimal control problem subject to system dynamics and constraints involving states and controls. The first part of the optimized control trajectory defined over control horizon is implemented at the real system. Then, the whole procedure is repeated with a prediction and a control horizons moved one step forward. A good introduction to theoretical and practical issues associated with the MPC method can be found in [5, 7, 10, 32, 33]. Recently, thanks to increasing computation capabilities and novel efficient algorithms [15, 29], the MPC is gaining popularity also in the field of robotics [9, 13, 14, 22]. The combination of the predictive scheme along with ECSA has been discussed in [25], where the general algorithm of finding inverse kinematics for mobile manipulators has been presented.

An integration of the MPC scheme with the open-loop trajectory tracking algorithm [17] is straightforward. It consists of replacing a certain optimal control problem by a constrained motion planning problem which by definition is formulated over a finite time horizon [17]. At each MPC iteration, the most currently observed system state serves as an initial value in the planning problem. The problem is solved with the use of the task priority Lifted Newton algorithm. All other steps remain unchanged. Unfortunately, this modification entails the need to review a theoretical aspects of a closed-loop algorithm stability and robustness. These issues will be a subject of a further publication. In order to verify a theoretical concept, the closed-loop algorithm will be tested by computer simulations. Nevertheless, we expect that the modified MPC will have similar properties as the standard moving horizon MPC.

The design of a robust stabilizing control laws for robotics systems is challenging mainly due to their complex nonlinear dynamics (eg. nonholonomic and underactuated systems), possible model uncertainties, disturbances, and physical constraints imposed on the system state and controls. Compared to the traditional techniques such as a Model-Based Control and an Adaptive Control, the Model Predictive Control can be considered as a general framework for a wide class of linear and nonlinear control systems, that offers robustness and good dynamic performance while ensuring operation within certain physical limits. Since the MPC tuning parameters are directly related to a certain cost function, it is relatively easy to achieve good performance, in contrast to the the traditional techniques for which the control laws are not intuitively obtained. The main drawback of MPC schemes is related to its computational complexity which limits application areas only to a sufficient slow or a simple dynamic systems.

To demonstrate differences between the optimization-based MPC and the presented approach, let's consider as an example a problem of finding solution of systems of nonlinear equations. This problem can be solved using a class of the gradient based methods such as ECSA, or can be formulated as an unconstrained optimization problem and solved by applying optimization methods [12, 28]. The ECSA is a Newton-type method, thus is local, assumes system of class $\mathcal{C}^1$, demonstrates a very fast local convergence rate, and a low computational complexity. Incorporation of constraints is intricate and involves an additional effort [19]. Optimization methods are global, but suffer from a local minima, assume system of class $\mathcal{C}^2$, demonstrate fast convergence rate at the expense of increased computational complexity. Incorporation of constraints of any type is straightforward. A more in-depth comparison of these two methods in the context of MPC, will be subject of further studies.

The organization of this paper is the following. Section 2 introduces the trajectory tracking problem formulation along with all necessary theoretical preliminaries. Section 3 presents derivation of the open-loop task priority Lifted Newton algorithm. The composition of the predictive scheme with open-loop trajectory tracking algorithm is discussed in Section 4. An application of the closed-loop predictive algorithm to selected trajectory tracking problems is demonstrated in Section 5. The paper concludes with Section 6.

2 Problem Formulation

We shall examine a nonlinear control system with output

$$\begin{cases} \dot{q} \triangleq f(q,u), \\ y \triangleq k(q) = \big(k_1(q), \ldots, k_r(q)\big)^T. \end{cases} \tag{1}$$

characterized by generalized coordinates $q \in \mathbb{R}^n$ and velocities $\dot{q} \in \mathbb{R}^n$, where $f(q,u)$ and $k(q)$ are continuously differentiable, $y \in \mathbb{R}^r$ denotes a vector of task space coordinates, $u \in \mathbb{R}^m$ denotes a control vector, and $m \leq n$. Admissible control functions $u(\cdot)$ are chosen Lebesgue square integrable, $u(\cdot) \in \mathcal{U} \subset L^2_m[t_b, t_e]$, on a time interval $T \triangleq [t_b, t_e]$, where $t_e > t_b$. It will be assumed that the state trajectory $q(t) \triangleq \varphi_{q_0,t}\big(u(\cdot)\big)$ exists for every initial state $q_0 = q(0)$, and every control $u(\cdot) \in \mathcal{U}$.

The trajectory tracking problem consists in finding a control function $u(\cdot)$ driving the system (1) over a prescribed time interval $[0, T_t]$, such that the system output trajectory $y(t)$ is as close as possible to a given demanded task space trajectory $y_d(t) \in \mathbb{R}^r$, thus

$$\int_0^{T_t} \big(y(t) - y_d(t)\big)^T \big(y(t) - y_d(t)\big) dt \leq \epsilon, \tag{2}$$

where ϵ is a small positive number.

The formula (2) specifies a system trajectory, and can be regarded as a constraint imposed on the system behavior. Thus, the trajectory tracking problem can be defined as a constrained motion planning problem. To solve such a problem the idea presented in [19] can be adopted. This approach consists in replacing the constrained problem by an unconstrained one addressed in an extended control system representation. The constraint (2) can be included into the system (1) by adding a set of extra state variables $x \in \mathbb{R}^r$,

$$\begin{cases} \dot{q} = f(q,u), \quad y = k(q), \\ \dot{x} \triangleq g(q), \end{cases} \tag{3}$$

where

$$g(q) \triangleq \begin{pmatrix} (y_1 - y_{d1})^2 \\ \vdots \\ (y_r - y_{dr})^2 \end{pmatrix} = \begin{pmatrix} \big(k_1(q) - y_{d1}\big)^2 \\ \vdots \\ \big(k_r(q) - y_{dr}\big)^2 \end{pmatrix}. \tag{4}$$

Theoretically, the proposed form of vectorial-constraints (4) can introduce unnecessary singularities at locations in the search space $\mathcal{U}$, for which elements of the system task space trajectory $y_i(t)$ fully coincide with equivalent elements of the desired trajectory $y_{di}(t)$, resulting in $\int_0^{T_t} g_i\big(q(t)\big)dt = 0, i = 1, \ldots, r$. This issue can be solved in two ways, either by including to $g(q)$ only active constraints [8] or by the Jacobian regularization [19]. Practically, mainly because of control discretization, numerical integration and errors, such a situation is very rare.

Now, assuming that $x_0 \triangleq x(t_0) = 0$, the constrained motion planning problem becomes equivalent to the unconstrained problem of finding a control function $u(\cdot)$ in the extended system (3), such that

$$||E_{q_0,x_0,T_t}\big(u(\cdot)\big)|| \le \epsilon \tag{5}$$

where $E_{q_0,x_0,T_t}\big(u(\cdot)\big)$ is the end-point map of the extended system (3)

$$E_{q_0,x_0,T_t}\big(u(\cdot)\big) \triangleq x(T_t) = \psi_{x_0,T_t}\big(u(\cdot)\big),$$

and $x(t) \triangleq \psi_{x_0,t}\big(u(\cdot)\big)$ represents the flow of the extra state equation.

3 Open-Loop Algorithm

Following the idea presented in [17], the motion planning problem (5) will be solved using a task priority version of the Lifted Newton Method [1]. This method involves a number of intermediate variables that correspond to the original system's states $s^i \triangleq q(t_i), i = 1 \ldots N$, defined at time grid spanned over a given time horizon $T = [t_b, t_e]$

$$t_b = t_0 < \ldots < t_i < t_{i+1} < \ldots < t_N = t_e. \tag{6}$$

In order to simplify an algorithm implementation, we assume fixed time intervals $t_{i+1} - t_i = \tau$, $\tau = T/N$, but this assumption is not mandatory. Further on we will use the superscript to denote the interval to which a marked element belongs. Intermediate states can be grouped into a vector $s \triangleq (s^1, \ldots, s^N) \in \mathbb{R}^{nN}$. Moreover, for a further notational simplicity, and only for this purpose, we define $s^0 \triangleq q(t_0)$ that refers to the original system's initial state, and should not be regarded as intermediate state. On the time grid (6) we perform a piecewise continuous control discretization

$$u^i(t) \triangleq \chi_{[t_{i-1},t_i)}(t) P^i(t) \lambda^i,$$

$$\chi_{[t_b,t_e)}(t) \triangleq \begin{cases} 1, & t_b \le t < t_e \\ 0, & \text{otherwise,} \end{cases}$$

where $P^i(t) \triangleq \text{block diag}\{P_1^i(t), \ldots, P_m^i(t)\}$ is a block diagonal matrix comprising certain basic orthogonal functions $P_k^i(t)$ in the Hilbert control space $\mathbb{L}_m^2[t_{i-1}, t_i]$,

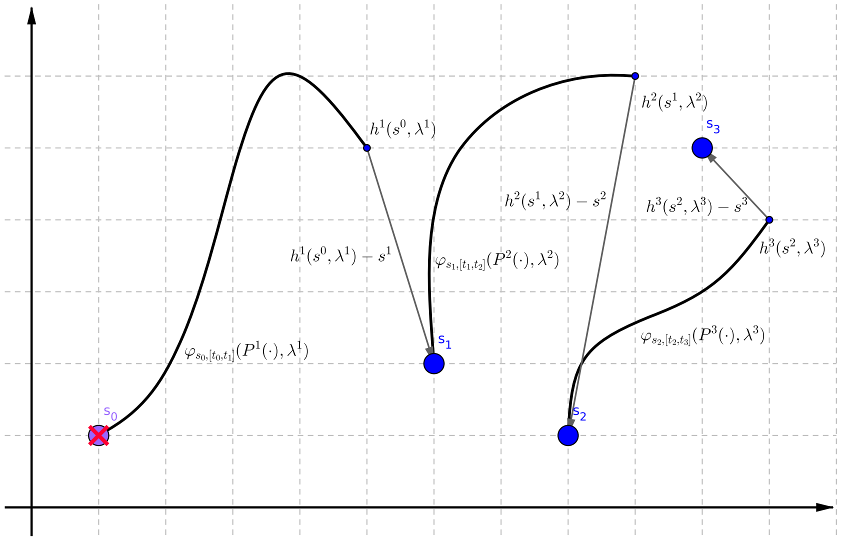
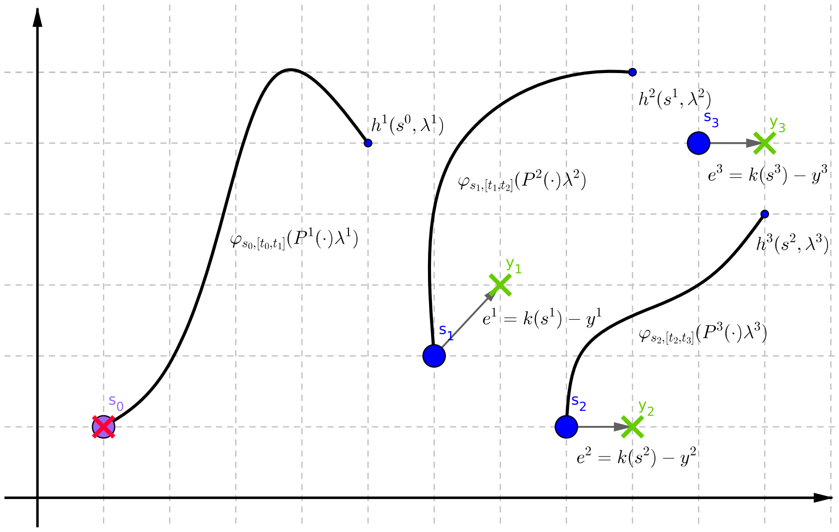

$h^1(s^0,\lambda^1)$
$h^2(s^1,\lambda^2)$
s_3
$h^2(s^1,\lambda^2) - s^2$
$h^3(s^2,\lambda^3) - s^3$
$h^1(s^0,\lambda^1) - s^1$
$\varphi_{s_1,[t_1,t_2]}(P^2(\cdot),\lambda^2)$
$h^3(s^2,\lambda^3)$
s_1
$\varphi_{s_0,[t_0,t_1]}(P^1(\cdot),\lambda^1)$
$\varphi_{s_2,[t_2,t_3]}(P^3(\cdot),\lambda^3)$
s_0
s_2

Fig. 1 Discontinuity issue of the system flow

Fig. 2 Task space error



$\lambda^i \triangleq (\lambda^i_{1,1}, \ldots \lambda^i_{1,M}, \ldots, \lambda^i_{m,1}, \ldots \lambda^i_{m,M})^T \in \mathbb{R}^{mM}$ denotes a collection of control parameters, and M is the length of each control series chosen in such a way that $Mm \geq n$. As a result of discretization, the control space is represented by $\mathbb{R}^{mMN}$, thus $u(t)$ depends only on a finite number of control parameters $\lambda \triangleq (\lambda^1, \ldots, \lambda^N)^T$.

The flow of the original system (1) is divided into a number of subsequent flows,

$$h^i(s^{i-1}, \lambda^i) \triangleq \varphi_{s^{i-1},\tau_i}(\lambda^i), \tag{7}$$

calculated separately on each interval $[t_{i-1}, t_i]$, $\tau_i \in [t_{i-1}, t_i]$. This breaks the system flow and make it discontinuous – see Fig. 1. This is an inherent nature of all “multiple shooting” techniques including the Lifted Newton Method. The flow of the system preserves continuity, when the following relation holds

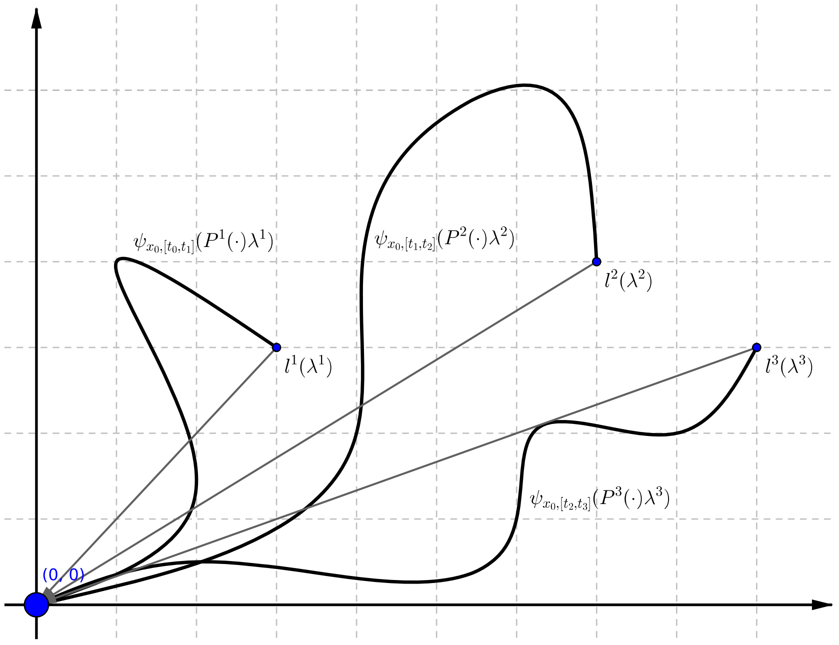
$$H_{s^0,T}(s, \lambda) - s = 0, \tag{8}$$

where

$$H_{s^0,T}(s,\lambda) \triangleq \begin{pmatrix} h^1(s^0,\lambda^1) \\ h^2(s^1,\lambda^2) \\ \vdots \\ h^N(s^{N-1},\lambda^N) \end{pmatrix},$$

is a function that describes the evolution of states of the original system due to the applied controls. Each

Fig. 3 Constraints violation function


$\psi_{x_0,[t_0,t_1]}(P^1(\cdot)\lambda^1)$
$\psi_{x_0,[t_1,t_2]}(P^2(\cdot)\lambda^2)$
$l^2(\lambda^2)$
$l^1(\lambda^1)$
$l^3(\lambda^3)$
$\psi_{x_0,[t_2,t_3]}(P^3(\cdot)\lambda^3)$
(0, 0)

intermediate state is linked with the point in task space by the associated error function

$$e(s) \triangleq \begin{pmatrix} e^1(s^1) \\ e^2(s^2) \\ \vdots \\ e^N(s^N) \end{pmatrix} = \begin{pmatrix} k(s^1) - y_d(t_1) \\ k(s^2) - y_d(t_2) \\ \vdots \\ k(s^N) - y_d(t_N) \end{pmatrix}. \qquad (9)$$

These functions define the distance of the system's output to selected waypoints defined along the desired trajectory – see Fig. 2.

To proceed further, similarly to Eq. 7, we divide the flow of the remaining part of the extended system (3) into a sequence of independent sub-flows

$$l^i(\lambda^i) \triangleq \psi_{x_0,\tau_i}(\lambda^i),$$

calculated separately on each time interval $[t_{i-1}, t_i]$, $\tau_i \in [t_{i-1}, t_i]$, where $i = 1, \dots, N$. Again, we can group these sub-flows, and define the function that describes the evolution of the extra state variables due to the applied controls

$$G_{0,T}(\lambda) \triangleq \begin{pmatrix} l^1(\lambda^1) \\ l^2(\lambda^2) \\ \vdots \\ l^N(\lambda^N) \end{pmatrix}, \qquad (10)$$

where $G_{0,T} : \mathbb{R}^{mMN} \longrightarrow \mathbb{R}^{rN}$. This function reflects the constraints violation (4) at the end of each interval, as presented in Fig. 3.

Any constraints violation $g(q) \neq 0$ drives the evolution of the flow $G_{0,T}(\lambda)$ moving away extra state variables from 0.

The trajectory tracking problem consists in determining the control parameters λ and the intermediate states s so that the system's flow preserves continuity (8), task-space error (9) vanishes

$$F_{s^0,T}(s,\lambda) \triangleq \begin{pmatrix} H_{s^0,T}(s,\lambda) - s \\ e(s) \end{pmatrix} = 0, \qquad (11)$$

$F_{s^0,T} : \mathbb{R}^{nN} \times \mathbb{R}^{mMN} \longrightarrow \mathbb{R}^{nN+rN}$, and constraints are not violated

$$G_{0,T}(\lambda) = 0. \qquad (12)$$

The original trajectory tracking problem has been transformed into two conjugated, finite dimensional, nonlinear root finding problems. The former, the primary task (11) reflects the motion through waypoints problem and reaching the terminal point, the later, the secondary task (12) maintains the trajectory tracking constraints. Task separation along with task priorities should ensure continuity of the system's flow even when the constraints are violated.

To solve these two conjugated problems (11) and (12) with distinct priorities the combination of the Newton method and the task priority approach [31] will be applied. A conventional Newton algorithm

$$\begin{pmatrix} s(\kappa+1) \\ \lambda(\kappa+1) \end{pmatrix} = \begin{pmatrix} s(\kappa) \\ \lambda(\kappa) \end{pmatrix} + \begin{pmatrix} \gamma_s \Delta s(\kappa) \\ \gamma_\lambda \Delta\lambda(\kappa) \end{pmatrix}, \qquad (13)$$

iterates from an initial guess $\big(s(0), \lambda(0)\big)$ until either $\kappa > \kappa_{max}$ or a sum $||F_{s^0,T}(s,\lambda)|| + ||G_{0,T}(\lambda)|| \leq TOL(\epsilon)$, where κ_{max} defines the maximum number of algorithm's iterations, $TOL(\epsilon)$ is a total error tolerance, γ_s and γ_λ are damping factors, and $(\Delta s, \Delta\lambda)$ represents the Newton step. The algorithm's convergence can be improved when the dumping factors γ_s and γ_λ are calculated in each iteration basing on formulas presented in [3]

$$\gamma_s(\kappa) \triangleq \min\left\{\sqrt{\frac{2\gamma}{||\Delta s(\kappa)||_2}}, 1\right\}, \quad \gamma_\lambda(\kappa) \triangleq \min\left\{\sqrt{\frac{2\gamma}{||\Delta\lambda(\kappa)||_2}}, 1\right\},$$

where the tolerance $\gamma > 0$ is a design parameter of the algorithm. According to the task priority approach [31] calculation of the Newton step length involves using a Jacobian inverse along with the projection onto the Jacobian kernel

$$\begin{pmatrix} \Delta s(\kappa) \\ \Delta\lambda(\kappa) \end{pmatrix} \triangleq -J^{F}_{s^0,T}{}^{\#}\big(s(\kappa), \lambda(\kappa)\big) F_{s^0,T}\big(s(\kappa), \lambda(\kappa)\big)$$
$$- X_{s^0,T}\big(s(\kappa), \lambda(\kappa)\big) J^{G}_{0,T}{}^{\#}\big(\lambda(\kappa)\big) G_{0,T}\big(\lambda(\kappa)\big), \qquad (14)$$

where $J^F_{s^0,T}(s,\lambda) \triangleq \frac{\partial F_{s^0,T}}{\partial(s,\lambda)}(s,\lambda)$ and $J^G_{0,T}(\lambda) \triangleq \frac{\partial G_{0,T}}{\partial\lambda}(\lambda)$ represent the primary and the secondary task Jacobian, respectively, the symbol $[\cdot]^{\#}$ denotes a right Jacobian inverse, and $X_{s^0,T}(s,\lambda)$ is the projection onto $\ker J^F_{s^0,T}(s,\lambda)$, so

$$X_{s^0,T}(s,\lambda) \triangleq \mathbb{I}_{N(n+mM)} - J^F_{s^0,T}(s,\lambda)^{\#} J^F_{s^0,T}(s,\lambda),$$

and $\mathbb{I}$ stands for the unit matrix. In accordance with [17], the part of the system (14) referring to the primary task can be decomposed into the following system of linear equations

$$H_{s^0,T}(s,\lambda) - s + \left(\frac{\partial H_{s^0,T}(s,\lambda)}{\partial s} - \mathbb{I}_{n \cdot N}\right)\Delta s + \frac{\partial H_{s^0,T}(s,\lambda)}{\partial \lambda}\Delta\lambda = 0,$$

$$e(s) + \frac{\partial e(s)}{\partial s}\Delta s = 0,$$

which after the transformation take the form

$$\Delta s = \underbrace{-\mathbf{M}^{-1}\big(H_{s^0,T}(s,\lambda) - s\big)}_{\mathbf{z}} \underbrace{-\mathbf{M}^{-1}\frac{\partial H_{s^0,T}(s,\lambda)}{\partial \lambda}}_{\mathbf{Z}}\Delta\lambda,$$

$$0 = \underbrace{e(s) + \frac{\partial e(s)}{\partial s}\mathbf{z}}_{\mathbf{w}} + \underbrace{\frac{\partial e(s)}{\partial s}\mathbf{Z}}_{\mathbf{W}}\Delta\lambda. \qquad (15)$$

where the index κ has been dropped for notational convenience, and the matrix $\mathbf{M} \triangleq \left(\frac{\partial H_{s^0,T}(s,\lambda)}{\partial s} - \mathbb{I}_{nN}\right)$ is lower triangular, square and invertible [1]. Because the

Fig. 4 Task priority Lifted Newton trajectory tracking algorithm

```
Input  : $q_0$, $y_d(t)$, $t_b$, $t_e$, $s_{init}$, $\lambda_{init}$
Output : $\lambda^*$, $s^*$
Require: $N$, $M$, $P(t)$, $\gamma$, $\kappa_{max}$, $TOL$
begin
    $(t_0, \dots, t_N) = (t_b : \frac{(t_e - t_b)}{N} : t_e)$;
    $\kappa = 0$;
    $s(\kappa) = s_{init}$; $\lambda(\kappa) = \lambda_{init}$;            // Set intermediate states and control
    while $(\kappa < \kappa_{max}) \wedge (||F_{s^0,T}(s(\kappa), \lambda(\kappa))|| + ||G_{0,T}(\lambda)|| \geq TOL)$ do
        // Calculate $H_{s^0,T}(s, \lambda)$, $G_{0,T}(\lambda)$, $e(s)$ and respective derivatives
        $s^0(\kappa) = q_0$;
        for $i = 1$ to $N$ do                                     // or use parfor instead
            $t \in [t_{i-1}, t_i]$;
            $u = P^i(t)\lambda^i(\kappa)$;
            $q(0) = s^{i-1}(\kappa)$;
            $x(0) = 0$; $\Xi(0) = 0_{n \times mM}$; $\Upsilon(0) = 0_{r \times mM}$;
            if $i > 1$ then
                $\Gamma(0) = \mathbb{I}_{n \times n}$;
                $\begin{pmatrix} h^i \\ l^i \\ \frac{\partial h^i}{\partial s^{i-1}} \\ \frac{\partial h^i}{\partial \lambda^i} \\ \frac{\partial l^i}{\partial \lambda^i} \end{pmatrix} \leftarrow$ solve( $\begin{cases} \dot{q} = f(q, u) \\ \dot{x} = g(q, u) \\ \dot{\Gamma} = A_f(t)\Gamma \\ \dot{\Xi} = A_f(t)\Xi + B_f(t)P^i(t) \\ \dot{\Upsilon} = A_g(t)\Xi \end{cases}$ , $\begin{pmatrix} q(0) \\ x(0) \\ \Gamma(0) \\ \Xi(0) \\ \Upsilon(0) \end{pmatrix}$, $t$);
            else
                $\begin{pmatrix} h^i \\ l^i \\ \frac{\partial h^i}{\partial \lambda^i} \\ \frac{\partial l^i}{\partial \lambda^i} \end{pmatrix} \leftarrow$ solve( $\begin{cases} \dot{q} = f(q, u) \\ \dot{x} = g(q, u) \\ \dot{\Xi} = A_f(t)\Xi + B_f(t)P^i(t) \\ \dot{\Upsilon} = A_g(t)\Xi \end{cases}$ , $\begin{pmatrix} q(0) \\ x(0) \\ \Xi(0) \\ \Upsilon(0) \end{pmatrix}$, $t$);
            end
            $e^i(s^i(\kappa)) = k(s^i(\kappa)) - y_d(t_i)$;
            $\frac{\partial e^i(s^i(\kappa))}{\partial s^i} = \frac{\partial k(s^i(\kappa))}{\partial s^i}$;
        end
        $\mathbf{M} = \frac{\partial H_{s^0,T}(s(\kappa), \lambda(\kappa))}{\partial s} - \mathbb{I}_{n \cdot N}$;
        // Calculate z, Z, w, W, X, and v
        $\mathbf{z} = -\mathbf{M}^{-1}\left(H_{s^0,T}(s(\kappa), \lambda(\kappa)) - s(\kappa)\right)$;
        $\mathbf{Z} = -\mathbf{M}^{-1}\frac{\partial H_{s^0,T}(s(\kappa), \lambda(\kappa))}{\partial \lambda}$;
        $\mathbf{w} = e(s(\kappa)) + \frac{\partial e(s(\kappa))}{\partial s}\mathbf{z}$;
        $\mathbf{W} = \frac{\partial e(s(\kappa))}{\partial s}\mathbf{Z}$;
        $\mathbf{X} = \mathbb{I}_{nMN} - W^{\#P}W$;
        $\mathbf{v} = J^G_{0,T}(\lambda(\kappa))^{\#P}G_{0,T}(\lambda(\kappa))$;
        // Calculate Newton step length
        $\Delta\lambda(\kappa) = -\mathbf{W}^{\#}\mathbf{w} - \mathbf{X}\mathbf{v}$;
        $\Delta s(\kappa) = \mathbf{z} + \mathbf{Z}\Delta\lambda(\kappa)$;
        // Calculate damping factors
        $\gamma_\lambda(\kappa) = \min(\sqrt{\frac{2\gamma}{||\Delta\lambda(\kappa)||_2}}, 1)$;
        $\gamma_s(\kappa) = \min(\sqrt{\frac{2\gamma}{||\Delta s(\kappa)||_2}}, 1)$;
        // Perform Newton step
        $\lambda(\kappa + 1) = \lambda(\kappa) + \gamma_\lambda(\kappa)\Delta\lambda(\kappa)$;
        $s(\kappa + 1) = s(\kappa) + \gamma_s(\kappa)\Delta s(\kappa)$;
        // Next iteration
        $\kappa = \kappa + 1$;
    end
    // Set solution
    $\lambda^* = \lambda(\kappa)$;
    $s^* = s(\kappa)$;
end
```

matrix $\mathbf{W}_{r\times mMN}$ is not square, a general solution of Eq. 15 involves the Jacobian inverse with projection [34]

$$\Delta\lambda = -\mathbf{W}^{\#P}\mathbf{w} + \mathbf{X}\zeta, \tag{16}$$

where $\mathbf{W}^{\#P} \triangleq \mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}$ stands for the Moore-Penrose pseudoinverse, $\mathbf{X} \triangleq (\mathbb{I}_{mMN} - \mathbf{W}^{\#P}\mathbf{W})$ denotes the projection onto ker $\mathbf{W}$, and $\zeta \in \mathbb{R}^{mMN}$ is any element in the parametrized control space. This element will be determined by the secondary task (12), with the use of the Jacobian inverse

$$\Delta\lambda = -\underbrace{J^{G\ \#P}_{0,T}(\lambda)G_{0,T}(\lambda)}_{\mathbf{v}}.$$

For the secondary task, Eq. 16 needs to be satisfied only within the kernel of $\mathbf{W}^{\#P}$, so by projecting (16) onto ker $\mathbf{W}$, we end up with

$$\mathbf{X}\zeta = -\mathbf{X}\mathbf{v},$$

assuming that by definition the projection is idempotent $\mathbf{X}\mathbf{X} = \mathbf{X}$, and $\mathbf{X}\mathbf{W}^{\#P} = 0$. Finally, the formula for calculating the Newton step, including two tasks with different priorities has the following form

$$\begin{cases} \Delta\lambda = -\mathbf{W}^{\#P}\mathbf{w} - \mathbf{X}\mathbf{v} \\ \Delta s = \mathbf{z} + \mathbf{Z}\Delta\lambda \end{cases}. \tag{17}$$

For more technical details on the structure of the matrices $\frac{\partial H_{s0,T}(s,\lambda)}{\partial s}$, $\frac{\partial H_{s0,T}(s,\lambda)}{\partial \lambda}$, $\frac{\partial G_{0,T}(\lambda)}{\partial \lambda}$ and $\frac{\partial e(s)}{\partial s}$, please refer to [17]. Beside $\frac{\partial e(s)}{\partial s}$, which can be calculated analytically, the derivatives of the remaining matrices have to be calculated numerically, using e.g. the forward sensitivity analysis method [2]. To this aim, two additional systems of differential equations

$$\dot{\Gamma}^i \triangleq A_f(t)\Gamma^i, \quad \Gamma^i(0) \triangleq \mathbb{I}_n, \tag{18}$$

and

$$\begin{cases} \dot{\Xi}^i \triangleq A_f(t)\Xi^i + B_f(t)P^i(t), & \Xi^i(0) \triangleq 0_{n\times mM}, \\ \dot{\Upsilon}^i \triangleq A_g(t)\Xi^i, & \Upsilon^i(0) \triangleq 0_{r\times mM}, \end{cases} \tag{19}$$

need to be integrated along with Eq. 3 over each time interval, where $i = 1,\ldots N$. Matrices in Eqs. 18 and 19 come from the linear approximation of Eq. 3 along the control-trajectory pair $\big(u(t), q(t)\big)$, so

$$A_f(t) \triangleq \frac{\partial f\big(q(t), u(t)\big)}{\partial q}, B_f(t) \triangleq \frac{\partial f\big(q(t), u(t)\big)}{\partial u},$$
$$A_g(t) \triangleq \frac{\partial g\big(q(t)\big)}{\partial q}.$$

The sensitivity of the system (3) to initial conditions is calculated as

$$\frac{\partial h^i(s^{i-1}, \lambda^i)}{\partial s^{i-1}} \triangleq \Gamma(t_i),$$

while the sensitivity to control parameters

$$\frac{\partial h^i(s^{i-1}, \lambda^i)}{\partial \lambda^i} \triangleq \Xi(t_i), \qquad \frac{\partial l^i(\lambda^i)}{\partial \lambda^i} \triangleq \Upsilon(t_i).$$

The algorithm presented in Fig. 4 summarizes the described Lifted Newton procedure with task priorities.

The most time-consuming part of the algorithm is related to solving at each algorithm's step, a set of ordinary differential Eqs. 3, 18, and 19. Fortunately, these equations are not dependent across the time intervals, thus can be solved concurrently. In computation it is recommended to set initial values of the intermediate state s^i so that

$$y_d(t_i) = k(s^i), \quad i = 1,\ldots,N. \tag{20}$$

Typically, the initial values of the control parameters λ^0 should be chosen sufficiently small, not too far from the solution of the problem (5).

4 Closing the Loop

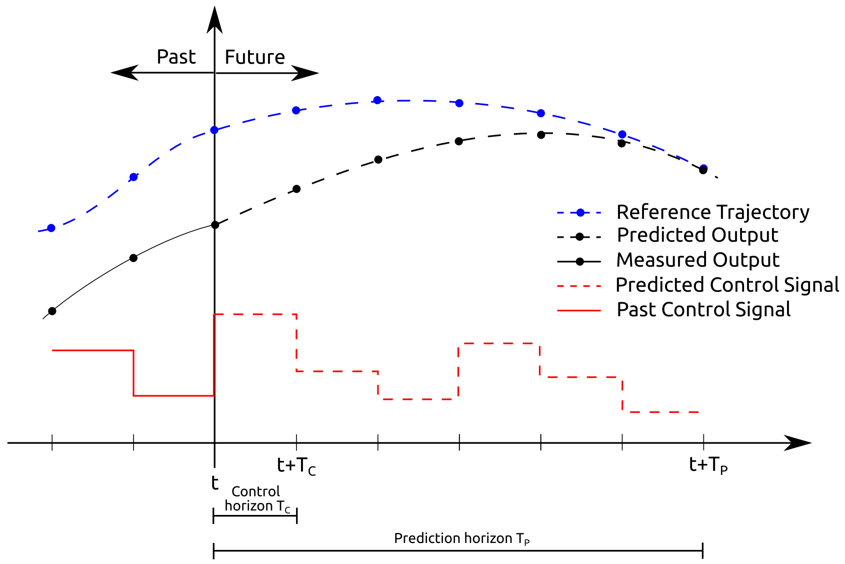
By design, the algorithm derived in the previous section is an open-loop trajectory tracking algorithm. It solves the tracking problem on entire time horizon without any feedback from the system. In the presence of any internal or external disturbances the final tracking quality may by significantly reduced, what makes the open-loop algorithm unsuitable for practical applications. In order to overcome these difficulties, the model predictive control method will be adopted. Originally, it is an iterative, a feedback control technique based on the online solution of a finite horizon open-loop optimal control problem subject to system dynamics and constraints. The basic principle of the predictive scheme is illustrated in Fig. 5,

it may be summarized as follows. At each control step, the most currently observed system state serves as the initial value in the optimization problem. The problem is solved on a predefined interval called the prediction horizon T_p. Only a part of the resulting solution is applied to the system over a chosen one-step control horizon T_c. In the next step, the whole procedure is repeated with the one-step control and prediction horizons moving forward and new measured initial state.

In the proposed approach, instead of solving a certain optimization problem, the root finding problem (5) will be solved using the open-loop trajectory tracking algorithm presented in Section 3. All other steps remain unchanged. The general procedure of the closed-loop predictive control algorithm equipped with the Lifted Newton method is summarized in Fig. 6.

The algorithm iterates over an interval $[0, T_t]$ on which the desired trajectory $y_d(t)$ is defined. An acquisition process of the most recent system state q_0 is represented

Fig. 5 Principle of the predictive scheme



by the `getMeasure()` function (line 5). Typically, such a function returns a state estimate with a noise, not a real state of the system. In the next step, the Lifted Newton algorithm solves a finite time trajectory tracking problem defined on the prediction horizon T_p (line 6). Using the resulting control parameters λ, only the first part covering the control horizon T_c of the control signals $u(\cdot)$ is calculated and applied to the original system (line 7). Finally, the algorithm repeats the whole procedure moving the time horizon one step forward (line 8), and using the new values of λ and s. In case of the piecewise constant control discretication, a control parameters vector λ may by updated in each iteration, in such a way to discard parameters associated with first interval T_c, switch others λ one interval T_c left, and fill the last missing interval with values taken from a previous one.

The procedure presented in Fig. 6 does not include any optimizations which are required for a real-time application [6]. Mainly, it turns out that the computations of the new values of the control signals can largely be prepared without knowledge of the value of actual system state. Thus assumption can be made that the approximation of the feedback control is instantly available at the time that actual state is known. However, after feedback has been delivered, the full computing time is needed to prepare the next iteration. The procedure presented in the Fig. 6 will be used only to verify a theoretical idea. The performance of the closed-loop algorithm will be tested with computer simulations. The formal proof of the algorithm stability and robustness will be a subject of a further research.

5 Computer Simulations

In this section we shall present results of a series of computer simulations that have been performed in order to verify the performance of the closed-loop predictive trajectory tracking algorithm based on the task priority Lifted Newton method. For comparison, along with the presented algorithm we have tested a traditional model

Fig. 6 Closed-loop tracking algorithm

```
   Data: $y_d(t)$, $T_t$, $T_p$, $T_c$
1  begin
2      $\tau = 0$;
3      Set initial $s$ and $\lambda$;
4      while $\tau \leq T_t$ do
           // Get the actual system state (include $q_{\text{noise}}$)
5          $q_0 \leftarrow$ getMeasure();
           // Solve tracking problem on the prediction horizon $T_p$
6          $\begin{pmatrix}\lambda \\ s\end{pmatrix} \leftarrow$ liftedNewton($q_0$, $y_d(t)$, $\tau$, $\tau + T_p$, $s$, $\lambda$);
           // Apply $u(\cdot)$ over a one-step control horizon $T_c$
7          $u = P([\tau, \tau + T_c])\lambda$;
           // Move horizon a one step forward
8          $\tau = \tau + T_c$;
9      end
10 end
```

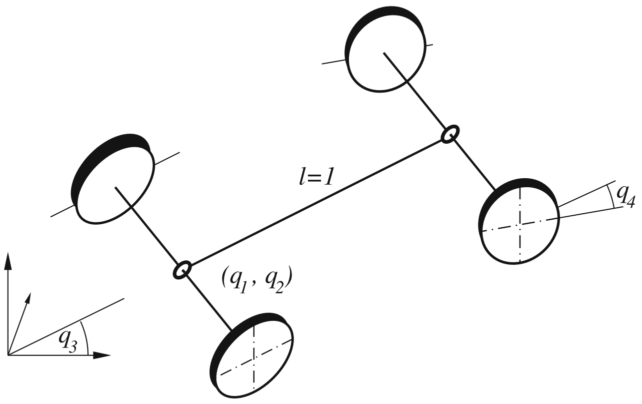


Fig. 7 Kinematic car

predictive controller provided by the ACADO toolkit [15]. Both algorithms have been applied to several trajectory tracking problems of the kinematic car type platform. The kinematic car is shown in Fig. 7,

its motion can be described by the following control system reflecting the exclusion of the side-slip of the platform

$$\begin{cases} \dot{q}_1 = u_1 \cos q_3 \cos q_4, \quad \dot{q}_2 = u_1 \sin q_3 \cos q_4, \\ \dot{q}_3 = u_1 \sin q_4, \quad \dot{q}_4 = u_2, \\ y = k(q) = (q_1, q_2, q_3), \end{cases} \tag{21}$$

where (q_1, q_2) denote the robot position in the XY plane, q_3 is the platform orientation, and q_4 represents the front wheels direction angle. The output function selects the part of the state that reflects the platform position and orientation on the plane, therefore the discussed problems will be defined with respect to these platform coordinates. The desired trajectory is defined over the horizon $T_t = 2\pi$s as a Lissajous curve with ratio 0.5

$$\begin{cases} y_{d1} = 5\sin(t + \frac{\pi}{2}), \\ y_{d2} = 5\sin(2t), \\ y_{d3} = \arctan_2\big(10\cos(2t), 5\cos(t + \frac{\pi}{2})\big) + r(t), \end{cases}$$

where $r(t)$ represents a regularization function that makes orientation component continous

$$r(x) = \begin{cases} 2\pi & \frac{\pi}{4} \leq x \pmod{2\pi} < \frac{3\pi}{4}, \\ 0 & \text{otherwise.} \end{cases}$$

The platform starts from the origin of the coordinate system $q_0 = (0, 0, 0, 0)$.

Control signals of the ECSA algorithm have the form of piecewise constant signals, with initial value of control parameters set to $\lambda^i_{1,j} = 2$, and $\lambda^i_{2,j} = 0$, $i = 1, \ldots, N$, $j = 1, \ldots, M$, resulting in the platform going straight-forward. The control interval is set to $T_c = 0.01$s. The initial value of the intermediate states have been chosen to meet the condition (20), with not bounded elements set to zero. Parameters of the ECSA algorithm are picked up as $\gamma = 0.7$, and the tolerance error $TOL = 10^{-3}$. The algorithm has been implemented in Matlab R2014a 64-bit, and tested on the PC with Intel Core i7-2600K 3.40GHz and the 8GB RAM platform hosted by the Ubuntu Linux 16.04LTS 64-bit. Results of computations for various values

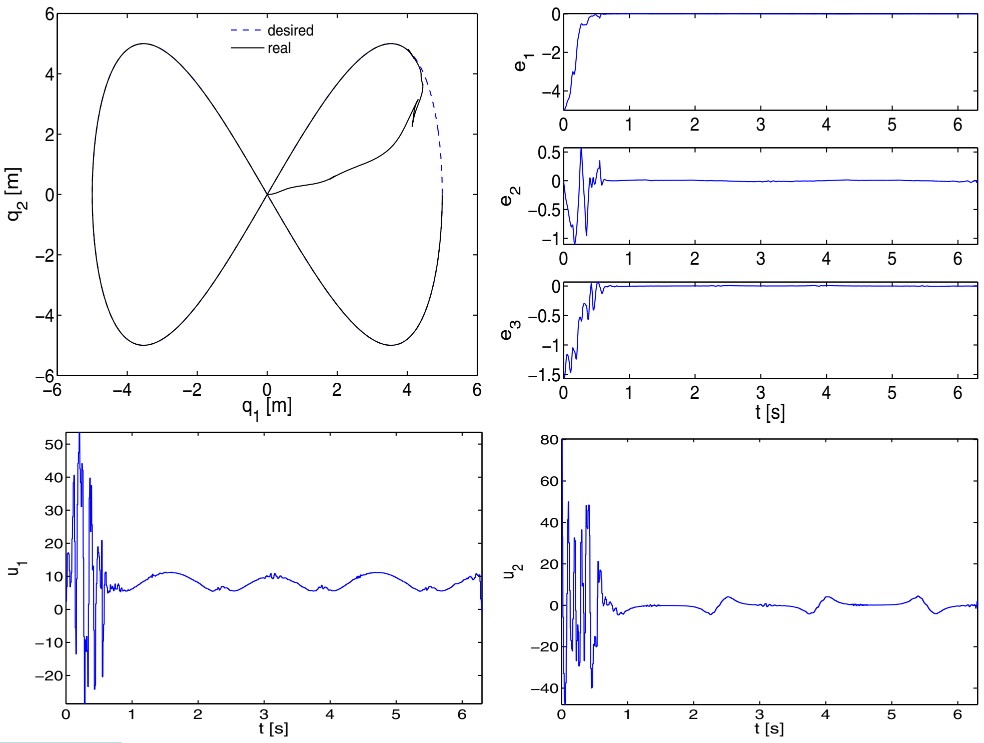


Fig. 8 ECSA 1: $T_p = 0.5$, $N = 5$, $M = 10$, $\kappa_{max} = 1$, without noise

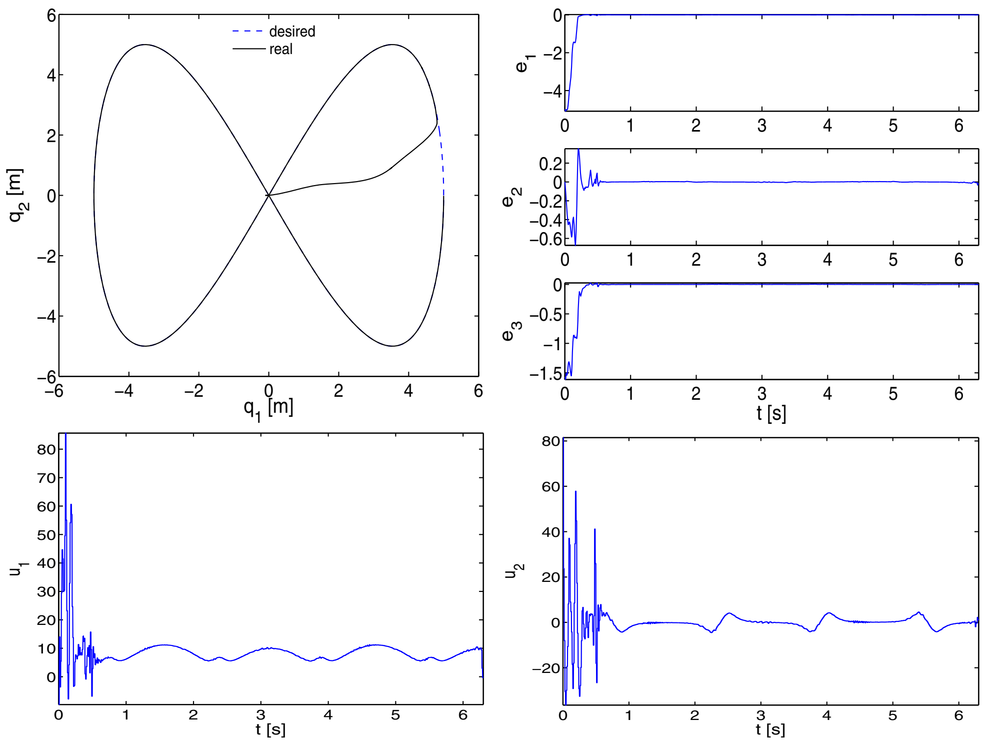


Fig. 9 ECSA 2: $T_p = 0.5$, $N = 5$, $M = 10$, $\kappa_{max} = 2$, without noise

of parameters T_p, N, M, κ_{max} selected in such a way that the condition $T_c = \frac{T_p}{NM}$ holds, excluding measurement noise, are displayed in Figs. 8, 9, 10, 11, 12 and 13.

Simulation results in which a uniform random noise q_{noise} has been added to the state feedback signal are presented in Figs. 14 and 15.

The order of the plots is the following. The first plot presents the resulting platform paths corresponding to the solution of the tracking problem. The continuous line represents the real path of the platform, the dashed line represents the desired trajectory. The neighbor plot, illustrates the tracking error components: the platform

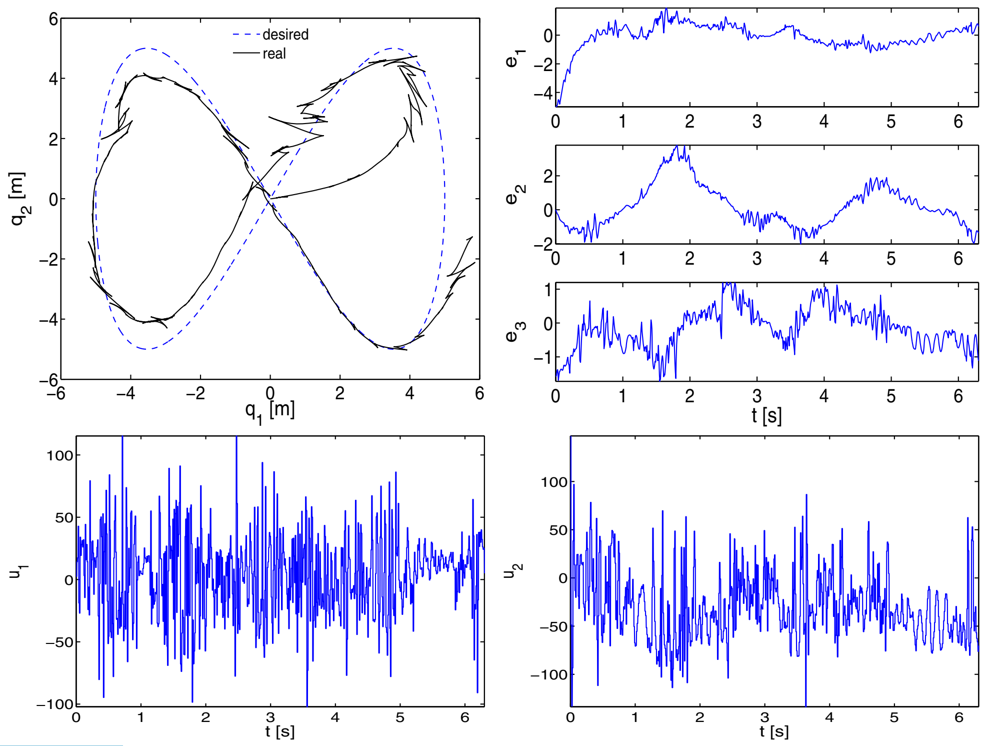
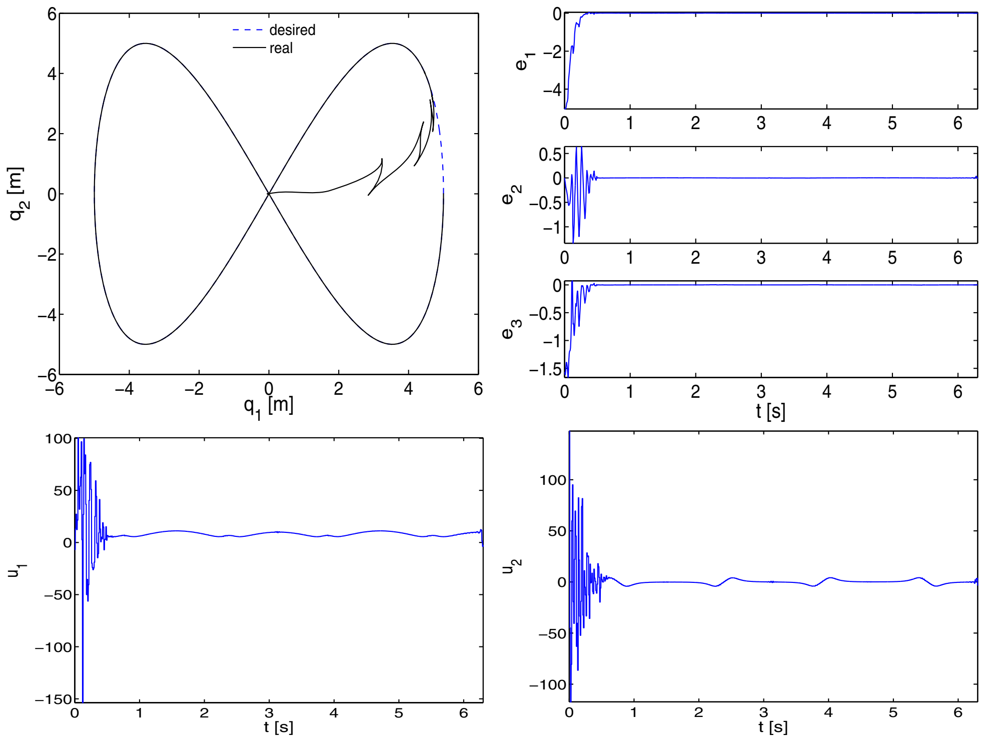


Fig. 10 ECSA 3: $T_p = 0.5$, $N = 10$, $M = 5$, $\kappa_{max} = 1$, without noise

Fig. 11 ECSA 4: $T_p = 0.5$, $N = 10$, $M = 5$, $\kappa_{max} = 2$, without noise



position and orientation in the XY plain. Two plots in the bottom row show a control signals u_1 and u_2, respectively, that were generated by the closed-loop tracking algorithm. The plots in Figs. 16, 17, 18 and 19 illustrate the algorithm's convergence. The continuous line represents the primary task error, the dashed line represents the secondary task error which corresponds to the constraints violation.

Obtained results have shown that the derived predictive algorithm is able to track a trajectory in task space very efficiently. This is manifested by a low tracking error, as

Fig. 12 ECSA 5: $T_p = 1.0$, $N = 10$, $M = 10$, $\kappa_{max} = 1$, without noise

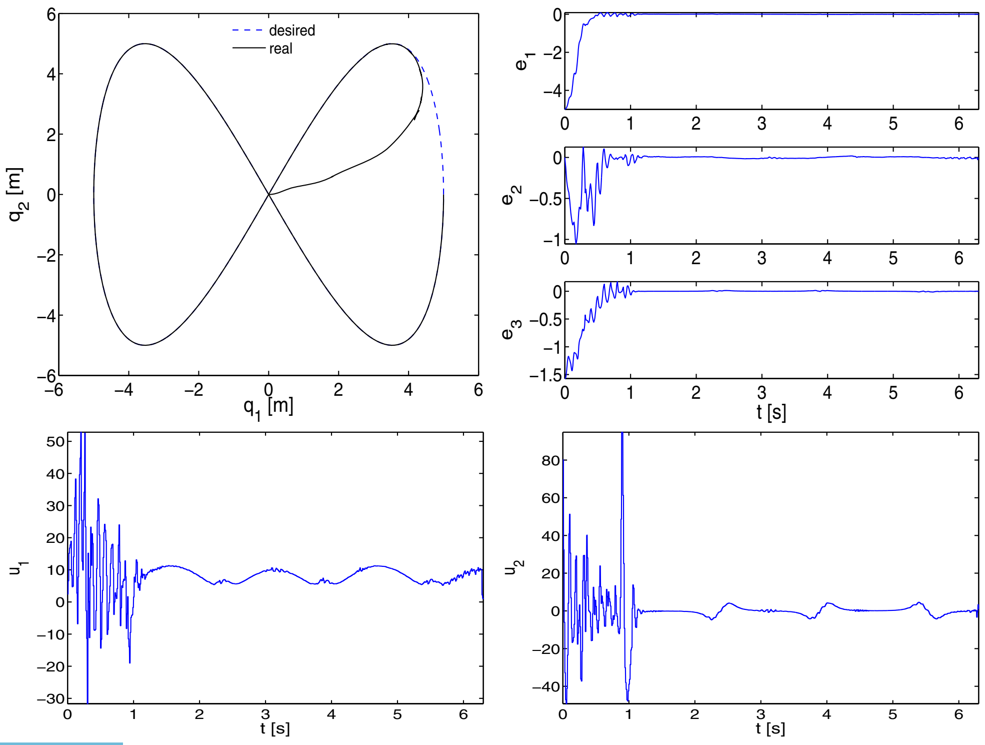
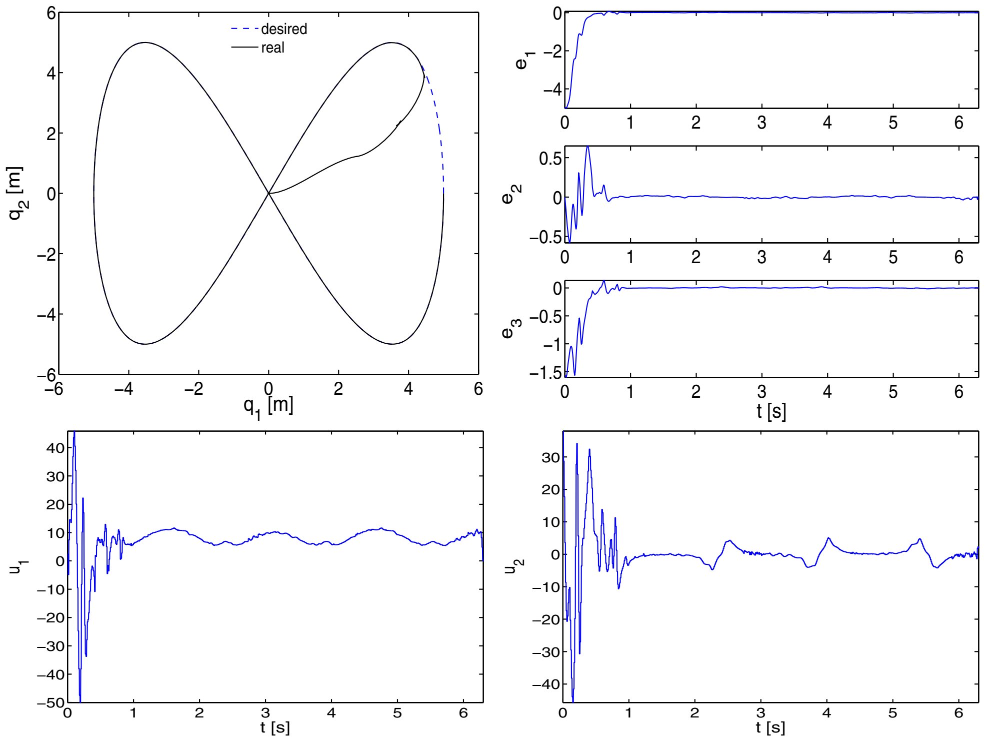
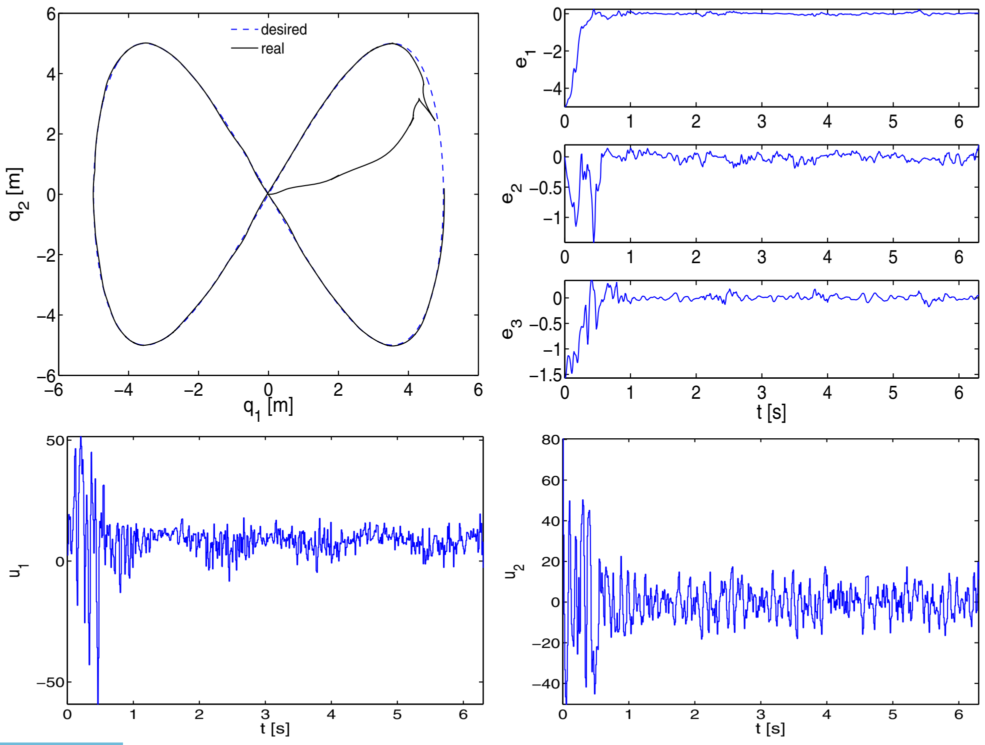


Fig. 13 ECSA 6: $T_p = 1.0$, $N = 5$, $M = 20$, $\kappa_{max} = 1$, without noise



well as a very good algorithm's convergence and stability, also in case of the relatively high state feedback random noise (ECSA 7: $q_{\text{noise}} = \pm(0.1\text{m}, 0.1\text{m}, 5°, 0°)$, and ECSA 8: $q_{\text{noise}} = \pm(0.5\text{m}, 0.5\text{m}, 10°, 0°)$). The algorithm preserves these properties even when the root finding solver performs only one step in each MPC iteration, a prediction horizon is relatively short $T_p = 0.5s$, and algorithm's parameters are selected appropriately (ECSA 1: $N = 5, M = 10$ vs ECSA 3: $N = 10, M = 5$). The tracking quality and stability of the predictive algorithm can be

Fig. 14 ECSA 7: $T_p = 0.5$, $N = 5$, $M = 10$, $\kappa_{max} = 1$, $q_{\text{noise}} = \pm(0.1\text{m}, 0.1\text{m}, 5°, 0°)$



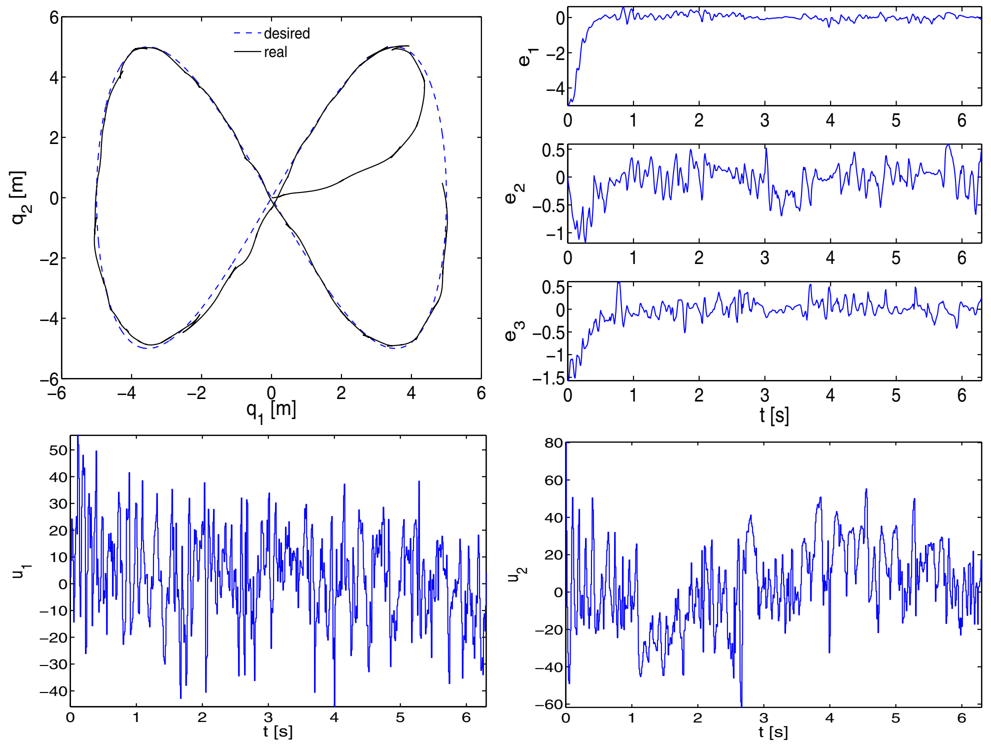


Fig. 15 ECSA 8: $T_p = 0.5$, $N = 5$, $M = 10$, $\kappa_{max} = 1$, $q_{\text{noise}} = \pm(0.5\text{m}, 0.5\text{m}, 10^\circ, 0^\circ)$

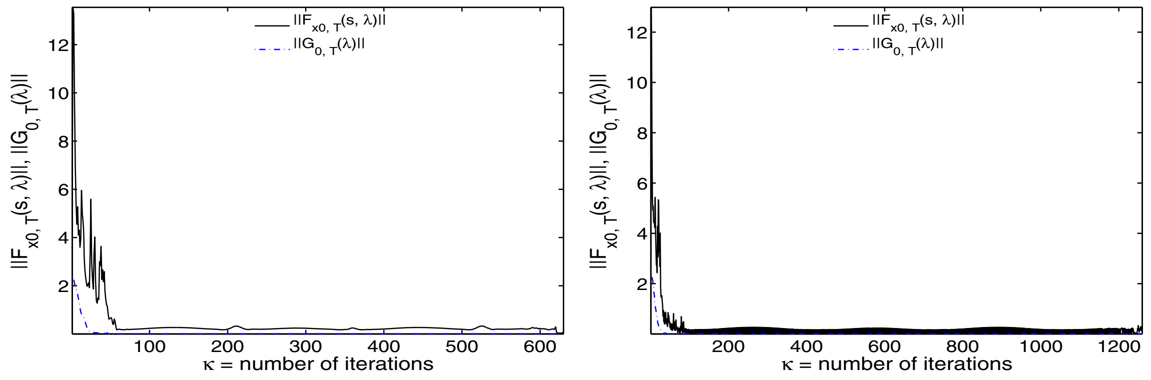


Fig. 16 Primary and secondary task error (left: ECSA 1, right: ECSA 2)

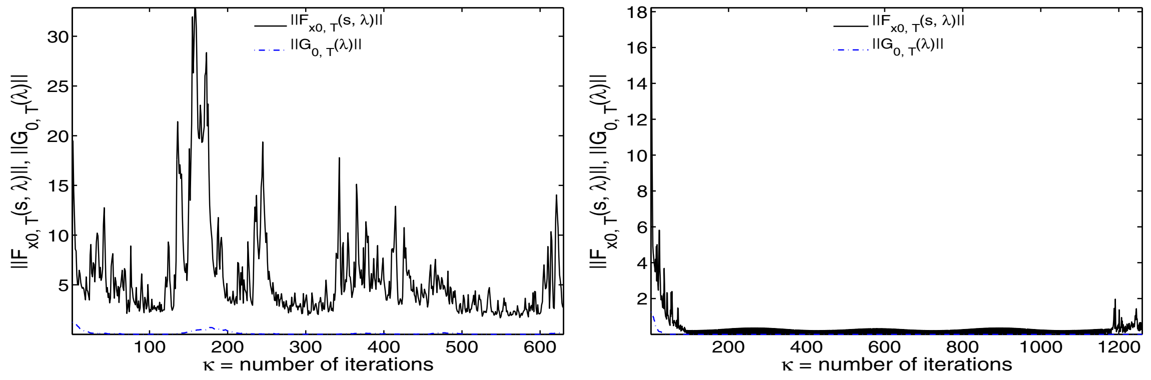


Fig. 17 Primary and secondary task error (left: ECSA 3, right: ECSA 4)

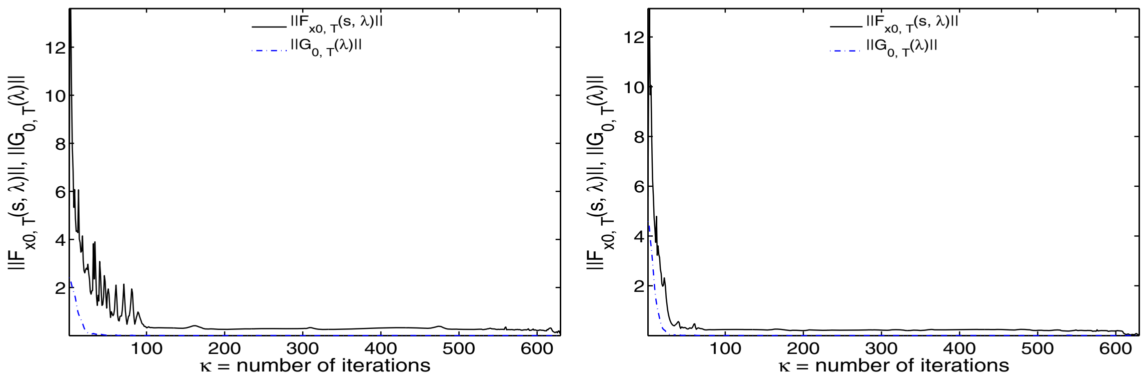


Fig. 18 Primary and secondary task error (left: ECSA 5, right: ECSA 6)

improved either by increasing the number of solver steps executed in each iteration (compare ECSA 1: $\kappa_{max} = 1$ with ECSA 2: $\kappa_{max} = 2$, and ECSA 3: $\kappa_{max} = 1$ with ECSA 4: $\kappa_{max} = 2$) or by extending the prediction horizon along with the dimension of the control parametrization (see ECSA 5: $T_p = 1s$, $N = 10$, $M = 10$, $\kappa_{max} = 1$, and ECSA 6: $T_p = 1s$, $N = 5$, $M = 20$, $\kappa_{max} = 1$). Such algorithm properties were expected, and in this respect are similar to the properties of the classical MPC algorithms. Slightly surprising is the fact that the closed-loop algorithm behaves better when the dimension of the control parametrization M is greater than the number of intermediate states N – compare ECSA 1 and ECSA 3. Due to the properties of the Lifted Newton algorithm, a different behavior might be expected. A closer look at the Figs. 16, 17, 18 and 19 shows that the moving horizon of the predictive algorithm significantly disturbs convergence of the primary task that is responsible for maintaining continuity of the system flow. In each algorithm step, waypoints are moving forward along with the desired trajectory, which results in an increase in the distance between waypoints and respective intermediate states. Therefore, above a certain number of intermediate states, the increase in task space error may exceed the rate of convergence of the control algorithm, resulting in losing stability. In this case the only remedy is to perform more solver steps in each iteration. It is worth stressing that the trajectory tracking task has a lower priority, thus according to the task priority approach it operates inside the Jacobian kernel of the primary task. The kernel dimension depends directly on the dimension of the control parametrization M. For greater M, the root finding algorithm with priorities has more degrees of freedom whereby provides a better solution.

In case of the NMPC algorithm, the trajectory tracking problem is formulated in terms of the following optimal control problem

$$\min_{u(\cdot)} \mathcal{J}\big(u(\cdot)\big) = \int_{\tau}^{\tau+T_p} \big(y(t)-y_d(t)\big)^T P\big(y(t)-y_d(t)\big)+u(t)^T Qu(t)\,dt + \big(y(T)-y_d(T)\big)^T P\big(y(T)-y_d(T)\big), \quad (22)$$

subject to Eq. 21, where weight matrices $P = 100 \cdot \mathbb{I}_3$, and $Q = 0.005 \cdot \mathbb{I}_2$. Control signals have the same form of piecewise constant signals, with the same initial values, and the control interval T_c as in the ECSA algorithm. The problem (22) has been implemented in C++ language [35] with use of the master branch of the ACADO Toolkit (the short hash commit `21be39c`), and the g++ v5.4.0 compiler. Tests have been carried out on the same machine. Results of

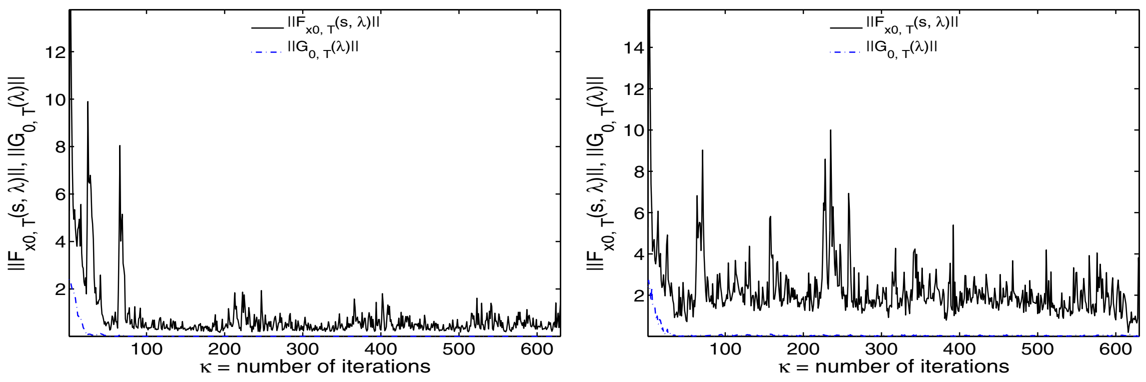


Fig. 19 Primary and secondary task error (left: ECSA 7, right: ECSA 8)

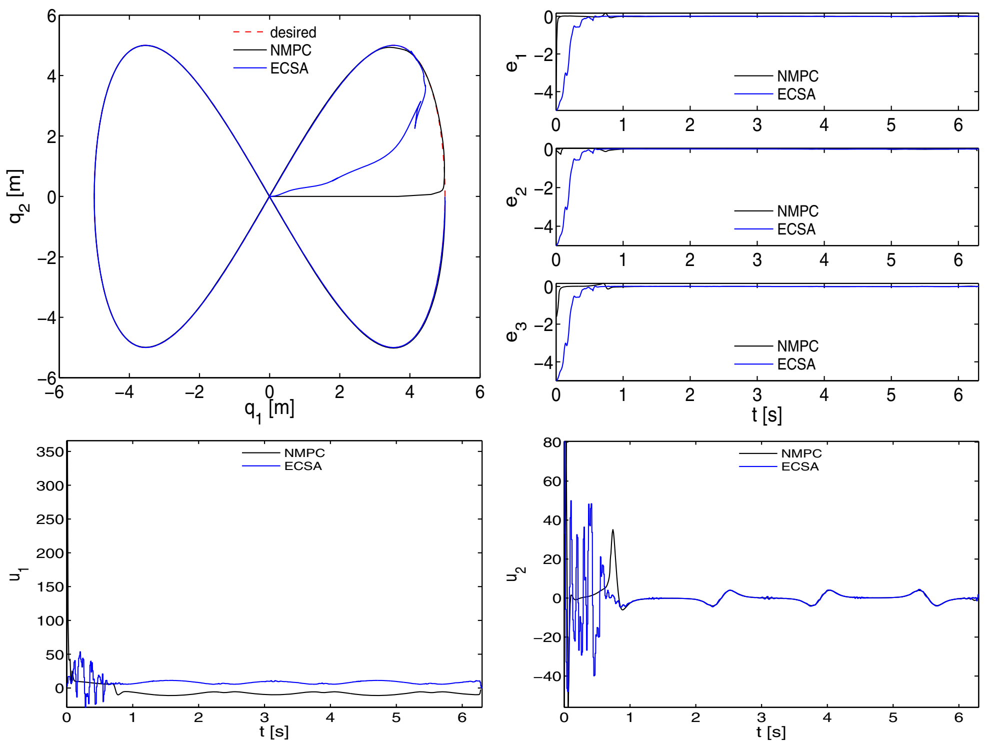


Fig. 20 ECSA 1 vs NMPC 1 ($T_p = 0.5$, $N = 50$, $\kappa_{max} = 1$), without noise

computations for various values of parameters T_p, N, κ_{max}, with and without noise q_{noise}, are displayed in Figs. 20, 21, 22 and 23 along with corresponding ECSA results.

In each case, the following condition holds $T_c = \frac{T_p}{N}$. The plots layout is the same as before.

Comparison of the ECSA and the NMPC algorithms for the same simulation conditions (T_p, T_c, κ_{max}) in the ideal case without state disturbances, shows that the NMPC behaves better than the ECSA algorithm. The NMPC algorithm convergences faster, and generates

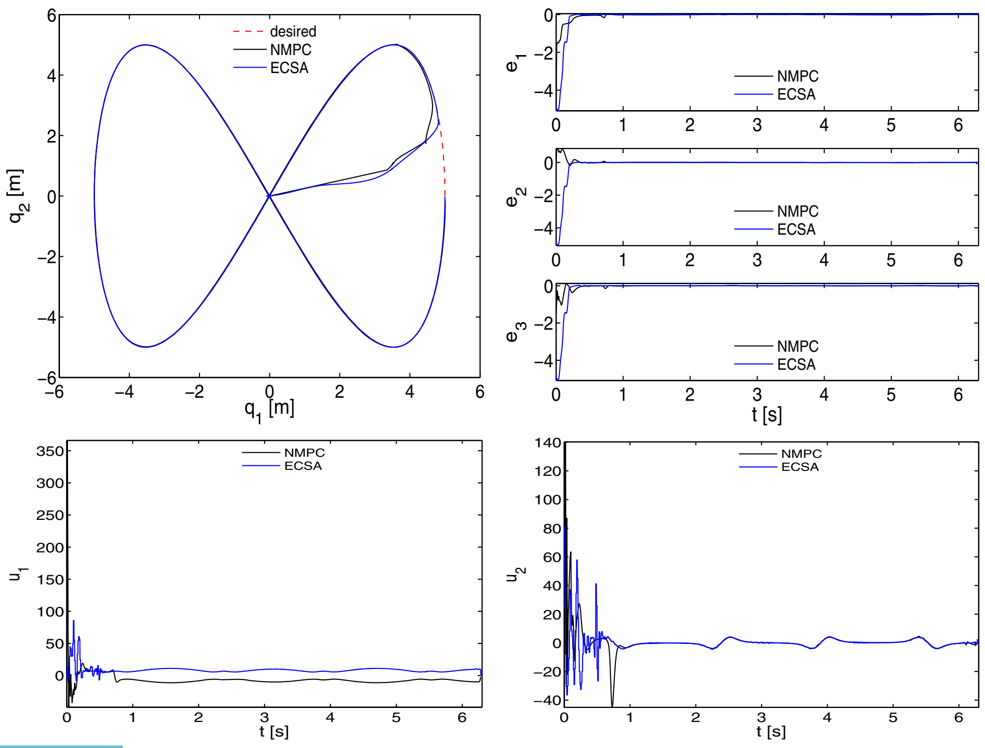


Fig. 21 ECSA 2 vs NMPC 2 ($T_p = 0.5$, $N = 50$, $\kappa_{max} = 2$), without noise

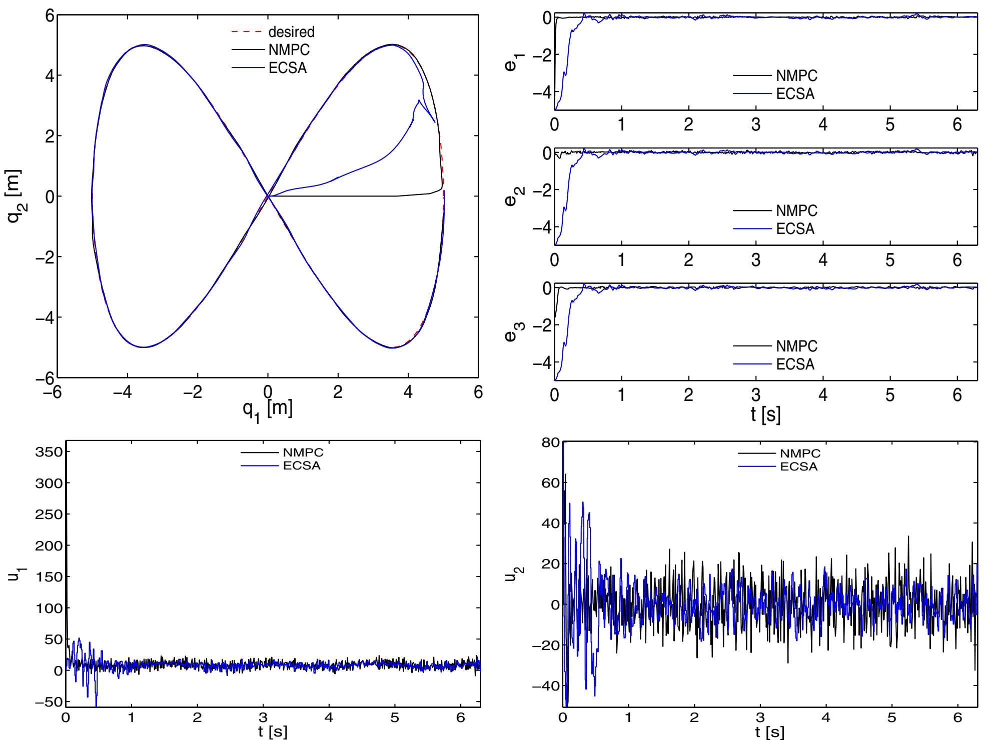


Fig. 22 ECSA 7 vs NMPC 3 ($T_p = 0.5$, $N = 50$, $\kappa_{max} = 1$), $q_{\text{noise}} = \pm(0.1\text{m}, 0.1\text{m}, 5°, 0°)$

smother trajectories (ECSA 1 vs NMPC 1, and ECSA 2 vs NMPC 2). Due to the nature of the ECSA algorithm, that defines the trajectory tracking task as a low priority, secondary task, its resulting performance may be slightly worse than the NMPC algorithm. The deterioration of the trajectory tracking quality in the initial simulation phase of the NMPC 2, when the control algorithm performs two optimization steps in each iteration, is unexpected and counterintuitive. In this case the resulted trajectory starts to resemble the ECSA 2 trajectory, what raise the

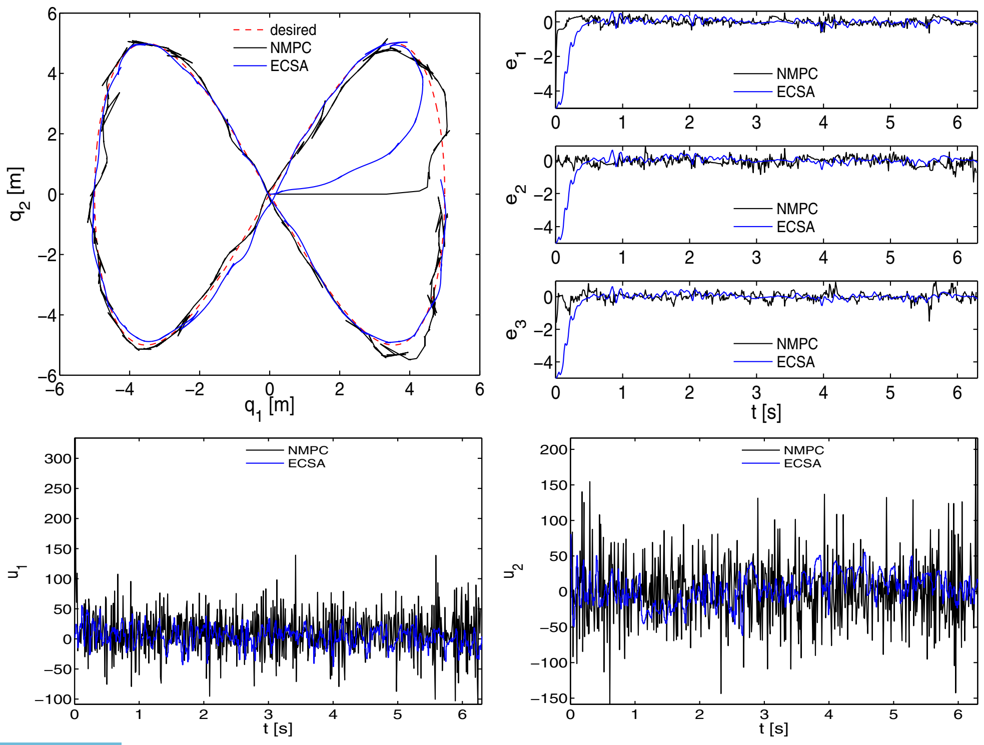


Fig. 23 ECSA 8 vs NMPC 4 ($T_p = 0.5$, $N = 50$, $\kappa_{max} = 1$), $q_{\text{noise}} = \pm(0.5\text{m}, 0.5\text{m}, 10°, 0°)$

question about the optimality of the ECSA solution. This will be the subject of further investigation. In the initial phase of simulation, the NMPC control signals have a larger amplitude than the ECSA, mainly due to a faster convergence and lack of a predefined control constraints. Both algorithms, when reach the desired trajectory, behave similar and generate a comparable control signals.

Interesting results are given by simulations that include feedback noise. A relatively low, uniform random noise $q_{\text{noise}} = \pm(0.1\text{m}, 0.1\text{m}, 5°, 0°)$ only slightly affects the resulted trajectories and control signals generated by both algorithms (ECSA 7 and NMPC 3). The noise propagation can be observed on the error and control plots. Both algorithms are stable, and convergent fast, however ECSA algorithm seems to produce less noisy signals, especially e_2 and u_2. In the presence of larger noise $q_{\text{noise}} = \pm(0.5\text{m}, 0.5\text{m}, 10°, 0°)$, this observation becomes more evident – compare ECSA 8 and NMPC 4. The ECSA behaves much better than NMPC, the trajectory is smother, and the error and control signals are much less noisy in the amplitude and frequency range. The ECSA algorithm can be considered as a low pass filter for feedback noise, due to its slower convergence rate than the NMPC algorithm.

Since all simulations have been carried out without any control constraints, the control signals have significant amplitudes in the initial phase of the platform movement, when tracking error is relatively large. Such controls can be difficult to implement in a real object. To make resulting controls closer to practical applications, the constraints can be included using the approach presented in [19]. This will be a subject of further research.

6 Summary

Following the MPC approach, a predictive closed-loop trajectory tracking algorithm has been presented. The trajectory tracking problem has been formulated as two conjugated root finding problems. In each iteration of the MPC scheme, the solution of the tracking problem is provided by the open-loop task priority Lifted Newton method. The performance of the closed-loop algorithm has been tested with computer simulations. The computer simulations have shown that the presented algorithm is able to track desired trajectory very efficiently even in the presence of a relatively large state feedback noise. This is manifested by a very good algorithm's convergence, stability and robustness. The tracking quality and stability of the presented predictive algorithm can be improved in the very similar way as with a traditional MPC, either by extending the prediction horizon or by increasing the number of solver steps executed in each iteration. Due to the properties of the Lifted Newton algorithm, it is better to increase the dimension of the control parametrization M than number of intermediate states N, so the algorithm has more degrees of freedom whereby provides a better solution. Comparison of the ECSA with the NMPC algorithm shows that in close proximity to the desired trajectory both algorithms behaves similar, however the NMPC converges faster at the expense of larger initial control signals values, and greater sensitivity to the state disturbances. The ECSA algorithm seems to be more robust to feedback noise, and in the presence of control constraints may show equivalent convergence rate.

As a further research we plan to develop a formal proof of the closed-loop algorithm's stability and robustness, and implement the real-time iteration scheme. The presented algorithm is open to further improvements. A more general framework can be defined that involves multiple tasks with different priorities e.g. tracking different trajectories by subsystems, respecting the state and control bounds, and minimizing the control energy.

Acknowledgements This research was supported by the Wrocław University of Science and Technology under a statutory research project. The authors are indebted to the anonymous referees whose comments unveiled advantages and limitation of the approach presented in this paper.

References

1. Albersmeyer, J., Diehl, M.: The lifted newton method and its application in optimization. SIAM J. on Optimization **20**(3), 1655–1684 (2010)
2. Alexe, M., Sandu, A.: Forward and adjoint sensitivity analysis with continuous explicit Runge-Kutta schemes. Appl. Math. Comput. **208**(2), 328–346 (2009)
3. Amrein, M., Wihler, T.P.: An adaptive newton-method based on a dynamical systems approach. Commun. Nonlinear Sci. Numer. Simul. **19**(9), 2958–2973 (2014)
4. Biegler, L.T.: A survey on sensitivity-based nonlinear model predictive control. IFAC Proceedings **46**(32), 499–510 (2013)
5. Cueli, J.R., Bordons, C.: Iterative nonlinear model predictive control. Stability, robustness and applications. Control. Eng. Pract. **16**(9), 1023–1034 (2008)
6. Diehl, M., Bock, H.G., Schlöder, J.P.: A real-time iteration scheme for nonlinear optimization in optimal feedback control. SIAM J. Control. Optim. **43**(5), 1714–1736 (2005)
7. Diehl, M., Findeisen, R., Allgower, F., Bock, H.G., Schloder, J.P.: Nominal stability of real-time iteration scheme for nonlinear

model predictive control. IEE Proceedings - Control Theory Appl. **152**(3), 296–308 (2005)
8. Divelbiss, A.W., Wen, J.T.: A path space approach to nonholonomic motion planning in the presence of obstacles. IEEE Trans. Robot. Autom. **13**(3), 443–451 (1997)
9. van Duijkeren, N., Verschueren, R., Pipeleers, G., Diehl, M., Swevers, J.: Path-Following NMPC for Serial-Link Robot Manipulators Using a Path-Parametric System Reformulation. In: 2016 European Control Conference (ECC), pp. 477–482 (2016)
10. Findeisen, R., Imsland, L., Allgower, F., Foss, B.A.: State and output feedback nonlinear model predictive control: an overview. Eur. J. Control. **9**(2), 190–206 (2003)
11. Grne, L., Pannek, J.: Nonlinear Model Predictive Control: Theory and Algorithms. Springer Publishing Company, Incorporated, New York (2013)
12. Grosan, C., Abraham, A.: A new approach for solving nonlinear equations systems. IEEE Trans. Syst., Man, Cybernetics - Part A: Syst. and Humans **38**(3), 698–714 (2008)
13. Guerreiro, B.J., Silvestre, C., Cunha, R., Pascoal, A.: Trajectory Tracking Nonlinear Model Predictive Control for Autonomous Surface Craft. In: 2009 European Control Conference (ECC), pp. 1311–1316 (2009)
14. Guerreiro, B.J., Silvestre, C., Cunha, R., Pascoal, A.: Trajectory tracking nonlinear model predictive control for autonomous surface craft. IEEE Trans. Control Syst. Technol. **22**(6), 2160–2175 (2014)
15. Houska, B., Ferreau, H.J., Diehl, M.: ACADO toolkit—An Open-source framework for automatic control and dynamic optimization. Optim. Control Appl. Meth. **32**, 298–312 (2011)
16. Janiak, M.: Lifted Newton Motion Planning Algorithm. In: 2015 10th International Workshop on Robot Motion and Control (Romoco), pp. 223–228 (2015)
17. Janiak, M.: From Motion Planning through Waypoints to Open-Loop Trajectory Tracking Algorithm. In: 2017 11Th International Workshop on Robot Motion and Control (Romoco), pp. 142–147 (2017)
18. Janiak, M., Tchoń, K.: Motion Planning through Waypoints for a Skid-Steering Mobile Platform. In: 2015 10th International Workshop on Robot Motion and Control (Romoco), pp. 58–63 (2015)
19. Janiak, M., Tchon, K.: Constrained motion planning of nonholonomic systems. Syst. Control Lett. **60**(8), 625–631 (2011)
20. Karpińska, J., Tchoń, K.: Continuation Method Approach to Trajectory Planning in Robotic Systems. In: 2011 16th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 51–56 (2011)
21. Kowalczyk, W., Michałek, M., Kozłowski, K.: Trajectory tracking control with obstacle avoidance capability for unicycle-like mobile robot. Bulletin of the Polish Academy of Sciences. Tech. Sci. **60**(3), 537–546 (2012)
22. Maurović, I., Baotić, M., Petrović, I.: Explicit Model Predictive Control for Trajectory Tracking with Mobile Robots. In: 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 712–717 (2011)
23. Mazur, A., Cholewiński, M.: Implementation of factitious force method for control of 5R manipulator with skid-steering platform REX. Bulletin of the Polish Academy of Sciences. Tech. Sci. **64**(1), 71–80 (2016)
24. Morin, P., Samson, C.: Stabilization of Trajectories for Systems on Lie Groups. Application to the Rolling Sphere. In: 17th IFAC World Congress, pp. 508–513 (2008)
25. Muszyniski, R., Jakubiak, J.: On Predictive Approach to Inverse Kinematics of Mobile Manipulators. In: 2007 IEEE International Conference on Control and Automation, pp. 2423–2428 (2007)
26. Qin, S., Badgwell, T.A.: A survey of industrial model predictive control technology. Control. Eng. Pract. **11**(7), 733–764 (2003)
27. Qin, S.J., Badgwell, T.A.: An Overview of Nonlinear Model Predictive Control Applications, pp. 369–392. Basel, Birkhäuser Basel (2000)
28. Quarteroni, A., Sacco, R., Saleri, F.: Nonlinear Systems and Numerical Optimization, pp. 285–331. Springer, Berlin (2007)
29. Quirynen, R., Vukov, M., Zanon, M., Diehl, M.: Autogenerating microsecond solvers for nonlinear mpc: a tutorial using acado integrators. Optim. Control Appl. Method **36**(5), 685–704 (2015)
30. Ratajczak, A.: Trajectory reproduction and trajectory tracking problem for the nonholonomic systems. Bull. Polish Academy of Sci. **64**(1), 63–70 (2016)
31. Ratajczak, A., Tchoń, K.: Multiple-task motion planning of nonholonomic systems with dynamics. Mech. Sci. **4**(1), 153–166 (2013)
32. Rawlings, J.B.: Tutorial overview of model predictive control. IEEE Control. Syst. **20**(3), 38–52 (2000)
33. Sistu, P.B., Bequette, B.W.: Nonlinear model-predictive control: Closed-loop stability analysis. AIChE J **42**(12), 3388–3402 (1996)
34. Tchoń, K., Jakubiak, J.: Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of Jacobian inverse kinematics algorithms. Int. J. Contr. **76**(14), 1387–1419 (2003)
35. Łukasz, C.: Example of model predictive control simulation. https://bitbucket.org/lukych92/acado_mpc_simulation
36. Walsh, G., Tilbury, D., Sastry, S., Murray, R., Laumond, J.P.: Stabilization of trajectories for systems with nonholonomic constraints. IEEE Trans. Autom. Control **39**(1), 216–222 (1994)

Mariusz Janiak is an Assistant Professor at Department of Cybernetics and Robotics in the Wrocław University of Science and Technology. He received his PhD in Control Engineering and Robotics from the Wrocław University of Science and Technology in 2010. His research interest includes constrained motion planning of nonholonomic robotic systems, nonlinear control, and real-time distributed control systems. He has participated in several EU and national funded research projects: LIREC, RobREx, ReMeDi. He has also served as reviewer for multiple relevant journals and conferences.

Łukasz Chojnacki received B.S. and M.S. in Control Engineering and Robotics from Wroclaw University of Science and Technology, Wroclaw, Poland in 2016 and 2017, respectively. His current research interests include the area of mobile robots, in particular, motion planning, trajectory tracking, navigation, and mapping.