
Available from: http://dx.doi.org/10.1080/00207179.2011.618514
RESEARCH ARTICLE

Minimum-time Trajectory Generation for Constrained Linear Systems Using Flatness and B-Splines

Fajar Suryawan\textsuperscript{a}, José De Doná\textsuperscript{a}, María Seron\textsuperscript{a}

\textsuperscript{a}Centre for Complex Dynamic Systems and Control (CDSC), University of Newcastle, Australia.

(Received 00 Month 200x; final version received 00 Month 200x)

In this paper we discuss minimum-time trajectory generation for input-and-state constrained continuous-time LTI systems in the light of the notion of flatness and B-splines parameterisation. Flat systems have the useful property that the input and the state trajectories can be completely characterised by the so-called flat output. We propose a splines parameterisation for the flat output, and the corresponding parameterisations for the performance output, the states, and the inputs. Using this parameterisation the problem of minimum-time constrained trajectory planning is cast into a feasibility-search problem in the splines control-point space, in which the constraint region is characterised by a polytope. A close approximation of the minimum-time trajectory is obtained by systematically searching the end-time that makes the constraint polytope to be minimally feasible.

Keywords: Minimum-time control, input constraints, state constraints, B-splines, differential flatness, trajectory generation.

1 Introduction

In some control systems certain tasks are required to be performed as fast as possible under a number of constraints. This problem, often referred to as time-optimal control or minimum-time control, has been a long standing problem in systems and control literature, as well as in applied mathematics. The problem can be traced back to, e.g., the work of Bellman et al. (1956). Despite the inherently interesting nature of the problem, analytical solutions are often very complex, even for low dimensional linear systems. In this regard, there are only very few treatments in the literature dealing with relatively complex problems.

Our approach to minimum-time control here stems from our previous work (Suryawan et al. 2010, 2011), where, using differential flatness and B-splines, every signal and constraint are mapped to the control-point space of B-splines. The constraints form a polytope in this space whose shape changes as the end-time of the parameterisation is varied. This fact is exploited to search for a polytope that is minimally feasible, at which point a minimum time is reached. Other works that employ flatness and splines in connection to optimal control problems, albeit from a different perspective, can be found in, e.g., Milam et al. (2000). Connections between optimal control and flatness-based control have been discussed in Fliess and Marquez (2000) and in Sira-Ramírez and Agrawal (2004).

It is well known that for linear systems constrained only on the input, the resulting time-optimal control solution is bang-bang (see, for example, Kirk 1970). For these systems, the method proposed here is sub-optimal compared to bang-bang control. However, advantages of the method include: 1) the ability to specify initial and final conditions, including the derivatives, of the inputs, states, and outputs, 2) the ability to naturally deal with constraints, including the derivatives, of inputs, states, and outputs (even non-symmetric ones), 3) the ability to naturally
deal with non-minimum phase and unstable systems, 4) the absence of intersampling issues (since no discretisation is involved), 5) the increased smoothness of the resulting signals due to the use of splines.

The paper is organised as follows. In Section 2, the problem is formulated. Section 3 briefly reviews the notion of flatness. B-splines concepts are reviewed in Section 4. B-splines are then used to parameterise flat outputs, as explained in Section 5. Section 6 discusses quadratic optimal control using flatness and B-splines parameterisation, to explain the idea of optimisation in the control-points space. The concept is then extended to minimum-time control in Section 7. A number of numerical examples are presented in Section 8 (for SISO systems) and 9 (for MIMO systems), which includes an input constrained system, an input-and-state constrained system, a non-minimum phase system with output constraint, an unstable system, and a non-minimum phase higher order system. In particular we present two examples discussed in recent work by Consolini and Piazzi (2009) on generalised bang-bang control and compare the results. Experimental results using a laboratory-scale magnetic levitation system are presented in Section 10. In Section 11 the trade-off between accuracy of the solution and computational time is discussed. Finally, conclusions and future research directions are drawn in Section 12.

2 Problem Formulation

Consider a controllable linear system

\[ \dot{x}(t) = Ax(t) + Bu(t), \]
\[ z(t) = Cx(t) \] (1)

with \( x(t) \in \mathbb{R}^n \), \( A \in \mathbb{R}^{n \times n} \), \( B \in \mathbb{R}^{n \times m} \), \( u(t) \in \mathbb{R}^m \), \( z(t) \in \mathbb{R}^m \). With a specified initial output \( z_0 \) together with a reachable target output \( z_f \), the time-optimal control problem is: find \( u(t) \) (that is, generate a trajectory for \( u(t) \)) such that the following cost function is minimised

\[ J_{TOC} = \int_{t_0}^{t_0+T} 1 \, dt \] (2)

subject to

\[ \overline{C}(s) \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \leq \overline{\tau}, \quad \tilde{C}_0(s) \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \bigg|_{t=t_0} = c_0, \quad \tilde{C}_f(s) \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \bigg|_{t=t_0+T} = c_f \] (3)

where \( \overline{C}(s) \), \( \tilde{C}_0(s) \), and \( \tilde{C}_f(s) \) are polynomial matrices in \( s \) (where \( s \) is the differential operator) with \( n + m \) columns, and \( \overline{\tau} \), \( c_0 \) and \( c_f \) are vectors. The inequality relationships are element-wise. The equality constraints in (3) include the desired initial and final values for the output \( z(t) \), but they can also include other initial and final constraints for states and inputs. The following ones are imposed for the majority of the examples presented in this paper:

\[ \dot{z}(t_0) = 0 \quad \text{and} \quad \dot{z}(t_0 + T) = 0. \] (4)

3 Review of Flatness

Differential flatness (Fliess et al. 1995, Lévine 2009, Sira-Ramírez and Agrawal 2004) 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 variables, called flat outputs, and a finite number of their time derivatives. Consider a general system

$$\dot{x}(t) = f(x(t), u(t)), \quad (5)$$

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, we can write all trajectories $(x(t), u(t))$ satisfying the differential equation in terms of the flat output $y(t) \in \mathbb{R}^m$ as follows:

$$x(t) = \Upsilon(y(t), \dot{y}(t), \ddot{y}(t), \ldots, y^{(r)}(t)), \quad u(t) = \Psi(y(t), \dot{y}(t), \ddot{y}(t), \ldots, y^{(r+1)}(t)). \quad (6)$$

In the case of linear systems, a system is flat if and only if it is controllable (Fliess et al. 1995, Lévine 2009, Sira-Ramírez and Agrawal 2004). For a linear flat system model, there exists a flat output $y \in \mathbb{R}^m$ such that

$$x_i = \sum_{j=1}^{m} \sum_{k=0}^{r} \alpha_{i,j,k} y_j^{(k)}, \quad i = 1, \ldots, n, \quad (7)$$

$$u_i = \sum_{j=1}^{m} \sum_{k=0}^{r+1} \beta_{i,j,k} y_j^{(k)}, \quad i = 1, \ldots, m,$$

for some real parameters $\alpha_{i,j,k}$ and $\beta_{i,j,k}$. Here, $r = \max\{q_j \mid j = 1, \ldots, m\}$, where, $q_j + 1$ is the highest derivative required for the $j$-th flat output.

4 Overview of B-splines

We briefly describe below some properties of B-Splines. A $d$-th degree B-spline curve $y(t)$ is a piecewise polynomial function described by

$$y(t) = \sum_{i=0}^{N} \lambda_{i,d}(t) P_i \triangleq \Lambda_{d}(t) P : \quad t \in [t_0, t_f], \quad (8)$$

where $P = [P_0 \ldots P_N]^T$, $P_i$ are the control points, $\Lambda_{d}(t) = [\lambda_{0,d}(t) \ldots \lambda_{N,d}(t)]$, and $\{\lambda_{i,d}(t), i = 0, \ldots, N\}$ are piecewise polynomial functions forming a basis for the vector space of all piecewise polynomial functions of the desired degree $d$ and continuity (for the notion of continuity involved here, see the first item in the list of B-spline properties mentioned later in this section). The basis functions are defined on the nonperiodic, nondecreasing, and nonuniform knot vector

$$V = \{\underbrace{\tau_0, \ldots, \tau_0}_{d+1}, \tau_{d+1}, \ldots, \tau_{v-d-1}, \underbrace{\tau_f, \ldots, \tau_f}_{d+1}\} \quad (9)$$

where $v+1$ is the number of elements in the knot vector and $\tau_{d+1}, \ldots, \tau_{v-d-1}$ are chosen following a certain rule. For example, as can be seen in the above equation, the leftmost and rightmost breakpoints\footnote{A breakpoint is defined as a distinct knot without considering multiplicity. For example, the knot vector $V = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$ has breakpoints $\{0, 1, 2, 3, 4\}$ where breakpoint “0” has multiplicity 3, breakpoint “1” has multi-} are repeated $d+1$ times. Two knot vectors are said to be internally similar if they
have the same elements except for the leftmost and the rightmost breakpoints, which differ in their multiplicities. The relation between \( v \), degree \( d \) and \( N \) is \( v = N + d + 1 \).

The \( i \)-th B-spline basis function of degree \( d \), denoted by \( \lambda_i,d(t) \), is defined recursively as (see de Boor 1978)

\[
\lambda_i,0(t) = \begin{cases} 
1 & \tau_i \leq t < \tau_{i+1} \\
0 & \text{otherwise}
\end{cases}
\]

\[
\lambda_i,d(t) = \frac{t - \tau_i}{\tau_{i+d} - \tau_i} \lambda_{i,d-1}(t) + \frac{\tau_{i+d+1} - t}{\tau_{i+d+1} - \tau_{i+1}} \lambda_{i+1,d-1}(t). 
\]

The following are some useful properties of B-Splines (Piegl and Tiller 1997).

1. Smoothness and differentiability: a spline curve of order \( d \) is \( C^{d-r} \)-continuous at its breakpoints with multiplicity \( r \) and \( C^1 \)-continuous at any other point.
2. Endpoint interpolation: the first control point coincides with \( y(t_0) \), the last control point coincides with \( y(t_f) \).
3. Convex hull property: the curve is contained in the convex hull of its control polygon. The control polygon is the polygon that has the control points as its vertices. This is a result of the fact that the basis functions in (8) satisfy the partition of unity property; namely, \( 0 \leq \lambda_i,d(t) \leq 1 \), and \( \sum_{i=0}^{N} \lambda_i,d(t) = 1 \), \( \forall t \).

The following are some theorems on B-Splines relevant to our exposition below.

**Theorem 4.1:** The \( r \)-th derivative of \( \Lambda_d(t) \) can be expressed as a linear combination of elements of \( \Lambda_{d-r}(t) \), where \( \Lambda_d(t) \) and \( \Lambda_{d-r}(t) \) are defined over the internally same knot vector. In other words, \( \Lambda_d^{(r)}(t) = \Lambda_{d-r}(t) M_{d,d-r} \), where \( M_{d,d-r} \) is a \((N + 1 - r) \times (N + 1)\) matrix.

**Proof** This is a well known result. See, e.g., Piegl and Tiller (1997).

The following theorem is valid for Bézier representation (a special case of B-splines).

**Theorem 4.2:** A set of Bézier basis functions of certain degree can be represented as a unique linear combination of Bézier basis functions of higher degree. We denote this relationship as \( \Lambda_{d-r}(t) = \Lambda_d(t) L_{d,d-r} \).

**Proof** Every set of Bézier basis functions spans the same space as polynomials of the same degree in power basis form (de Boor 1978). But every lower degree polynomial is a subspace of higher degree polynomials. Hence Bézier basis functions of lower degree can be represented as a linear combination of Bézier basis functions of higher degree.

A variant of the last theorem, applicable to more general B-Splines, can be found in Suryawan et al. (2011).

### 5 Flat-Output Parameterisation by B-splines

We parameterise each component of the flat output \( y_j(t) \), \( j = 1, \ldots, m \), as in (8), that is,

\[
y_j(t) = \sum_{i=0}^{N} \lambda_i,d(t) P_{ij} \triangleq \Lambda_d(t) P_j; \quad t \in [t_0, t_f],
\]

where \( \Lambda_d(t) = [\lambda_0,d(t) \ldots \lambda_N,d(t)] \) and \( P_j = [P_{0j} \ldots P_{Nj}]^T \).
Here $\Lambda_d(t)$ is a vector-valued function of time. Where the context is clear we will sometimes drop the subscripts and simply write $\Lambda(t)$. The flat output derivatives can be obtained by taking the derivatives of $\Lambda_d(t)$. For example, for $\dot{y}_j(t)$ we have

$$
\dot{y}_j(t) = \sum_{i=0}^{N} \dot{\lambda}_{i,d}(t) P_{ij} = \dot{\Lambda}_d(t) P_j.
$$

(12)

Now, using Theorem 4.1, Eq. (12) can be rewritten as $\dot{y}_j(t) = \Lambda_d(t) M_{d,d-1} P_j$. Using Theorem 4.2, in the case of Bezier representation, this can be further compacted as

$$
\dot{y}_j(t) = \Lambda_d(t) L_{d,d-1} M_{d,d-1} P_j = \Lambda_d(t) K_{d,d-1} P_j,
$$

(13)

where $K_{d,d-1} \triangleq L_{d,d-1} M_{d,d-1}$. From (7), (13), and similar expressions for higher derivatives of $y_j(t)$, we have

$$
x_i(t) = \sum_{j=1}^{m} \sum_{k=0}^{r} \alpha_{i,j,k} \Lambda_d(t) K_{d,d-k} P_j \triangleq \sum_{j=1}^{m} \Lambda_d \chi_{i,j} P_j,
$$

(14)

$$
i = 1, \ldots, n,
$$

where $\chi_{i,j} \triangleq \sum_{k=0}^{r} \alpha_{i,j,k} K_{d,d-k}$.

Now, with $N+1$ basis functions and $m$ flat outputs define the following $(N+1)m \times 1$ vector

$$
\bar{P} \triangleq [P_1^T \ldots P_j^T \ldots P_m^T]^T
$$

$$
= [P_{01} \ldots P_{N1} \ldots P_{0j} \ldots P_{Nj} \ldots P_{0m} \ldots P_{Nm}]^T
$$

(15)

which collects the parameters $P_{ij}$ used in (11). Hence (14) can be rewritten as

$$
x_i(t) = \Lambda_d(t) \chi_i \bar{P}, \quad \chi_i \triangleq [\chi_{i,1} \ldots \chi_{i,j} \ldots \chi_{i,m}] .
$$

(16)

In a similar way, we have from (7) that

$$
u_i(t) = \sum_{j=1}^{m} \sum_{k=0}^{r+1} \beta_{i,j,k} \Lambda_d(t) K_{d,d-k} P_j
$$

$$
\triangleq \sum_{j=1}^{m} \Lambda_d(t) \nu_{i,j} P_j \triangleq \Lambda_d(t) \nu_i \bar{P}, \quad i = 1, \ldots, m,
$$

(17)

where $\nu_{i,j} \triangleq \sum_{k=0}^{r+1} \beta_{i,j,k} K_{d,d-k}$ and $\nu_i \triangleq [\nu_{i,1} \ldots \nu_{i,m}]$.

Any performance output $w(t) \in \mathbb{R}^{m_w}$ given by a linear combination of states and inputs (including the output $z(t)$ in (1) as a particular case) can be parameterised in the same way. Indeed, let $w(t) \in \mathbb{R}^{m_w}$ satisfying

$$
w(t) = C_w \begin{pmatrix} x(t) \\ u(t) \end{pmatrix},
$$

(18)

where $C_w = (c_{i,j}), 1 \leq i, \leq m_w, 1 \leq j \leq (n+m)$. Using (16), (17), and (18), we then have, for
\[ w_i = \sum_{j=1}^{n} c_{i,j} x_j + \sum_{j=1}^{m} c_{i,j+n} u_j = \Lambda_d W_i \bar{P}, \quad (19) \]

where \( W_i = \left( \sum_{j=1}^{n} c_{i,j} x_j + \sum_{j=1}^{m} c_{i,j+n} u_j \right) \).

To summarise, we have that all the signals involved are parameterised linearly by \( \bar{P} \). That is,

\[
\begin{align*}
    y_j &= \Lambda_d P_j & j = 1, \ldots, m \quad \text{(flat outputs)} \\
    x_i &= \Lambda_d X_i \bar{P} & i = 1, \ldots, n \quad \text{(states)} \\
    w_i &= \Lambda_d W_i \bar{P} & i = 1, \ldots, m_w \quad \text{(performance outputs)} \\
    u_i &= \Lambda_d U_i \bar{P} & i = 1, \ldots, m \quad \text{(inputs)}.
\end{align*}
\]

(20)

Note that every signal has the same set of basis functions and, thus, each signal is contained in the convex hull of its own control points, namely \( P_j, X_i \bar{P}, W_i \bar{P}, \) and \( U_i \bar{P} \), respectively.

6 Quadratic optimal control

As a prelude to the minimum-time control problem, we first discuss the use of a B-spline parameterisation in trajectory generation for a fixed end-time. This will be one of the ingredients of the minimum-time control strategy discussed in the next section.

The problem we will address in this section is: given system (1), with a specified initial state \( x_0 \) together with a reachable target state \( x_f \), find \( u(t) \) (that is, generate a trajectory for \( u(t) \)) such that the following cost function is minimised

\[
J_{QOC} = \frac{1}{2} (x(t_f) - x_f)^T S (x(t_f) - x_f) + \frac{1}{2} \int_{t_0}^{t_f} \left( x^T(t) Q x(t) + u^T(t) R u(t) \right) dt. \quad (21)
\]

subject to the following inequality constraints in the states and inputs

\[
\overline{C} \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \leq \bar{c}, \quad t \in [t_0, t_f],
\]

(22)

where \( \overline{C} \) is a matrix with \( n + m \) columns and \( \bar{c} \) is a vector. Furthermore, if some points in the state-and-input space are known to be reachable, equality constraints can be added:

\[
D_i \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \bigg|_{t=t^*_i} = d_i, \quad t^*_i \in [t_0, t_f];
\]

(23)

where \( D_i \) is a full row-rank real matrix that has \( n + m \) columns with \( m_{d_i} \leq n + m \) rows, and \( d_i \) is a vector with real elements. The cost function (21) can be re-represented, using (20), as

\[
\bar{J} \triangleq \frac{1}{2} \bar{P}^T (S + \bar{Q} + \bar{R}) \bar{P} + \bar{F}^T \bar{P} \triangleq \frac{1}{2} \bar{P}^T \bar{Q} \bar{P} + \bar{F}^T \bar{P} \quad (24)
\]

as explained below. We start with the matrix \( \bar{Q} \in \mathbb{R}^{(N+1)m \times (N+1)m} \). The state-cost term can be
written as
\[ x(t)^T Q x(t) = [x_1(t) \ x_2(t) \ldots] Q [x_1(t) \ x_2(t) \ldots]. \] (25)

Using the parameterisation (20),
\[ x(t)^T Q x(t) = [(\Lambda(t) \mathcal{X}_1 \mathcal{P})^T \ (\Lambda(t) \mathcal{X}_2 \mathcal{P})^T \ldots] Q \begin{bmatrix} \Lambda(t) \mathcal{X}_1 \mathcal{P} \\ \Lambda(t) \mathcal{X}_2 \mathcal{P} \\ \vdots \end{bmatrix}. \] (26)

By defining \( \mathbf{\Lambda}(t) = \text{diag}(\Lambda(t), \ldots, \Lambda(t)) \), we then have
\[ x(t)^T Q x(t) = \bar{P}^T \begin{bmatrix} \mathcal{X}_1^T \mathcal{X}_2^T \ldots \end{bmatrix} \mathbf{\Lambda}(t) \mathcal{Q} \mathbf{\Lambda}(t) \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 \\ \vdots \end{bmatrix} \bar{P} = \bar{P}^T \mathcal{X}^T \mathbf{\Lambda}(t) Q \mathbf{\Lambda}(t) \mathcal{X} \bar{P} \] (27)

with \( \mathcal{X}^T = [\mathcal{X}_1^T \mathcal{X}_2^T \ldots] \) and \( \mathcal{X} \in \mathbb{R}^{(N+1)n \times (N+1)m} \). Now, defining
\[ \bar{\mathcal{X}} \triangleq \int_{t_0}^{t_f} \mathbf{\Lambda}(t) Q \mathbf{\Lambda}(t) dt \in \mathbb{R}^{(N+1)n \times (N+1)m}, \] (28)
we obtain
\[ \int_{t_0}^{t_f} x(t)^T Q x(t) = \bar{P}^T \left( \mathcal{X}^T \int_{t_0}^{t_f} (\mathbf{\Lambda}(t) Q \mathbf{\Lambda}(t)) dt \mathcal{X} \right) \bar{P} = \bar{P}^T \mathcal{X}^T \bar{\mathcal{X}} \bar{P} \triangleq \bar{P}^T \bar{Q} \bar{P} \] (29)

The matrix \( \bar{R} \), from the input cost, can be computed similarly. The matrices \( \bar{S} \) and \( \bar{F} \), from the final state cost, are also simple to compute, and only involve the last rows of the matrices \( \mathcal{X}_i \)'s (due to Property 2 of B-splines mentioned in Section 4, that the last point of the curve coincides with the last control point). Thus, we define:
\[ \mathcal{X}_i \triangleq \begin{bmatrix} 0 & 0 & \ldots & 0 \end{bmatrix} \mathcal{X}_i, \quad i = 1 \ldots n, \] (30)
that is, \( \mathcal{X}_i \) is the last row of \( \mathcal{X}_i \). Then, since matrix \( \bar{S} \) is symmetric, we have
\[ (x(t_f) - x_f)^T S(x(t_f) - x_f) \]
\[ = ((\bar{P}^T \mathcal{X}_1^T \bar{P}^T \mathcal{X}_2^T \ldots) - [x_{f1} \ x_{f2} \ldots]) S \begin{bmatrix} \mathcal{X}_1 \bar{P} \\ \mathcal{X}_2 \bar{P} \\ \vdots \end{bmatrix} - \begin{bmatrix} x_{f1} \\ x_{f2} \\ \vdots \end{bmatrix} \]
\[ = \bar{P}^T \begin{bmatrix} \mathcal{X}_1^T \mathcal{X}_2^T \ldots \end{bmatrix} S \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 \\ \vdots \end{bmatrix} \bar{P} - 2 [x_{f1} \ x_{f2} \ldots] S \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 \\ \vdots \end{bmatrix} \bar{P} + \bar{V} \]
\[ \triangleq \bar{P}^T \bar{S} \bar{P} + 2 \bar{F}^T \bar{P} + \bar{V}, \] (31)

where the term \( \bar{V} \) is independent of \( \bar{P} \). The overall problem can then be cast as a quadratic
programme where constraints can be naturally added, as follows:

$$\min_{\bar{P}} \frac{1}{2} \bar{P}^T \bar{Q} \bar{P} + F^T \bar{P}$$

subject to

$$L_{eq} \bar{P} = W_{eq}, \quad L_{ineq} \bar{P} \leq W_{ineq}.$$  \hspace{1cm} (32)

The matrices $L_{eq}$ and $L_{ineq}$, and vectors $W_{eq}$ and $W_{ineq}$, are constructed by considering the required constraints in (23) and (22), and translating them to the control points via the relationships in (20). In particular, for inequality constraints, it follows from the convex hull property that, by confining the control points, the corresponding signals will satisfy the same bounds.

A more general cost function can also be used, for example in Suryawan et al. (2010), where one has a reference trajectory for a performance output (and hence a set of reference control points $\bar{P}_{ref}$). In this case the target is to have the trajectory as close as possible to the reference trajectory while respecting the constraints, that is, the cost function will be $(\bar{P} - \bar{P}_{ref})^T \tilde{Q}(\bar{P} - \bar{P}_{ref})$.

The concept of generating a trajectory under constraints is extended to minimum-time control in the following section. The idea is to iteratively decrease the end-time, until the constraint set is not feasible anymore.

7 Minimum-time control

Based on the optimal control strategy developed above, we will develop a numerical solution to time-optimal control in the light of the notions of flatness and B-splines parameterisation. First we need the following theorem.

**Theorem 7.1:** Consider the following relation (cf. Theorem 4.1)

$$\Lambda_d^{(i)}(t) = \Lambda_{d-i}(t) M_{d,d-i,T}, \quad t \in [0, T]$$  \hspace{1cm} (33)

where we have introduced a third subscript to indicate the end-time of the trajectory (or, equivalently, the end-value of the knot vector on which the B-splines $\Lambda_d(t)$ are defined). That is, $M_{d,d-i,T}$ is the same matrix as $M_{d,d-i}$ in Theorem 4.1 but with the dependence on the end-time $T$ explicitly indicated. Now, if the B-spline basis functions support is compressed by a factor of $h$ (that is, the knot vector is divided by $h$), we will have

$$M_{d,d-i,T/h} = h^i M_{d,d-i,T}.$$  \hspace{1cm} (34)

The proof of the above theorem is included in Appendix A.

We will illustrate the implications of this fact on the constraints of a generic state $x_i(t)$. From (14),

$$x_i(t, T^{\text{init}}) = \sum_{j=1}^{m} \Lambda_d(t) \left( \sum_{k=0}^{r} \alpha_{i,j,k} K_{d,d-k} \right) P_j$$

$$= \sum_{j=1}^{m} \Lambda_d(t) \left( \alpha_{i,j,0} K_{d,d-0} + \alpha_{i,j,1} K_{d,d-1} + \ldots \right) P_j$$  \hspace{1cm} (35)

where $t \in [0, T^{\text{init}}]$, $x_i(t, T^{\text{init}})$ is state $i$ as a function of time $t$ and end-time $T^{\text{init}}$, and $K_{d,d-i} =$
Now, if the end-time is contracted as \( T = T^{\text{init}} / h \), we have, using Theorem 7.1,

\[
x_i(t, T^{\text{init}} / h) = \sum_{j=1}^{m} \Lambda_d(t) \left( \alpha_{i,j,0} h^0 K_{d,d} + \alpha_{i,j,1} h^1 K_{d,d-1} + \ldots \right) P_j
\]

\[
\triangleq \sum_{j=1}^{m} \Lambda_d(t) H_{x,i,j}(T) P_j = \Lambda_d(t) \left[ H_{x,1}(T) H_{x,2}(T) \ldots \right] \bar{P} \triangleq \Lambda_d(t) H_{x_i}(T) \bar{P}
\]  

where \( t \in [0, T] \) and \( \bar{P} \) is as in (15). Hence, if the state \( x_i(t) \) is constrained to be within a minimum and a maximum value, we have (using the convex hull property) that we can impose the following bounds on the set of control points in (36):

\[
x_{i,\text{min}} \leq H_{x_i}(T) \bar{P} \leq x_{i,\text{max}}.
\]  

The constraints in the input can be similarly dealt with. Overall, we then have the following formulation

\[
\min_{T, \bar{P}} \quad T
\]

subject to

\[
L_{\text{eq}}(T) \bar{P} = W_{\text{eq}}, \quad L_{\text{ineq}}(T) \bar{P} \leq W_{\text{ineq}}
\]  

where \( L_{\text{ineq}}(T) \) is similar to \( L_{\text{ineq}} \) in (32), but is now a function of the end-time \( T \) as a consequence of (36) and the corresponding equations for the input constraints. The equality constraint matrix \( L_{\text{eq}}(T) \) consists, in general, only of the initial and final condition requirement (for states, inputs, and outputs). It can be seen that the decision variables are \( \bar{P} \) and \( T \).

### 7.1 Iterative procedure

The strategy to obtain a minimum-time trajectory proposed in this paper consists in approximating the end-time of the parameterisation using a binary-search algorithm. In each iteration, the algorithm checks whether the constraint set in (38), in the space of control points \( \bar{P} \), is empty or not. If it is not empty (which means that the optimisation problem is feasible), then the end-time is decreased. The procedure is summarised in the following algorithm.

**Algorithm 1:**

Binary-search computation of minimum-time \( T \).

1. **Step 1** Begin with a feasible \( T^{\text{init}} \). Assign \( T_{\text{low}} = 0 \). Assign \( T_{\text{high}} = T^{\text{init}} \).
2. **Step 2** Assign \( T = (T_{\text{high}} - T_{\text{low}}) / 2 \)
3. **Step 3** Check for feasibility, using linear programming:

\[
\min \quad 0
\]

subject to

\[
L_{\text{eq}}(T) \bar{P} = W_{\text{eq}}, \quad L_{\text{ineq}}(T) \bar{P} \leq W_{\text{ineq}}
\]

4. **Step 4** If feasible, assign \( T_{\text{high}} = T \). If infeasible, assign \( T_{\text{low}} = T \)
5. **Step 5** Repeat from Step 2 until stopping criteria.
6. **Step 6** Assign \( T^{\text{final}} = T \). Assign \( \bar{P}^{\text{final}} = \bar{P} \)

**Remark 1:** The zero cost function is used to test the feasibility of the constraint region in
the $\bar{P}$ space corresponding to the value of $T$ in the current iteration. It means that any feasible value is similarly good. Mathematically, it will produce plus infinity if infeasible, and will produce zero if feasible. In this way, the minimum-time is successively approximated, and $\bar{P}$ converges to a single point, a process that has a straightforward geometric interpretation, as explained in the following subsection. In the examples presented in the foregoing Sections 8–11, the same results have been obtained, as expected, with a zero cost, linear cost, quadratic cost, etc., in Step 3 of the algorithm. The complexity of the algorithm is related to, (i) the complexity of the binary search procedure, whose number of iterations is $\log_2(N_s)$, where $N_s$ is the number of partitions of the search space determining the desired precision of the approximation (see, e.g., Johnsonbaugh 2008), (ii) the complexity of the Linear Programming routine in Step 3, which is well known to be polynomial time (see, e.g., Karmarkar 1984), and (iii) the number of control points (see Remark 2 below).

Remark 2: The minimum degree for the B-Spline functions is the highest derivative required for the flat output in the constraints plus one. For example, if the highest required derivative is 4, that is, $y^{(4)}_j$, for some $j$, then the minimum degree is 5. Higher degrees will add further flexibility and smoothness to the signals. In the numerical examples presented in Sections 8 and 9 below, we have used this minimum degree or added one or two more degrees. In addition, the larger the number of control points in the representation, the more flexible the curves are (around 40 is usually adequate with a reasonable computation time). In general, there are several factors that have to be taken into account in determining the degree and the number of control points required for a particular problem. Some of them are listed below:

- the order of the system,
- the number of constraints,
- the severity of the constraints,
- the number of inputs,
- the location of the poles and zeros of the system.

As expected, more control points lead to a more complex optimisation. Furthermore, in higher order systems, the signals that require, in their parameterisation, the flat outputs’ higher derivatives (usually the inputs) tend to be more conservative (viewed from the point of view of their control points and the constraints imposed on them). In Section 11 we will illustrate, with two numerical examples, the tradeoff—between computational performance of the algorithm versus accuracy of the solution—associated to the choice of the number of control points.

7.2 Geometric interpretation of the iterative procedure

The proposed approach has an intuitive geometric interpretation. As explained before, the inequality constraints typically define a polytope. When the end-time $T$ of the trajectory is decreased, the polyhedral constraint set, defined by $L_{ineq}(T)\bar{P} \leq W_{ineq}$, changes shape. The “last feasible” point $\bar{P}^{final}$ corresponds to the unique point of contact, corresponding to $T = T^{final}$, when the boundary of that polyhedral set intersects with the hyperplane defined by the equality constraints, i.e., $L_{eq}(T^{final})\bar{P} = W_{eq}$. At that ultimate contact point, the minimum time is achieved. The simple example below illustrates the idea.

Consider the system $\dot{y} + y = u$, $u \in [-2, 5]$. The variable $y$ is required to move from $y(0) = 0 = P_0$ to $y(t_f) = 2.5 = P_1$ as fast as possible. For simplicity of illustration, and to be able to visualise the geometric objects, we will use only two control points, hence the equality constraints become the single point $(P_0, P_1) = (0, 2.5)$ in the control-point space (see Fig. 1(a)). Note in Fig. 1(a) that the constraint polytope (shown for final times of 1, 0.667, and 0.5 seconds) changes shape. For this problem, the minimum time is $t_f = 0.5$ seconds, which corresponds to the time when

---

\[ T^{final} = 1 \]

This implies that if one starts with $T^{final} = 1$, for example, then in 10 iterations the algorithm will reach a resolution of $2^{-10}$, or about up to three decimal places.
Figure 1. Geometric interpretation of Algorithm 1.

the boundary of the constraint polytope touches the point $(0, 2.5)$. For end-times smaller than 0.5 seconds, there are no more intersections between the constraint polytope (corresponding to inequality constraints) and the single point (corresponding to the equality constraint), which means that for those end-times the problem is infeasible. Figure 1(b) shows the corresponding output $y(t)$ trajectory as a function of time from $y(0) = 0$ to $y(t_f) = 2.5$ for different end-times.

8 Single-input single-output examples

To illustrate the method proposed, we draw a number of representative SISO examples: an input constrained system, an input-and-state constrained system, a non-minimum phase system with output constraint, an unstable system, and a higher-order non-minimum phase system. In particular we present two systems discussed in Consolini and Piazzi (2009), and compare the results.

8.1 A mass-damper system

In the first three examples we consider a system of the form

$$u(t) = a_2 \ddot{x}(t) + a_1 \dot{x}(t) + a_0 x(t), \quad t \in [0, T]. \quad (39)$$

The flat output is chosen as $y(t) \triangleq x(t)$. For this first example, we set $a_2 = 0.1$, $a_1 = 0.2$, $a_0 = 0$; that is, a mass-damper system. The input is constrained non-symmetrically as $-11 \leq u(t) \leq 20$. The initial and final conditions are $x(0) = 0$, $\dot{x}(0) = 0$, $x(T) = 15$, $\dot{x}(T) = 0$. After applying the procedure, using B-splines of degree 4 and 30 control points, the result is depicted in Fig. 2. The end-time is $T_{\text{final}} = 0.636$. As one can see, the input trajectory resembles the bang-bang control solution, in that it first pushes forward as hard as it can and then it pushes backward as hard as it can. Due to the smooth nature of the B-splines, the resulting input trajectory is not a discontinuous curve (as one would have with bang-bang control). In this sense the solution is sub-optimal compared to bang-bang control. However there are advantages in the method, as illustrated in the examples that follow.

8.2 A speed-constrained mass-damper system

Here, we consider a similar system as in the first example. But now, in addition to the input constraints and the equality constraints, the speed is constrained to $\dot{x}(t) \leq 24$. The result is depicted in Fig. 3. The resulting end-time is $T_{\text{final}} = 0.797$. Note that the force is initially at
full strength as in the first example, but since the speed has to be kept below 24, the force is reduced to a value such that the speed stays at the boundary.

### 8.3 A mass-spring-damper system

In this example, we consider the same constraints as in the example of Section 8.2, but now we set $a_0 = 1.2$. That is, it is now a mass-spring-damper system. As in the example of Section 8.2, the speed is limited to $\dot{x}(t) \leq 24$, so that the force has to be adjusted such that the speed is kept below the constraint. Figure 4 shows that, to do so, the force takes a ramp trajectory shape (to maintain constant speed, since there is a spring). After a while, the force pulls back to smoothly reach the target with zero speed.

### 8.4 A rest-to-rest second-order system

In this example we consider a system discussed by Consolini and Piazzi (2009), Example 1, with

$$\frac{Z(s)}{U(s)} = \frac{10(s + 2)}{(s + 1)^2 + 9} \triangleq H(s),$$

where $u(t) \in [-1.8, 1.8]$, $z(t) \in [-0.1, 3.1]$, $z(0) = 0$, $z(T) = 3$, $u(0) = 0$. At the end-time $T$ it is required that only the zeroth-order mode be active (steady-state gain). That is, $z(T) = H(0)u(T)$. Hence $u(T) = 3/2$. Using the controller canonical form of the state-space
The flat output can be chosen as \( y(t) \) (since the system (41) is already in controller canonical form). Then we have that all the other signals can be represented by the flat output as follows (subsequent examples use an identical parameterisation, only differing in the coefficients):

\[
\begin{align*}
  x_1(t) &= y(t) \\
  x_2(t) &= \dot{y}(t) \\
  u(t) &= 10y(t) + 2\dot{y}(t) + \ddot{y}(t) \\
  z(t) &= 20y(t) + 10\dot{y}(t)
\end{align*}
\]

The rest-to-rest condition requires \( x_2(T) = 0 \). Also, we impose \( x_1(0) = x_2(0) = 0 \) (similar initial state requirements are also imposed in the subsequent examples). After applying the method using 52 control points, minimal-time trajectories are obtained, and are depicted in Fig. 5. The resulting end-time is \( T_{\text{final}} = 1.885 \) seconds. This is essentially the same result (here it is marginally faster) as the one obtained with the generalised bang-bang control presented in Consolini and Piazzi (2009). Moreover, our solution has a smoother input trajectory and a truly rest-to-rest output trajectory, as can be seen in Fig. 5(b), where the output’s derivative has zero initial and final values.

8.5 A second-order non-minimum phase system

In this example we consider a non-minimum phase system

\[
\frac{Z(s)}{U(s)} = \frac{10s - 20}{s^2 + 2s + 10},
\]

where the input is constrained as \( u(t) \in [-1.8, 1.8] \), and \( z(0) = 0, z(T) = 3, \dot{z}(0) = 0, \dot{z}(T) = 0 \). The treatment is similar as in the example of Section 8.4 and the resulting trajectories are depicted in Fig. 6.
8.6 An undershoot-constrained system

We consider the same non-minimum phase system, with the same constraints, as in the previous subsection. In addition, we now restrict the maximum undershoot as $z(t) \geq -1$, which induces a linear state constraint. The resulting trajectories are shown in Fig. 7. The output constraint is satisfied, which is also shown in the phase plot. The dashed line in the phase plot is the output constraint translated to the state space.

8.7 An unstable, non-minimum phase system

In this example we consider a minimum-phase, unstable system

$$\frac{Z(s)}{U(s)} = \frac{s - 20}{s^2 + 2s - 10}. \tag{44}$$

where the input is constrained as $u(t) \in [-1.8, 1.8]$, and $z(0) = 0$, $z(T) = 3$, $\dot{z}(0) = 0$, $\dot{z}(T) = 0$. We also constrain the derivative of the output: $\dot{z}(t) \leq 6$. The result is depicted in Fig. 8.
8.8 A rest-to-rest fourth-order system

This last SISO example is again taken from the work by Consolini and Piazzi (2009), Example 2. Consider a fourth order system

\[ \frac{Z(s)}{U(s)} = \frac{10(3.5-s)(s^2+25)}{(s+2)(s+3)(s+4)(s+5)} \triangleq H(s), \quad (45) \]

with \( u(t) \in [-2, 2], z(t) \in [-0.1, 3.1], z(0) = 0, z(T) = 3 \). As in the example of Section 8.4, at the end-time \( T \) it is required that only the zeroth-order mode be active (steady state gain).

That is, \( z(T) = H(0)u(T) \). In the controller canonical form (see the example of Section 8.4), this requirement means that \( x_1(0) = x_2(0) = x_3(0) = x_4(0) = 0 \), and \( x_1(T) = z(T) = 3 \), \( x_2(T) = x_3(T) = x_4(T) = 0 \).

Using B-splines of degree 6 with 80 control points, the result is depicted in Fig. 9. The achieved end-time is \( T_{final} = 1.459 \) seconds [c.f., \( T_{final} = 1.382 \) seconds achieved by Consolini and Piazzi (2009)]. The shape of the signals closely resembles the ones obtained by Consolini and Piazzi (2009) using generalised bang-bang control. Note, however, that we do not require discretisation and, moreover, thanks to the use of splines our control signal is smooth.
9 Multiple-input multiple-output examples

Consider the following system described by a transfer matrix (Sira-Ramírez and Agrawal 2004, page 75):

\[
\begin{pmatrix}
    z_1 \\
    z_2
\end{pmatrix} = G(s) \begin{pmatrix}
    u_1 \\
    u_2
\end{pmatrix} = \begin{pmatrix}
    1/1+s & 1/1+2s \\
    1/1+2s & 1/1+s
\end{pmatrix} \begin{pmatrix}
    u_1 \\
    u_2
\end{pmatrix}.
\] (46)

A choice of flat output can be obtained after the above transfer matrix is represented in a state space form as shown below. Using controller canonical form for each column, we have

\[
\dot{x}(t) = \begin{bmatrix}
    0 & 1 & 0 & 0 \\
    -0.5 & -1.5 & 0 & 0 \\
    0 & 0 & 0 & 1 \\
    0 & 0 & -0.5 & -1.5
\end{bmatrix} x(t) + \begin{bmatrix}
    0 & 0 \\
    1 & 0 \\
    0 & 0 \\
    0 & 1
\end{bmatrix} u(t) \tag{47}
\]

\[z(t) = \begin{bmatrix}
    0.5 & 1 & 0.5 & 0.5 \\
    0.5 & 0.5 & 0.5 & 1
\end{bmatrix} x(t) \]

where

\[
x(t) = \begin{bmatrix}
    x_1(t) \\
    x_2(t) \\
    x_3(t) \\
    x_4(t)
\end{bmatrix}, \quad u(t) = \begin{bmatrix}
    u_1(t) \\
    u_2(t)
\end{bmatrix}, \quad z(t) = \begin{bmatrix}
    z_1(t) \\
    z_2(t)
\end{bmatrix}. \tag{48}
\]

The flat output is given by \(y_1(t) = x_1(t)\), and \(y_2(t) = x_2(t)\). Then all variables in the system can be parameterised as follows:

\[
x_1(t) = y_1(t), \quad x_2(t) = \ddot{y}_1(t), \quad x_3(t) = y_2(t), \quad x_4(t) = \dot{y}_2(t),
\]

\[
u_1(t) = \frac{1}{2} y_1(t) + \frac{3}{2} \dot{y}_1(t) + \ddot{y}_1(t)
\]

\[
u_2(t) = \frac{1}{2} y_2(t) + \frac{3}{2} \dot{y}_2(t) + \ddot{y}_2(t) \tag{49}
\]

\[
z_1(t) = \frac{1}{2} y_1(t) + \dot{y}_1(t) + \frac{1}{2} y_2(t) + \frac{1}{2} \dot{y}_2(t)
\]

\[
z_2(t) = \frac{1}{2} y_1(t) + \frac{1}{2} \dot{y}_1(t) + \frac{1}{2} y_2(t) + \dot{y}_2(t)
\]
In the scenarios that follow, B-splines of degree 4 with 40 control points are used.

9.1 A MIMO system with constrained inputs

Suppose now that the system is required to move the output from \( z(0) = [0; 0]^T \) to \( z(T_{\text{final}}) = [3; 5]^T \) in minimum-time, subject to constraints: \( u_1(t) \in [-3, 5, 0] \) and \( u_2(t) \in [-2, 4, 2, 7] \). Translating the inequalities to the flat output space using (49), and applying the method proposed, the result is depicted in Fig. 10. One can see that the inputs are pushed to the limits. Note also that the “switching times” are different between input \( u_1 \) and input \( u_2 \).

9.2 A MIMO system with constrained input and output

Figure 11 shows the same example, but, in addition, the outputs are constrained as \( z_1(t) \leq 3.7 \) and \( z_2(t) \leq 5.4 \).

9.3 An unstable MIMO system with non-minimum phase entries in the transfer matrix and with input inequality constraints

Consider an unstable system described by the following transfer matrix

\[
G(s) = \begin{pmatrix}
\frac{2s - 20}{s^2 + 2s + 10} & \frac{s + 12}{s^2 - 2s + 10} \\
\frac{s + 10}{s^2 + 2s + 10} & \frac{2s - 8}{s^2 - 2s + 10}
\end{pmatrix}
= \begin{pmatrix}
\frac{2s - 20}{s + 10} & \frac{s + 12}{s^2 + 2s + 10} \\
\frac{2s}{s^2 - 2s + 10} & \frac{8}{s^2 + 2s + 10}
\end{pmatrix}.
\] (50)

In state-space form the system can be written as,

\[
\dot{x}(t) = \begin{bmatrix}
0 & 1 & 0 & 0 \\
-10 & -2 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & -10 & 2
\end{bmatrix} x(t) + \begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix} u(t)
\]

\[
z(t) = \begin{bmatrix}
-20 & 12 & 1 \\
10 & 1 & -8 & 2
\end{bmatrix} x(t).
\] (51)

The inputs are constrained as \( u_1(t) \in [-2.3, 4.8] \) and \( u_2(t) \in [-3, 2.7] \). The system is required to move the output from \( z(0) = [0, 0]^T \) to \( z(T_{\text{final}}) = [3, 5]^T \). All initial states are zero. After applying the method, the result is depicted in Fig. 12.

9.4 An unstable MIMO system with non-minimum phase entries in the transfer matrix, with input and output inequality constraints and output equality constraints

Consider the same system, but now the output is further required to have zero derivative at the end-time. Furthermore, we limit the second component of the output’s undershoot as \( z_2(t) \geq -0.8 \). The result is depicted in Fig. 13. It can be seen that the undershoot in \( z_2 \) is reduced, but at the expense of more undershoot in \( z_1 \).

10 Experimental laboratory example

In this section we apply the proposed method to a real plant, a laboratory-scale magnetic levitation system. We will describe the model of the plant, including the flatness parameterisation,
the trajectory generation problem, and the experimental results.

10.1 Plant Description

The Quanser MAGLEV system Quanser Inc. (2006) is an electromagnetic suspension system acting on a solid one-inch steel ball. It consists of an electromagnet, which can lift the ball from a post and sustain it in the air by counteracting the ball’s weight with the electromagnetic force. As illustrated in Fig. 14, the positive direction of vertical displacement is downwards, with the origin of the global Cartesian frame of coordinates on the electromagnetic core flat face. Only
Figure 11. Example of Section 9.2. In each of the left figures (Fig. (a), (c), (e)), the thick lines correspond to the first components. (a) time versus flat outputs $y_1(t)$ and $y_2(t)$. (b) $y_1(t)$ vs $y_2(t)$. (c) time versus inputs $u_1(t)$ and $u_2(t)$. (d) $u_1(t)$ vs $u_2(t)$. (e) time versus outputs $z_1(t)$ and $z_2(t)$. (f) $z_1(t)$ vs $z_2(t)$ (also shown with red dashed lines are the output constraints). Here, $T_f = 6,974$ seconds.

Define the vector of state variables as $\bar{x} \triangleq (\bar{x}_1, \bar{x}_2)$, where $\bar{x}_1 = x_b$ is the ball position, $\bar{x}_2 = \dot{x}_b$ is the ball velocity. A nonlinear model for the Maglev system is

$$\begin{align*}
\dot{x}_1(t) &= \bar{x}_2(t) \\
\dot{x}_2(t) &= -\frac{K_m}{2M_b} \bar{z}(t) + g
\end{align*}$$

(52)
Figure 12. Example of Section 9.3. (a) time versus inputs $u_1(t)$ and $u_2(t)$. (b) time versus outputs $z_1(t)$ and $z_2(t)$. Here, $T_{\text{final}} = 0.776$ seconds.

Figure 13. Example of Section 9.4. (a) time versus inputs $u_1(t)$ and $u_2(t)$. (b) time versus outputs $z_1(t)$ and $z_2(t)$. Here, $T_{\text{final}} = 0.809$ seconds.

Figure 14. Diagram of the magnetic levitation system

where $\bar{u} = I_c$ is the coil current, regarded as the control input. From Quanser Inc. (2006) and system identification, the parameter values are $M_b = 0.068$ Kg, $a = 7.7$ mm, $K_m = 2.64 \times 10^{-4}$ N$m^2/A^2$.

To obtain a linearised model of the Maglev, we linearise system (52) around a nominal operating point. A static equilibrium at a point $x_{eq} = (x_{b,eq}, 0)$ is characterised by the ball being suspended in air at a constant position $x_{b,eq}$ due to a constant electromagnetic force generated by
Using Taylor's series approximation to obtain the linearisation around \((x_{eq}, i_{eq})\), the resulting linear incremental model can be written as the following state space representation:

\[
\begin{align*}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{2g}{(x_{b,eq} + a)} x_1(t) - \frac{\sqrt{2K_m M_b g}}{M_b(x_{b,eq} + a)} u(t),
\end{align*}
\]

where the equilibrium current is

\[
i_{eq} = \sqrt{\frac{2M_b g}{K_m}} (x_{b,eq} + a)
\]

and the state variables \(x = (x_1, x_2)\) and the input variable \(u\) have been defined as the incremental values around the equilibrium point \(x_{eq} = (x_{b,eq}, 0), i_{eq}\), that is,

\[
x_1 = \bar{x}_1 - x_{b,eq}, \quad x_2 = \bar{x}_2, \quad u = \bar{u} - i_{eq},
\]

where \(x_{b,eq} = 6\) mm and \(i_{eq} = 0.97\) A. The resulting linear model is controllable and unstable. From the linearised model in (53), we can choose the flat output as \(y = x_1\). We then have the flatness parameterisation

\[
x_1 = y, \quad x_2 = \dot{y}, \quad u = \frac{i_{eq}}{x_{b,eq} + a} y - \frac{i_{eq}}{2g} \ddot{y}.
\]

### 10.2 Trajectory and constraints specification

In the first scenario the ball is required to travel from position \(4\) mm to \(13\) mm (all are measured from the coil’s core) in the fastest possible way. The ball must be in zeroth-order mode (static gain) at both the initial and final states (this amounts to have zero speed and zero acceleration at the initial and final condition). That is, the desired trajectory is rest-to-rest. Now, denote by \(t = 0\) the initial time and \(t = T\) the final time. Overall, we have to satisfy the following equality constraints

\[
\begin{align*}
x_1(0) + x_{b,eq} &= 4, & x_1(T) + x_{b,eq} &= 13, \\
x_2(0) &= 0, & x_2(T) &= 0,
\end{align*}
\]

and inequality constraints:

\[
\begin{align*}
0 \leq u(t) &\leq 2 \\
-12 \leq \dot{u}(t) &\leq 12 \\
0 \leq x_1(t) + x_{b,eq} &\leq 14 \\
\dot{x}_1(t) &\leq g,
\end{align*}
\]

where \(g\) is the gravity constant. Note that we also impose a constraint on the derivative of the input. Applying the procedure, using 50 control points, we obtain the trajectories depicted in Figs. 15 and 16.

It can be seen from the figures that the trajectories satisfy the constraints; for this case the only active inequality constraint is the rate of change of the current, \(\dot{u}(t)\). The trajectories can be interpreted as follows. The system is initially in zeroth-order mode (i.e., static gain), close to
The coil core is regarded as the zero position, so when the ball travels down to the pedestal, the corresponding curve goes up.
Figure 17. Reference trajectory for position, speed and acceleration of the ball for the second scenario. Here, $T_{final} = 0.1375$ seconds.

Also included in the figures is the solution to the problem using bang-bang control. The solution is constructed using the iterative procedure in De Doná (2000), where, for this problem, the state vector is augmented with the input, so the new (constrained) input —corresponding to the derivative of the original input— is of bang-bang nature (see Fig. 16). We highlight that the exact bang-bang solution can only be obtained for reduced classes of systems (this experimental example being one of them, thus allowing to perform a comparison with the exact solution). The method proposed in this paper, on the other hand, while being suboptimal, allows to obtain solutions for much more general examples, as presented in the previous sections and also in the forthcoming second scenario for the Maglev system (which includes a state constraint).

The second scenario involves an additional constraint in the state,

$$x_2(t) \leq 0.1 \text{ m/s},$$

while all other constraints remain the same as in the first scenario. The resulting trajectory can be seen in Figs. 17 and 18.

10.3 Experimental results

In the real-plant experiment, the plant is computer-controlled using Matlab software and RTX software via Quanser’s data acquisition card with 1000 Hz sampling rate. The closed-loop stabilisation is performed using a state-feedback controller with an integrator in the first state (to allow for trajectory tracking). Figures 19 and 20 show the results for the first and second scenario, respectively.
Figure 18. Reference trajectory for (a) current input and (b) its derivative for the second scenario.

Figure 19. Experimental results for the first scenario. (a) Ball’s position, (b) coil current input. The red thick lines are the reference trajectories from Fig. 15(a) and Fig. 16(a). The thin blue lines are the measurements from the experiment. Notice that in these figures the movement is initiated at 0.2 seconds.

11 Computational performance vs accuracy

The proposed method is suboptimal when compared to the exact time-optimal solution. However, the method allows to obtain approximated solutions for much more general problems for which the actual time-optimal solution cannot be obtained. In this section we compare the minimum-time achieved by our method with that of the bang-bang control solution for a couple of examples where the bang-bang solution can be obtained. We analyse the accuracy of the solution, compared to the optimal bang-bang solution, obtained for different number of control points. The CPU-times required to obtain the flatness-spline solution for each number of control points are also reported. The algorithm was implemented with Matlab 7.1, running on Windows XP SP3, on a computer with an Intel Dual-Core Processor 1.7 GHz with 2 GB of RAM.

Consider first the double-integrator system

\[
\frac{d^2}{dt^2} z(t) = u(t) \iff \frac{Z(s)}{U(s)} = \frac{1}{s^2} \triangleq G(s),
\]

where \(|u(t)| \leq 1\). The system is required to move from \((\text{position}, \text{speed}) = (0, 0)\) to \((10,0)\) as fast as possible. The optimal solution is bang-bang (see, e.g., Athans and Falb 1966, Lee and Markus 1967) with end-time \(T_{\text{bang}} = \sqrt{40}\) seconds and switching-time \(\sqrt{10}\) seconds. Figure 21 shows the resulting trajectories with 20, 30, 40, 50, and 60 control points. The dashed line is the optimal trajectory with bang-bang control. Table 1 shows the comparison of the end-time-approximation
Figure 20. Experimental results for the second scenario. (a) Ball’s position, (b) coil current input. The red thick lines are the reference trajectories from Fig. 17(a) and Fig. 18(a). The thin blue lines are the measurements from the experiment. Notice that in these figures the movement is initiated at 0.2 seconds.

Table 1. Comparison of computational/accuracy performance for different number of control points, for double integrator plant.

<table>
<thead>
<tr>
<th>Control points</th>
<th>End-time difference</th>
<th>CPU-time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.7 %</td>
<td>0.61</td>
</tr>
<tr>
<td>30</td>
<td>0.65 %</td>
<td>1.70</td>
</tr>
<tr>
<td>40</td>
<td>0.34 %</td>
<td>5.10</td>
</tr>
<tr>
<td>50</td>
<td>0.21 %</td>
<td>11.6</td>
</tr>
<tr>
<td>60</td>
<td>0.14 %</td>
<td>18.1</td>
</tr>
</tbody>
</table>

Figure 21. Approximation of bang-bang control input (dashed line) with different number of control points (20, 30, 40, 50, 60) for a double integrator plant. The optimal end-time is $T_{init} = 6.3246$. See also Table 1.

and the CPU-time taken to compute the trajectories. The algorithm starts at $T_{init} = 8$ seconds, and the number of iterations is 20.

The second plant used in the comparison is the maglev system of Sec. 10 with constrained input derivative. Figure 22 shows the resulting trajectories with 20, 30, 40, 50, and 60 control points. Table 2 shows the comparison of the end-time-approximation and the CPU-times taken to compute the trajectories. The bang-bang solution is obtained using an iterative procedure developed in De Doná (2000), where, for this problem, the state-space equation is augmented with the input, so the new (constrained) input corresponds to the derivative of the original input. The algorithm starts at $T_{init} = 16$ seconds, and the number of iterations is 23.

In both cases, as expected, using more control points yields a better approximation to the optimal solution, at the expense of a longer processing time.

12 Conclusions and future research directions

A new method to generate minimum-time trajectories for input, output, and state constrained continuous-time LTI systems was presented. The method is based on the notions of differential
Table 2. Comparison of computational/accuracy performance for different number of control points, for the maglev plant.

<table>
<thead>
<tr>
<th>Control points</th>
<th>End-time difference</th>
<th>CPU-time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>11.5 %</td>
<td>1.04</td>
</tr>
<tr>
<td>30</td>
<td>6.7 %</td>
<td>2.61</td>
</tr>
<tr>
<td>40</td>
<td>4.4 %</td>
<td>6.02</td>
</tr>
<tr>
<td>50</td>
<td>3.0 %</td>
<td>9.85</td>
</tr>
<tr>
<td>60</td>
<td>2.2 %</td>
<td>19.8</td>
</tr>
</tbody>
</table>

Figure 22. Approximation of bang-bang control input (dashed line) for different number of control points (20, 30, 40, 50, 60) for a maglev system. The optimal end-time is 0.1236. See also Table 2.

flatness and B-splines. With the method, one can specify initial and final conditions, including the derivatives of the signals involved. Also, the method can naturally deal with constraints on inputs, states, and outputs, including their derivatives. The several representative examples presented have shown that the method can satisfactorily deal with non-minimum phase and unstable SISO and MIMO systems. The result is sub-optimal but it has the important practical advantage that the signals produced are smoother than those obtained with the bang-bang solution. A discussion of the complexity of the algorithm, and a geometric interpretation, were provided. Lastly, to validate the results experimentally, the method was successfully applied to a real laboratory-scale magnetic levitation system. An analysis of the tradeoff between accuracy of the solution and computational time was provided.

Future work will include extensions of the method to more general systems, e.g., parameter-varying systems and nonlinear systems. Reducing the complexity and conservatism mentioned in Remark 2, while maintaining flexibility and reliability, will also be one of the future research directions.

Acknowledgements

The authors want to thank Professor Jean Lévine from Mines-Paris Tech for very insightful discussions during his visit to Newcastle in 2007, and when the second author spent a study leave period in Fontainebleau in 2008/2009.

References

REFERENCES


Appendix A: Proof of Theorem 7.1

Given the knot vector \( V = \{\tau_0, \ldots, \tau_v\} \), the \( i \)-th B-spline basis function of degree \( d \), denoted by \( \lambda_{i,d}(t) \), is defined recursively as (de Boor 1978):

\[
\lambda_{i,0}(t) = \begin{cases} 1 & \tau_i \leq t < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}
\]

\[
\lambda_{i,d}(t) = \frac{t - \tau_i}{\tau_{i+d} - \tau_i} \lambda_{i,d-1}(t) + \frac{\tau_{i+d+1} - t}{\tau_{i+d+1} - \tau_{i+1}} \lambda_{i+1,d-1}(t),
\]

where \( i = 0 \ldots N \), and \( t \in [\tau_0, \tau_v] \). Without loss of generality we will assume that \( \tau_0 = 0 \). The derivatives of the basis functions are given by (see, e.g., de Boor 1978)

\[
\dot{\lambda}_{i,d}(t) = \frac{d}{\tau_{i+d} - \tau_i} \lambda_{i,d-1}(t) - \frac{d}{\tau_{i+d+1} - \tau_{i+1}} \lambda_{i+1,d-1}(t)
\]

Now, if the end-time is contracted by a factor \( h \), then the knot vector becomes \( \tilde{V} = \{\tilde{\tau}_0, \ldots, \tilde{\tau}_v\} = \{\tau_0/h, \ldots, \tau_v/h\} \). We denote the corresponding basis functions as \( \tilde{\lambda}_{i,d}(\tilde{t}) \), where \( (\tilde{\tau}_v - \tilde{t})/(\tilde{t} - \tilde{\tau}_0) = (\tau_v - t)/(t - \tau_0) \) (this means that \( \tilde{t} \) and \( t \) have the same relative position, respective to their knot vector).
It is easy to see from (A1) that \( \bar{\lambda}_{i,d}(\bar{t}) = \lambda_{i,d}(t) \). Now, the derivative of \( \bar{\lambda}_{i,d}(\bar{t}) \) is

\[
\dot{\bar{\lambda}}_{i,d}(\bar{t}) = \frac{d}{\tau_{i+d} - \tau_i} \bar{\lambda}_{i,d-1}(\bar{t}) - \frac{d}{\tau_{i+d+1} - \tau_{i+1}} \bar{\lambda}_{i+1,d-1}(\bar{t}) = \frac{d}{\tau_{i+d}/h - \tau_i/h} \bar{\lambda}_{i,d-1}(\bar{t}) - \frac{d}{\tau_{i+d+1}/h - \tau_{i+1}/h} \bar{\lambda}_{i+1,d-1}(\bar{t}) = h\dot{\lambda}_{i,d}(\bar{t}).
\]

(A3)

Now, denoting \( \Lambda_d(t) = [\lambda_{0,d}(t) \ldots \lambda_{N,d}(t)] \), and \( \bar{\Lambda}_d(\bar{t}) = [\bar{\lambda}_{0,d}(\bar{t}) \ldots \bar{\lambda}_{N,d}(\bar{t})] \), we have, using Theorem 4.1,

\[
\dot{\Lambda}_d(\bar{t}) = h\dot{\Lambda}_d(t) = \Lambda_{d-1}(t)hM_{d,d-1,\tau_v} = \bar{\Lambda}_{d-1}(\bar{t})M_{d,d-1,\tau_v}/h.
\]

(A4)

So that \( M_{d,d-1,\tau_v}/h = hM_{d,d-1,\tau_v} \). The expressions for higher derivatives can be obtained in a similar way. This concludes the proof.