A Flatness-Based Iterative Method for Reference Trajectory Generation in Constrained NMPC

J.A. De Doná, F. Suryawan, M.M. Seron, and J. Lévine

Abstract. This paper proposes a novel methodology that combines the differential flatness formalism for *trajectory generation* of nonlinear systems, and the use of a model predictive control (MPC) strategy for *constraint handling*. The methodology consists of a *trajectory generator* that generates a reference trajectory parameterised by splines, and with the property that it satisfies performance objectives. The reference trajectory is generated *iteratively* in accordance with information received from the MPC formulation. This interplay with MPC guarantees that the trajectory generator receives *feedback from present and future constraints* for real-time trajectory generation.

Keywords: flatness, trajectory generation, B-splines, Nonlinear MPC.

1 Introduction

Differential flatness [1] is a property of some controlled (linear or nonlinear) dynamical systems, often encountered in applications, which allows for a complete parameterisation of all system variables (inputs and states) in terms of a finite number of independent variables, called *flat outputs*, and a finite number of their time derivatives. We consider a general system

$$\dot{x}(t) = f\big(x(t), u(t)\big), \tag{1}$$

J.A. De Doná, F. Suryawan, and M.M. Seron
CDSC, School of Electrical Engineering and Computer Science,
The University of Newcastle, Callaghan, NSW 2308, Australia
e-mail: {jose.dedona,fajar.suryawan,maria.seron}@newcastle.edu.au

J.A. De Doná and J. Lévine
CAS, Mathématiques et Systèmes, Mines-ParisTech, 35 rue Saint-Honoré.
77300 Fontainebleau, France
e-mail: jean.levine@ensmp.fr

L. Magni et al. (Eds.): Nonlinear Model Predictive Control, LNCIS 384, pp. 325–333.
springerlink.com

where $x(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathbb{R}^m$ is the input vector. If the system is flat [1], we can write all trajectories $(x(t), u(t))$ satisfying the differential equation in terms of a finite set of variables, known as the flat output, $y(t) \in \mathbb{R}^m$ and a finite number of their derivatives:

$$\begin{aligned} x(t) &= \Upsilon\big(y(t), \dot{y}(t), \ddot{y}(t), \ldots, y^{(r)}(t)\big), \\ u(t) &= \Psi\big(y(t), \dot{y}(t), \ddot{y}(t), \ldots, y^{(r+1)}(t)\big). \end{aligned} \tag{2}$$

The parameterisation (2), afforded by the flatness property, allows to simplify (especially in the case of *nonlinear* flat systems) the generation of reference trajectories (trajectory planning). Typically, some 'desired' reference trajectory is prescribed for the flat output, y^{ref}, and the corresponding input and state trajectories for the system are obtained from (2); namely, $u^{\text{ref}}(t) = \Psi\big(y^{\text{ref}}(t), \dot{y}^{\text{ref}}(t), \ldots, (y^{\text{ref}}(t))^{(r+1)}\big)$, $x^{\text{ref}}(t) = \Upsilon\big(y^{\text{ref}}(t), \ldots, (y^{\text{ref}}(t))^{(r)}\big)$. However, a very common requirement in engineering applications is for some of the variables of the dynamical system to satisfy a number of constraints, usually expressed as *inequality constraints*. For example the input and state of the system can be required to satisfy $u \in \mathbb{U}$ and $x \in \mathbb{X}$, where $\mathbb{U} \subset \mathbb{R}^m$ and $\mathbb{X} \subset \mathbb{R}^n$ are specified *constraint sets*. The presence of such constraints makes trajectory generation for nonlinear systems (in general) a highly nontrivial task, due to the ensuing nonlinearity of the mappings $\Upsilon(\cdot)$ and $\Phi(\cdot)$ in (2). (In particular, it is typically very difficult, to specify constraint sets for the flat output variables y in terms of the constraint sets for u and x, respectively, $\mathbb{U}$ and $\mathbb{X}$.)

In this paper, we propose a novel methodology that exploits the flatness parameterisation (2) for *trajectory generation* and the use of the Model Predictive Control (MPC) strategy for *constraints handling*. The methodology consists of a *trajectory generator* module, that generates a reference trajectory $y^{\text{ref}}(t)$ with the property that it satisfies performance objectives (e.g., satisfies given *initial* and *final conditions*, passes through a given set of *way-points*, etc.). There are points of contact between some aspects of the approach advocated in this paper and, for example, the work in [3] where the problem of generation of a reference trajectory for a nonlinear flat system subject to constraints is formulated as a NonLinear Programming (NLP) problem. One of the main drawbacks of posing the problem as a NLP optimisation problem is that, in general, it is very difficult to prove convergence, or convergence to a global optimum. Hence, in this paper we explore an alternative algorithm for trajectory generation for nonlinear flat systems, in the presence of constraints, that is based on the information provided by a model predictive control (MPC) formulation. The approach is, to the best of the authors knowledge, the first attempt to combine the differential flatness formalism with model predictive control techniques in an iterative algorithm for constrained nonlinear trajectory generation. No proofs of convergence are available at present, due to the challenging nature of these problems, and this will be of concern in future work. However, simulation results, as the ones presented in this paper, are promissory and indicate that the effort of developing such algorithms and investigating formal proofs of convergence is worthwhile.

Thus, in the methodology investigated in this paper, the reference trajectory $y^{\text{ref}}(t)$ is generated *iteratively* in accordance with information (predicted in real time) received from an MPC formulation. That way, the trajectory generator receives "feedback from the (present and future) constraints" of the system while generating the desired trajectory. Thus, the proposed method unites two important properties. Firstly, since the trajectories are generated via the flatness parameterisation (2), with "feedback from the constraints," they constitute *natural* trajectories for the *nominal* model to follow. And, secondly, the information generated by an MPC formulation (via the solution of a Quadratic Programming optimisation, based on the linearised dynamics around the given reference trajectory) ensures that the system constraints are taken into account.

2 Flatness and Trajectory Parameterisation

We consider the problem of steering system (1) from an initial state at time t_0 to a final state at time t_f. Note that, in a Model Predictive Control context, this fixed interval problem is one window of a bigger scheme, implemented repeatedly in a receding horizon fashion. In order to generate a suitable reference trajectory, we will use a spline parameterisation, as explained in the following sections.

2.1 Parameterisation of Flat Outputs and Their Derivatives

We parameterise each of the flat outputs $y_j(t)$, $j = 1,\ldots,m$, as

$$y_j(t) = \sum_{i=1}^{N} \lambda_i(t) P_{ij}; \qquad t \in [t_0\ ,\ t_f], \tag{3}$$

where $\lambda_i, i = 1,\ldots,N$, is a set of basis functions, which is the same for each flat output y_j. The basis functions are assumed to be $\lambda_i \in \mathscr{C}^{r+1}[t_0,t_f], i = 1,\ldots,N$. This reduces the problem of characterising a function in an infinite dimensional space to finding a finite set of parameters P_{ij}. In a discrete set of $M+1$ sampling times, $t_0,t_1,\ldots,t_M = t_f$, this parameterisation becomes

$$Y_j = G_0 P_j, \tag{4}$$

where $Y_j \triangleq [y_j(t_0), y_j(t_1), \ldots, y_j(t_f)]^{\mathrm{T}}$, $P_j \triangleq [P_{1j},\ldots,P_{Nj}]^{\mathrm{T}}$ is a vector containing the parameters P_{ij}, $i = 1,\ldots,N$, defined in (3), and

$$G_0 \triangleq \begin{bmatrix} \lambda_1(t_0) & \ldots & \lambda_N(t_0) \\ \vdots & \ddots & \vdots \\ \lambda_1(t_f) & \ldots & \lambda_N(t_f) \end{bmatrix} \tag{5}$$

is the *basis function matrix* (also known as *blending matrix*). Collecting all the m flat outputs, we have

$$
\begin{aligned}
Y \triangleq \begin{bmatrix} Y_1 \; Y_2 \; \dots \; Y_m \end{bmatrix} &= \begin{bmatrix} y_1(t_0) & y_2(t_0) & \dots & y_m(t_0) \\ \vdots & \vdots & \ddots & \vdots \\ y_1(t_f) & y_2(t_f) & \dots & y_m(t_f) \end{bmatrix} \\
&= G_0 \cdot \begin{bmatrix} P_1 \; P_2 \; \dots P_m \end{bmatrix} = G_0 P = Y(P),
\end{aligned} \tag{6}
$$

where Y is an $(M+1)\times m$ output matrix, G_0 is the $(M+1)\times N$ blending matrix, and $P \triangleq \begin{bmatrix} P_1 \; P_2 \; \dots P_m \end{bmatrix}$ is an $N \times m$ matrix containing the coefficients P_{ij} of the parameterisation (3). The rows of P are m-dimensional vectors called *control points*.

Furthermore, we can also build the time-derivatives of y_i at discrete points in time, by successively differentiating (3) followed by time-discretisation. Doing this and using the notation as in (6), we obtain

$$
Y^{(1)} = G_1 P; \quad Y^{(2)} = G_2 P; \quad Y^{(3)} = G_3 P; \quad \dots \quad Y^{(r+1)} = G_{r+1} P; \tag{7}
$$

where $Y^{(q)} \triangleq [Y_1^{(q)} \; Y_2^{(q)} \; \dots \; Y_m^{(q)}]$, and

$$
Y_j^{(q)} \triangleq \begin{bmatrix} \frac{d^q}{dt^q} y_j(t)\big|_{t=t_0} \\ \vdots \\ \frac{d^q}{dt^q} y_j(t)\big|_{t=t_f} \end{bmatrix}; \qquad G_q \triangleq \begin{bmatrix} \frac{d^q}{dt^q}\lambda_1(t)\big|_{t=t_0} & \dots & \frac{d^q}{dt^q}\lambda_N(t)\big|_{t=t_0} \\ \vdots & \ddots & \vdots \\ \frac{d^q}{dt^q}\lambda_1(t)\big|_{t=t_f} & \dots & \frac{d^q}{dt^q}\lambda_N(t)\big|_{t=t_f} \end{bmatrix}, \tag{8}
$$

with $j = 1,\dots,m$ and $q = 1,\dots,r+1$.

2.2 Trajectory Parameterisation Using Splines

Given a reference trajectory parameterised as in (6), $Y^{\text{ref}} = G_0 P^{\text{ref}}$, with specified reference control points P^{ref}, in this section we will show how to parameterise variations around that reference trajectory using splines. In this paper, clamped B-splines [2] are chosen as basis functions which results in the blending matrix G_0 having a particular structure. Namely, G_0 has only one non-zero element in the first row (which lies in the first column) and only one non-zero element in the last row (which lies in the last column). The matrix G_1 has two non-zero elements in the first row (which lie in the first and second column) and two non-zero elements in the last row (which lie in the last and second-last column). The matrix G_2 has a similar property with three non-zero elements, etc. More properties of B-splines can be found in, e.g., [2].

Notice from (3) that,

$$
\frac{d^q}{dt^q} y_j(t)\big|_{t=t_0} = \sum_{i=1}^{N} \frac{d^q}{dt^q}\lambda_i(t)\big|_{t=t_0} P_{ij}, \tag{9}
$$

for $q = 0,1,\dots,r+1$; $j = 1,\dots,m$. We can see from (9) and the structure of G_0 discussed above that $y_j(t_0) = \lambda_1(t_0) P_{1j}$, $j = 1,\dots,m$, and hence, by fixing the first row of P, $P_{1j} = P_{1j}^{\text{ref}}$, $j = 1,\dots,m$, the flat outputs at time t_0 are fixed and equal to the corresponding values of the reference trajectory. Fixing more rows of P (up to the

order of the B-spline) fixes the flat output derivatives at time t_0 (e.g. fixing two rows fixes the first derivatives, three rows fixes the second derivatives, etc.). This property (made possible by the structure of G_q, $q = 0, 1, \ldots$) can be used to maintain fixed end-points. For example, prescribed position and first and second order derivatives of the flat output at times t_0 and t_f, as in the *rest-to-rest* case, can be maintained by holding the 'external' control points (the three topmost and the three bottommost rows of P in Eq. (6)) fixed. This can be achieved by reparameterising P as:

$$P = P^{\text{ref}} + \rho \hat{P}; \qquad \rho = [0\, I\, 0]^{\text{T}}, \tag{10}$$

where matrix $\hat{P}$ is an $[N-(l_1+l_2)] \times m$ matrix that parameterises the deviation from the 'internal' control points of P^{ref} and ρ is an $N \times [N-(l_1+l_2)]$ matrix with the l_1 top rows set equal to zero, the l_2 bottom rows set equal to zero and the identity matrix of dimension $[N-(l_1+l_2)] \times [N-(l_1+l_2)]$ in the middle.

3 Using MPC to Shape the Reference Trajectory

In this section we will develop an iterative algorithm for trajectory generation for nonlinear systems, subject to constraints, that is based on information provided by model predictive control (MPC). The main motivation for resorting to MPC is to exploit the well-known capabilities for handling constraints of this control technique. The basic idea is to propose an initial reference trajectory based purely on performance considerations, parameterised as in (6), i.e. $Y^{\text{ref},0} = G_0 P^{\text{ref},0}$ (it is assumed here that an initial set of *reference* control points $P^{\text{ref},0}$ is specified), and to then use an MPC formulation to give information as to how well that trajectory can be followed in the presence of constraints and, moreover, which parts of the original trajectory are problematic and should be modified. Then a new reference trajectory is generated based on a trade-off between the information obtained from MPC (this information can be regarded as the *feedback from the constraints*) and the original performance specifications. This interplay between performance objectives and MPC (feedback from constraints) is then iterated, and the challenge is to devise an algorithm such that the iteration converges to a suitable reference trajectory.

3.1 MPC Formulation

We will assume, for simplicity, that the flat output is given by a (possibly nonlinear) combination of the states:

$$y(t) = h\big(x(t)\big). \tag{11}$$

Note that, although this is not the most general case for flat systems, many examples of practical interest satisfy this assumption (e.g., models of cranes, nonholonomic cars, etc.). Given a specified reference trajectory for the flat output, parameterised by control points P^{ref} as explained in the preceding section:

$$y_j^{\text{ref}}(t) = \sum_{i=1}^{N} \lambda_i(t) P_{ij}^{\text{ref}}; \quad t \in [t_0, t_f], \tag{12}$$

for $j = 1, \ldots, m$, we compute the corresponding state and input reference trajectories, $x^{\text{ref}}(t)$ and $u^{\text{ref}}(t)$, respectively, from (2). Note, in particular, that the flatness formulation implies that these trajectories satisfy the system's equation $\dot{x}^{\text{ref}}(t) = f\big(x^{\text{ref}}(t), u^{\text{ref}}(t)\big)$. Then, the dynamics of (1) together with the output equation (11) are linearised along the reference trajectory $\big(u^{\text{ref}}(t), x^{\text{ref}}(t), y^{\text{ref}}(t)\big)$ as follows: $\dot{\tilde{x}}(t) = A(t)\tilde{x}(t) + B(t)\tilde{u}(t)$, $\tilde{y}(t) = C(t)\tilde{x}(t)$, where:

$$\tilde{u}(t) \triangleq u(t) - u^{\text{ref}}(t), \quad \tilde{x}(t) \triangleq x(t) - x^{\text{ref}}(t), \quad \tilde{y}(t) \triangleq y(t) - y^{\text{ref}}(t), \tag{13}$$

and $A(t) = \big(\partial f/\partial x\big)\big|_{x^{\text{ref}}(t), u^{\text{ref}}(t)}$, $B(t) = \big(\partial f/\partial u\big)\big|_{x^{\text{ref}}(t), u^{\text{ref}}(t)}$ and $C(t) = \big(\partial h/\partial x\big)\big|_{x^{\text{ref}}(t)}$. The resulting linear time varying system is then discretised in time, so that the following time varying discrete time system is obtained:

$$\tilde{x}_{k+1} = A_k \tilde{x}_k + B_k \tilde{u}_k, \qquad \tilde{y}_k = C_k \tilde{x}_k. \tag{14}$$

In the discretisation (14) we consider a sampling interval $T_s \triangleq (t_f - t_0)/M$, so that exactly M sampling intervals fit in the interval of definition of the splines, $[t_0, t_f]$. Moreover, we define a grid of equally spaced sampling times, $t_k = t_0 + kT_s$, $k = 0, \ldots, M$. Note that the variables in (14) (cf. (13)) are measured with respect to the reference trajectory. Thus we will consider an MPC formulation for the time varying system (14) where the performance objective is regulation to the origin (this will ensure tracking of the respective reference trajectories).

Given the current state of the plant at time t, $x(t)$, we compute $\tilde{x}_0 \triangleq x(t) - x^{\text{ref}}(t_0)$ (where $x^{\text{ref}}(t_0)$ is obtained from (12) using (2)). The aim is to find the M-move control sequence $\{\tilde{u}_k\} \triangleq \{\tilde{u}_0, \ldots, \tilde{u}_{M-1}\}$ that minimises the finite horizon objective function:

$$V_M(\{\tilde{x}_k\}, \{\tilde{u}_k\}, \{\tilde{y}_k\}) \triangleq \frac{1}{2}\tilde{x}_M^{\mathrm{T}} P \tilde{x}_M + \frac{1}{2}\sum_{k=0}^{M-1} \tilde{y}_k^{\mathrm{T}} Q \tilde{y}_k + \frac{1}{2}\sum_{k=0}^{M-1} \tilde{u}_k^{\mathrm{T}} R \tilde{u}_k, \tag{15}$$

subject to the system equations (14) and $\tilde{x}_0 \triangleq x(t) - x^{\text{ref}}(t_0)$, and where $P \geq 0$, $Q \geq 0$, $R > 0$, and M is the prediction horizon. Using the standard vectorised notation $\tilde{\mathbf{x}} \triangleq [\tilde{x}_1^{\mathrm{T}} \ldots \tilde{x}_M^{\mathrm{T}}]^{\mathrm{T}}$, $\tilde{\mathbf{u}} \triangleq [\tilde{u}_0^{\mathrm{T}} \ldots \tilde{u}_{M-1}^{\mathrm{T}}]^{\mathrm{T}}$, the cost function (15) can be written in compact form as:

$$V_M = \frac{1}{2}\tilde{x}_0^{\mathrm{T}} C_0^{\mathrm{T}} Q C_0 \tilde{x}_0 + \frac{1}{2}\tilde{\mathbf{x}}^{\mathrm{T}} \mathbf{Q} \tilde{\mathbf{x}} + \frac{1}{2}\tilde{\mathbf{u}}^{\mathrm{T}} \mathbf{R} \tilde{\mathbf{u}}, \tag{16}$$

where $\mathbf{Q} \triangleq \operatorname{diag}\{C_1^{\mathrm{T}} Q C_1, \ldots, C_{M-1}^{\mathrm{T}} Q C_{M-1}, P\}$ and $\mathbf{R} \triangleq \operatorname{diag}\{R, \ldots, R\}$.

The system's state evolution from $k = 0$ to M can be expressed as $\tilde{\mathbf{x}} = \Gamma \tilde{\mathbf{u}} + \Omega \tilde{x}_0$, where Γ and Ω are formed from the system's A_k and B_k matrices (see, e.g., [4]). Substituting this expression for $\tilde{\mathbf{x}}$ into (16) yields: $V_M = \bar{V} + \frac{1}{2}\tilde{\mathbf{u}}^{\mathrm{T}} H \tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{\mathrm{T}} F \tilde{x}_0$, where $\bar{V}$ is a constant term, $H \triangleq \Gamma^{\mathrm{T}} \mathbf{Q} \Gamma + \mathbf{R}$ and $F \triangleq \Gamma^{\mathrm{T}} \mathbf{Q} \Omega$.

If the problem is constrained, for example with input constraints $|u(t)| \le u_{\max}$, then the solution is obtained from the following quadratic program:

$$\begin{aligned} \tilde{\mathbf{u}}^{\text{opt}} = [(\tilde{u}_0^{\text{opt}})^{\text{T}} \dots (\tilde{u}_{M-1}^{\text{opt}})^{\text{T}}]^{\text{T}} \triangleq \arg\min_{\tilde{\mathbf{u}}} \frac{1}{2}\tilde{\mathbf{u}}^{\text{T}} H \tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{\text{T}} F \tilde{x}_0 \\ \text{subject to} \\ |\mathbf{u}^{\text{ref}} + \tilde{\mathbf{u}}| \le U_{\max}, \end{aligned} \tag{17}$$

where $\mathbf{u}^{\text{ref}} \triangleq [(u^{\text{ref}}(t_0))^{\text{T}}\ (u^{\text{ref}}(t_1))^{\text{T}} \dots (u^{\text{ref}}(t_{M-1}))^{\text{T}}]^{\text{T}}$, $U_{\max} \triangleq [u_{\max}^{\text{T}} \dots u_{\max}^{\text{T}}]^{\text{T}}$, and the absolute value and the inequality are interpreted element-wise. (Other types of constraints, e.g., state and output constraints, can be incorporated in (17) in a straightforward manner.)

The corresponding j-th flat output trajectory, $j = 1, \dots, m$, obtained by MPC is computed from the result of (17), using (13) and (14). Using the expression $\tilde{\mathbf{x}} = \Gamma\tilde{\mathbf{u}} + \Omega\tilde{x}_0$, the MPC flat output trajectory can be expressed as:

$$Y_j^{\text{mpc}} \triangleq \mathbf{C}_j \begin{bmatrix} \tilde{x}_0 \\ \Gamma\tilde{\mathbf{u}}^{\text{opt}} + \Omega\tilde{x}_0 \end{bmatrix} + Y_j^{\text{ref}}, \tag{18}$$

where Y_j^{mpc} and Y_j^{ref} are the j-th flat output sequences stacked over time [defined similarly to Y_j in (4)], and $\mathbf{C}_j \triangleq \text{diag}\{C_{0,j}, \dots, C_{M,j}\}$, where $C_{k,j}$ is the j-th row of the time-varying matrix C_k, defined in (14) for $k = 0, \dots, M$. In an MPC implementation, one then applies the first control move obtained in (17), $\tilde{u}_0^{\text{opt}}$, and the process is repeated in a receding horizon fashion. However, in our proposed implementation (see next subsection) this process is iterated before the actual control input is applied.

3.2 *Iterative Method for Reference Trajectory Generation*

In this section we present the iterative algorithm that is proposed in this paper. The algorithm starts from a set of specified initial control points $P^{\text{ref},0}$ that parameterise an initial reference trajectory $Y^{\text{ref},0} = G_0 P^{\text{ref},0}$ which is generated based on performance considerations, and then it utilises the information about the effect of the constraints, provided by the MPC formulation, to update the reference trajectory through successive sets of control points, $P^{\text{ref},0}, P^{\text{ref},1}, \dots, P^{\text{ref,k}}, \dots$, etc.

Step 1. Given a set of control points $P^{\text{ref,k}}$;

Step 2. Compute, from (6), $Y^{\text{ref,k}} = G_0 P^{\text{ref,k}}$;

Step 3. Compute $Y_j^{\text{mpc,k}}$ from (12)–(18). Note that $Y^{\text{mpc,k}}$ so obtained is a (in general nonlinear) function of $P^{\text{ref,k}}$, that is, $Y^{\text{mpc,k}} = G\big(P^{\text{ref,k}}\big)$.

Step 4. Given $Y^{\text{mpc,k}}$, find the variation of the 'internal' control points in the parameterisation (10), denoted $\hat{P}^{\text{mpc,k}}$, that gives a reference trajectory that is closest in a least-squares sense to $Y^{\text{mpc,k}}$. Namely,

$$\hat{P}_j^{\text{mpc,k}} = \big((G_0\rho)^{\text{T}} G_0 \rho\big)^{-1} \big(G_0\rho\big)^{\text{T}} \big(Y_j^{\text{mpc,k}} - G_0 P_j^{\text{ref,k}}\big). \tag{19}$$

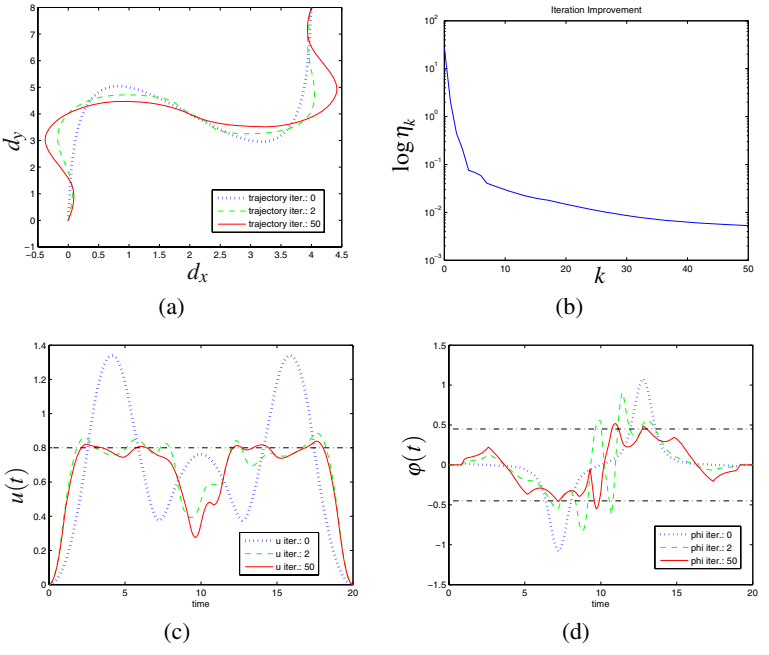

Fig. 1 Initial reference trajectory, 2nd and 50th iteration. (a) Flat output $y = \big(d_x, d_y\big)$; (b) Measure of convergence η_k; (c) Input $u(t)$; and, (d) Input $\varphi(t)$

Step 5. Update the control points according to: $P^{\text{ref,k+1}} = P^{\text{ref,k}} + \rho \hat{P}^{\text{mpc,k}}$.

Step 6. While (a weighted 2-norm of) the difference $\big(P^{\text{ref,k+1}} - P^{\text{ref,k}}\big)$ is larger than a prescribed tolerance level and the maximum number of iterations is not reached: assign $P^{\text{ref,k}} \leftarrow P^{\text{ref,k+1}}$ and go to Step 1.

Note from Steps 1–5 that the proposed algorithm implements a recursion $P^{\text{ref,k+1}} = F\big(P^{\text{ref,k}}\big)$, whose complexity depends predominantly on the (in general, nonlinear) mapping $Y^{\text{mpc,k}} = G\big(P^{\text{ref,k}}\big)$. The convergence properties of the recursive mapping, $P^{\text{ref,k+1}} = F\big(P^{\text{ref,k}}\big)$, will be investigated in future work.

4 Simulation Example

In this section we will test the previous algorithm on a classical example of a flat system, a nonholonomic car system. The system is modeled by the equations: $\dot{d}_x(t) = u(t)\cos\theta(t)$, $\dot{d}_y(t) = u(t)\sin\theta(t)$ and $\dot{\theta}(t) = (1/l)u(t)\tan\varphi(t)$, where the state $d_x(t)$ is the displacement in the "x-direction", the state $d_y(t)$ is the displacement in the "y-direction", the state $\theta(t)$ is the angle of the car with respect to the x-axis, the input $u(t)$ is the velocity of the car, the input $\varphi(t)$ is the angle of the steering wheels, and l is the distance between the front and the rear wheels. It is straightforward to determine that the flat output for this system is given by $y(t) = \big(d_x(t), d_y(t)\big)$.

A matrix of initial control points, $P^{\text{ref},0}$, is chosen so that, together with the parameterisation (12) using cubic B-splines $\lambda_i(t)$, gives the initial reference trajectory $y^{\text{ref},0}$ shown with a dotted line in Figure 1(a). The control inputs are assumed to be subject to the constraints $u(t) \leq 0.8$ and $|\varphi| \leq 0.45$. The inputs corresponding to the initial reference trajectory $y^{\text{ref},0}$ are shown with dotted lines in Figures 1(c) and (d), far exceeding the constraint limits. The result after 2, respectively 50, iterations of the algorithm is shown in Figures 1(a), 1(c), and 1(d) with dashed, respectively solid, lines. Notice that the algorithm produces a final reference trajectory which is close to the initial reference trajectory and with associated inputs only mildly exceeding the constraints. In addition, the initial and final end-point conditions are maintained. A measure of convergence of the algorithm, $\eta_k = \sum_{j=1}^{m} (P_j^{\text{ref,k}} - P_j^{\text{ref,k-1}})^{\text{T}} G_0^{\text{T}} G_0 (P_j^{\text{ref,k}} - P_j^{\text{ref,k-1}})$, is shown in Figure 1(b).

5 Conclusion

A novel methodology combining the differential flatness formalism for *trajectory generation* of nonlinear systems, and the use of a model predictive control strategy for *constraint handling* has been proposed. The methodology consists of a *trajectory generator* that generates a reference trajectory parameterised by splines, and with the property that it satisfies performance objectives. The reference trajectory is generated *iteratively* in accordance with information received from the MPC formulation. The performance of the iterative scheme has been illustrated with a simulation example. Future work will focus on investigating the conditions required to establish the convergence of the iterative algorithm, and on evaluating its computational performance for real-time applications.

References

1. Fliess, M., Lévine, J., Martin, P., Rouchon, P.: Flatness and defect of non-linear systems: Introductory theory and examples. International Journal of Control 61, 1327–1361 (1995)
2. Piegl, L., Tiller, W.: The NURBS Book. Springer, Heidelberg (1997)
3. Flores, M.E., Milam, M.B.: Trajectory generation for differentially flat systems via NURBS basis functions with obstacle avoidance. In: 2006 American Control Conference, Minneapolis, USA (2006)
4. Goodwin, G.C., Seron, M., De Doná, J.A.: Constrained Control and Estimation: An Optimisation Approach. Communications and Control Engineering Series. Springer, Heidelberg (2005)