



Time-Optimal Neural Feedback Control of Nilpotent Systems as a Binary Classification Problem

Sara Bicego¹ · Samuel Gue² · Dante Kalise¹ · Nelly Villamizar²

Received: 17 February 2026 / Revised: 18 May 2026 / Accepted: 19 May 2026
© The Author(s) 2026

Abstract

A computational method for the synthesis of time-optimal feedback control laws for linear nilpotent systems is proposed. The method is based on the use of the bang-bang theorem, which leads to a characterization of the time-optimal trajectory as a parameter-dependent polynomial system for the control switching sequence. A deflated Newton's method is then applied to exhaust all the real roots of the polynomial system. The root-finding procedure is informed by the Hermite quadratic form, which provides a sharp estimate of the number of real roots to be found. In the second part of this paper, the polynomial systems are sampled and solved to generate a synthetic dataset for the construction of a time-optimal deep neural network (NN)—interpreted as a binary classifier—via supervised learning. Numerical tests in integrators of increasing dimension assess the accuracy, robustness, and real-time control capabilities of the approximate control law.

Keywords Time-optimal control · Feedback control · Supervised learning · Deep neural networks (NNs) · Nilpotent systems · Hermite quadratic forms

Mathematics Subject Classification 93B52 · 93C05 · 68T07 · 14P10

1 Introduction

Time-optimal control problems have a prominent place in control theory due to their many applications in robotics, aerospace, and trajectory planning [7, 33, 35]. A classical result dating back to the early days of optimal control, known as the *bang-bang* theorem states

✉ Dante Kalise
dkaliseb@ic.ac.uk

Sara Bicego
sara.bicego@outlook.com

Samuel Gue
samcmgue@gmail.com

Nelly Villamizar
n.y.villamizar@swansea.ac.uk

¹ Department of Mathematics, Imperial College London, London, UK

² Department of Mathematics, Swansea University, Swansea, UK

that, for finite-dimensional, linear time-optimal processes, where the control variable is constrained to a convex, closed, and bounded polyhedron, the optimal control signal is piecewise constant, taking all its values at the vertices of the polyhedron (see the different formulations in, e.g., [4, 21, 30]). Moreover, under additional properties of the state-to-state map, an upper bound on the number of switches of the piecewise-constant control can be determined. Both statements will be made more precise in the next section. From a state space perspective, such a result transforms the time-optimal control synthesis into a geometric problem, that is, the identification of a switching surface splitting the state space into regions, where different vertices of the control polyhedron are selected as the instantaneous optimal control action. Hence, while the original bang-bang theorem follows from first-order optimality conditions for a given initial state, it also provides a characterization of the optimal feedback control law as a piecewise constant map with a finite number of output values. With no loss of generality, in this paper, we restrict our presentation to scalar control signals taking values in $[-1, 1]$, for which the optimal feedback law corresponds to a binary classifier onto $\{-1, 1\}$. We study the construction of such a binary classifier via supervised learning with synthetic data obtained from sampling optimal trajectories, which are parametrized in terms of the switching structure of the optimal control signal.

In the first part of this paper, we follow a similar approach to that in [34]. Here, the authors cast a time-optimal control problem, for which a parametrization of the switching control sequence and time integration of the dynamics leads to a polynomial system of equations for the switching times. Since there is a unique solution to the time-optimal control problem, we only need to determine the existence of a non-negative solution of the polynomial system given a choice of the initial term of the switching control sequence, -1 or 1 . This solvability problem is studied using Gröbner bases, Sturm sequences, and Macaulay2 [18] as an effective computer algebra system for related calculations. Extensions of this work to a more general class of linear dynamical systems have been studied in [27, 28, 31]. A comprehensive review of the use of Gröbner bases in control theory is given in [26]. However, to the best of our knowledge, the computational complexity associated with the use of Gröbner bases for solving polynomial systems such as those arising in time-optimal control hinders their applicability to higher-dimensional systems than those reported in the aforementioned references and makes them unsuitable for real-time feedback control. Our contribution is to propose a methodology that circumvents both issues.

- Instead of analyzing the existence of non-negative solutions to the polynomial system, we apply a variant of Newton's method, which can effectively exhaust all possible roots. We follow a deflation technique as originally proposed in [8] for polynomial systems and later extended to a wider class of problems and applications [6, 9, 16, 22].
- The stopping of the deflation method is determined by the number of roots it is expected to find. We apply Hermite's quadratic form as a mechanism to establish the number of roots of the polynomial systems under study. The applications of quadratic forms in root counting have been explored from as early as 1853 by Sylvester in [32]. The idea of using Hermite quadratic forms to solve polynomial systems with real parameters is discussed in [29] and its use for sign determination in [3, 5, 12]. A recent application of the above to semi-algebraic systems with real parameters can be found in [17].
- A real-time feedback control is built using supervised learning. The block associated with the solution of the polynomial system receives as an input an initial condition, which enters as a parameter in the polynomial system, and outputs the first control value of the optimal switching sequence, which leads to the construction of a binary classifier mapping the n th dimensional state space to $\{-1, 1\}$. Inspired by recent work on

supervised learning for optimal feedback control [1, 2, 11, 14, 19, 24, 25], we train a deep neural network (NN) to approximate the state-to-control feedback map using synthetic data. This approach bypasses the need for iterative recomputation of optimal trajectories, which is replaced by real-time model evaluation.

The rest of this paper is organized as follows. In Sect. 2, we introduce nilpotent linear time-invariant systems with a chain of integrators as a prototypical case, and we provide a characterization of the time-optimal control problem as a polynomial system for the switching times of the control sequence. In Sect. 3, we present the deflated Newton's method that we apply to find all possible roots of the polynomial system. In Sect. 4, we introduce the Hermite quadratic form and its use to determine the existence of non-negative real roots of a system of polynomial equations. In Sect. 5, we discuss the approximation of a binary classifier via supervised learning with deep NNs and synthetic data generated from sampling polynomial systems for different initial conditions. In Sect. 6, we assess the performance and robustness of the proposed approach on chains of integrators up to dimension 5. Section 7 presents concluding remarks and future research directions.

2 Time-Optimal Control of Nilpotent Systems

We study the linear time-optimal control problem

$$\min_{u \in \mathcal{U}} T \quad \text{subject to} \quad \begin{cases} \dot{y}(t) = Ay(t) + bu(t), \\ y(0) = x, \\ y(T) = 0, \end{cases} \quad (1)$$

where $\mathcal{U} \equiv L^\infty([0, +\infty); [-1, 1])$, A is an $n \times n$ real matrix, $b \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$ a given initial condition. We begin by recalling a well-known characterization of the solution to time-optimal control problem (1).

Theorem 1 (Chapter III, [30]) *Assume that Kalman's controllability condition holds for the pair (A, b) , and that all the eigenvalues of A are real. Then, for every initial condition x , there exists a unique time-optimal control signal u . This control signal is bang-bang, i.e., taking values in $\{-1, 1\}$ with at most $n - 1$ switches.*

The result above motivates a parametrization of the total time $T = t_1 + \dots + t_n$ for non-negative t_i 's, which determine the switching times of the control signal. The time domain Ω is the union of subintervals $\Omega_i = [T_{i-1}, T_i)$ with $T_0 = 0$ and $T_i = \sum_{j=1}^i t_j$ for every $1 \leq i \leq n - 1$, and $\Omega_n = [T_{n-1}, T_n]$. This induces a further parametrization of the optimal control signal u as a piecewise-constant function in time, taking the value $u(0) = \pm 1$ in Ω_1 , and $(-1)^{i-1}u(0)$ in the interval Ω_i . When a control has fewer than $n - 1$ switches, this coincides with $t_i = 0$ for some values of $i = 1, \dots, n$. When $t_i = 0$, we define $\Omega_i = \emptyset$. We study the case, where the matrix A is a nilpotent matrix with nilpotency index n . Here, $A = (A_{ij})_{i,j}$ is similar to an $n \times n$ nilpotent Jordan block and hence, without loss of generality, we assume that $A_{ij} = \delta_{i,j-1}$, for $i, j = 1, \dots, n$, where δ is the Kronecker delta. This covers a class of relevant problems, such as n th order integrators of the form

$$\begin{aligned} \dot{y}_1(t) &= y_2(t), \\ &\vdots \\ \dot{y}_{n-1}(t) &= y_n(t), \end{aligned}$$

$$\dot{y}_n(t) = u(t),$$

and allows the following characterization of the optimal control signal.

Proposition 1 *Let A be an $n \times n$ nilpotent Jordan block and $\mathbf{b} \in \mathbb{R}^n$ satisfying the assumptions in Theorem 1. Given an initial condition $\mathbf{x}$, the non-negative increments $\{t_i\}_{i=1}^n$, such that $T = t_1 + \dots + t_n$, associated with the time-optimal control signal $u(t) = u_i$, for all $t \in \Omega_i$, with $u_i = \pm 1$, satisfy the polynomial system:*

$$0 = -u(0) \sum_{k=1}^{n-i+1} b_{k+i-1} \sum_{\substack{\alpha_1, \dots, \alpha_n \geq 0 \\ \alpha_1 + \dots + \alpha_n = k}} (-1)^{\max\{j: \alpha_j \neq 0\}} \frac{t_1^{\alpha_1} \dots t_n^{\alpha_n}}{\alpha_1! \dots \alpha_n!} + \sum_{k=i}^n \frac{T^{k-i}}{(k-i)!} x_k, \quad (2)$$

for $i = 1, \dots, n$, where b_j, x_j denote the coordinates of $\mathbf{b}$ and $\mathbf{x}$, respectively.

Proof First, recall that for a given control signal, the trajectory associated with the linear system in (1) is given by

$$\mathbf{y}(t) = e^{tA} \mathbf{y}(0) + e^{tA} \int_0^t e^{-\tau A} \mathbf{b} u(\tau) d\tau.$$

Evaluating at $T = t_1 + \dots + t_n$, and imposing that $\mathbf{y}(T) = \mathbf{0}$ implies

$$\mathbf{0} = e^{TA} \mathbf{x} + e^{TA} \int_0^T e^{-\tau A} \mathbf{b} u(\tau) d\tau,$$

and integration by parts yields

$$\mathbf{0} = e^{AT} \mathbf{x} + e^{TA} \sum_{k=1}^n (A^{k-1} e^{-TA} \mathbf{b} U_k(T) - A^{k-1} \mathbf{b} U_k(0)) = e^{TA} \mathbf{x} + \sum_{k=1}^n A^{k-1} \mathbf{b} U_k(T), \quad (3)$$

where

$$U_k(t) = \int_0^t U_{k-1}(\tau) d\tau, \text{ for } k \geq 1 \text{ with } U_1(t) = \int_0^t u(\tau) d\tau.$$

Since u is piecewise constant on the partition of Ω determined by the t_i 's,

$$U_1(T) = u(0) \sum_{i=1}^n \int_{T_{i-1}}^{T_i} (-1)^{i-1} d\tau = u(0)(t_1 - t_2 + \dots + (-1)^{n-1} t_n),$$

where, as above, $T_i = t_1 + \dots + t_i$. For any $1 \leq k \leq n-1$, we have

$$\begin{aligned} U_k(T) &= \int_0^T U_{k-1}(\tau) d\tau = \sum_{i=1}^n \int_{T_{i-1}}^{T_i} U_{k-1}(\tau) d\tau = u(0) \sum_{i=1}^n \int_{T_{i-1}}^{T_i} (-1)^{i-1} \frac{\tau^{k-1}}{(k-1)!} d\tau \\ &= u(0) \sum_{i=1}^n (-1)^{i-1} \left[\frac{(t_i + T_{i-1})^k}{k!} - \frac{(T_{i-1})^k}{k!} \right] = u(0) \sum_{i=1}^n \sum_{j=1}^k \frac{(-1)^{i-1}}{(k-j)! j!} t_i^j T_{i-1}^{k-j}. \end{aligned}$$

Since

$$(t_1 + \dots + t_{i-1})^{k-j} = \sum_{\substack{\alpha_1 + \dots + \alpha_{i-1} = k-j \\ \alpha_1, \dots, \alpha_{i-1} \geq 0}} \frac{(k-j)!}{\alpha_1! \alpha_2! \dots \alpha_{i-1}!} t_1^{\alpha_1} \dots t_{i-1}^{\alpha_{i-1}},$$

$$\begin{aligned}
 U_k(T) &= u(0) \sum_{i=1}^n \sum_{\substack{\alpha_1 + \dots + \alpha_i = k \\ \alpha_1, \dots, \alpha_{i-1} \geq 0, \alpha_i > 0}} \frac{(-1)^{i-1}}{\alpha_1! \dots \alpha_i!} t_1^{\alpha_1} \dots t_i^{\alpha_i} \\
 &= -u(0) \sum_{\substack{\alpha_1 + \dots + \alpha_n = k \\ \alpha_1, \dots, \alpha_n \geq 0}} \frac{(-1)^{\max\{j: \alpha_j \neq 0\}}}{\alpha_1! \dots \alpha_n!} t_1^{\alpha_1} \dots t_n^{\alpha_n}. \quad (4)
 \end{aligned}$$

On the other hand, since A is a nilpotent Jordan block, we have $A_{ij} = \delta_{i,j-1}$, for $i, j = 1, \dots, n$. Then, for $1 \leq k \leq n$ we have $(A^{k-1})_{i,j} = \delta_{i,i+k-1}$. Thus, $(A^{k-1}\mathbf{b})_i = b_{k+i-1}$ for $i = 1, \dots, n-k+1$, and $(A^{k-1}\mathbf{b})_i = 0$ otherwise.

Finally, replacing (4) in (3), and noting that the i -th component of $e^{TA}\mathbf{x}$ is given by

$$(e^{TA}\mathbf{x})_i = \sum_{k=1}^n (e^{TA})_{i,k} x_k = \sum_{k=i}^n \frac{T^{k-i}}{(k-i)!} x_k,$$

we obtain the equality (2), as required.

Remark 1 Note that the control signal $u(t)$ need not be optimal for the proof of Proposition 1 to hold, as it only requires that $u(t)$ is parametrized by the non-negative increments $\{t_i\}_{i=1}^n$ and $u(0)$. The connection to optimality is provided by Theorem 1, which guarantees that the time-optimal control is bang-bang with at most $n-1$ switches and hence admits such a parametrization. Moreover, for the origin as target point, the sets of states reachable with $u(0) = 1$ and $u(0) = -1$ are disjoint up to a set of measure zero [28], so that for almost every x , there exists a unique $u(0) \in \{-1, +1\}$ for which (2) admits a non-negative solution. Among all non-negative solutions, the time-optimal one minimizes $T = t_1 + \dots + t_n$.

Example 1 In $\mathbb{R}^3$, consider the vector $\mathbf{b} = (0, 0, 1)^\top$. A solution of (1) leads to the system of polynomial equations:

$$\begin{aligned}
 0 &= u(0)(t_1 - t_2 + t_3) + x_3, \\
 0 &= u(0) \left(\frac{t_1^2}{2} + t_1 t_2 + t_1 t_3 - \frac{t_2^2}{2} - t_2 t_3 + \frac{t_3^2}{2} \right) + x_2 + (t_1 + t_2 + t_3)x_3, \\
 0 &= u(0) \left(\frac{t_3^3}{6} + \frac{t_1^2 t_2}{2} + \frac{t_1^2 t_3}{2} + \frac{t_1 t_2^2}{2} + t_1 t_2 t_3 + \frac{t_1 t_3^2}{2} - \frac{t_2^3}{6} - \frac{t_2^2 t_3}{2} - \frac{t_2 t_3^2}{2} + \frac{t_3^3}{6} \right) \\
 &\quad + x_1 + (t_1 + t_2 + t_3)x_2 + \frac{(t_1 + t_2 + t_3)^2}{2} x_3,
 \end{aligned}$$

and $u(0) = \pm 1$.

With no loss of generality,¹ we will henceforth assume $\mathbf{b} = (0, \dots, 0, 1)^\top$, leading to a simplified version of (2)

$$0 = -u(0) \sum_{\substack{\alpha_1, \dots, \alpha_n \geq 0 \\ \alpha_1 + \dots + \alpha_n = n-i+1}} (-1)^{\max\{j: \alpha_j \neq 0\}} \frac{t_1^{\alpha_1} \dots t_n^{\alpha_n}}{\alpha_1! \dots \alpha_n!} + \sum_{k=i}^n \frac{(t_1 + \dots + t_n)^{k-i}}{(k-i)!} x_k, \quad (5)$$

for $i = 1, \dots, n$. This system characterizes the optimal control signal in the following way. For every initial condition $\mathbf{x}$, there exists a unique optimal control signal, which is determined

¹ It is well-known that for scalar control systems in Jordan form, controllability is equivalent to requiring $b_n \neq 0$ [10].

by the choice of the initial control $u(0) = \pm 1$, and the existence of a set of non-negative t_i 's related to that choice. In other words, if a non-negative sequence of t_i 's is found for a given $\mathbf{x}$ and $u(0)$, then the switching sequence is the time-optimal solution to the problem. Otherwise, we are guaranteed a unique non-negative solution to (5) by taking $-u(0)$ as the starting element of the sequence and computing the associated switching times. In the following section, we turn our attention to Newton's method to find roots of the polynomial system.

3 A Deflated Newton's Method

In this section, we present a numerical technique that combines Newton's method with a deflation routine to determine all possible solutions of the polynomial system (5).

We denote by $f_1, \dots, f_n$ the polynomials in $\mathbf{t} = (t_1, \dots, t_n)$ from (5), which reads

$$\mathcal{F}(\mathbf{t}) = (f_1(\mathbf{t}), \dots, f_n(\mathbf{t}))^\top = \mathbf{0}. \quad (6)$$

Given an initial guess $\mathbf{t}_0$, Newton's method generates a sequence

$$\mathbf{t}_{h+1} = \mathbf{t}_h - \mathcal{F}'(\mathbf{t}_h)^{-1} \mathcal{F}(\mathbf{t}_h), \quad (7)$$

where $\mathcal{F}'$ denotes the Jacobian of $\mathcal{F}$. Assuming that this iteration converges to a root of (5), we have no guarantees that it will converge to a non-negative solution, and, unless all solutions have been found, this cannot be interpreted as the non-existence of non-negative solutions. Hence, we resort to a deflated Newton method to find all possible roots, which we describe in the following.

When using an iterative algorithm to find the roots of a function $\mathcal{F}(\mathbf{t})$, one can search for new, distinct roots of $\mathcal{F}(\mathbf{t})$ by applying the same iterative algorithm to a new deflated function $\mathcal{G}(\mathbf{t})$ whose zeros coincide with the roots of $\mathcal{F}(\mathbf{t})$, except for those already found. Specifically, if $\mathcal{F}(\mathbf{r}) = \mathbf{0}$, for some $\mathbf{r} \in \mathbb{R}^n$, we define

$$\mathcal{G}(\mathbf{t}) = \mathcal{M}(\mathbf{t}, \mathbf{r}) \mathcal{F}(\mathbf{t}),$$

where $\mathcal{M}(\mathbf{t}, \mathbf{r})$ is the *deflation matrix*, which is invertible for all $\mathbf{t} \in \mathbb{R}^n \setminus \{\mathbf{r}\}$, and $\mathcal{M}(\mathbf{t}, \mathbf{r}) \mathcal{F}(\mathbf{t}) = \mathbf{0}$ if and only if $\mathcal{F}(\mathbf{t}) = \mathbf{0}$. This involves systematically altering the residual of the polynomial system to eliminate solutions that have already been identified. In this work, we consider

$$\mathcal{G}(\mathbf{t}) = \left(\mathbf{I} \frac{1}{\eta(\mathbf{t})} + \xi \mathbf{I} \right) \mathcal{F}(\mathbf{t}), \text{ for } \eta(\mathbf{t}) = \|\mathbf{t} - \mathbf{r}\|_2^p, \quad (8)$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix, the *deflation power* p is a positive integer, and the *deflated residual* ξ is a non-negative real number. The *deflating term* $\eta(\mathbf{t})^{-1}$ ensures that $\mathcal{G}(\mathbf{r}) \neq \mathbf{0}$, while the shift $\xi \geq 0$ forces the deflated residual not to vanish artificially as $\|\mathbf{t} - \mathbf{r}\|_2 \rightarrow \infty$.

The use of Newton's method requires the computation of the derivative of $\mathcal{G}(\mathbf{t})$. This can be done efficiently both in memory and computational cost by leveraging the information about the previous deflation iterations. If $\mathbf{r}_0, \dots, \mathbf{r}_{m-1}$ are roots of $\mathcal{F}$, at the m -th iteration, the deflation term reads

$$\eta_m = \eta_{m-1}(\mathbf{t}) \tau(\mathbf{t}) \text{ with } \eta_{m-1}(\mathbf{t}) = \prod_{i=0}^{m-1} \|\mathbf{t} - \mathbf{r}_i\|_2^p,$$

for a temporal variable $\tau(t) = \|t - r_m\|_2^p$. The derivative of the polynomial system after m deflations is given by

$$\mathcal{G}' = \frac{\mathcal{F}'}{\eta_m} - \frac{\mathcal{F}}{\eta_m^2} \otimes (\eta'_{m-1} \tau + \eta_{m-1} \tau'). \quad (9)$$

For the time-optimal polynomial system (5), the partial derivative of the i -th component f_i of $\mathcal{F}$ with respect to the switching time t_q is given by the polynomial

$$\begin{aligned} \partial_{t_q} f_i = & -u(0) \sum_{\substack{\alpha_1, \dots, \alpha_n \geq 0, \alpha_q \neq 0 \\ \alpha_1 + \dots + \alpha_n = n-i+1}} (-1)^{\max\{j: \alpha_j \neq 0\}} \frac{t_1^{\alpha_1} \dots t_q^{\alpha_q-1} \dots t_n^{\alpha_n}}{\alpha_1! \dots (\alpha_q-1)! \dots \alpha_n!} \\ & + \sum_{k=i+1}^n \frac{T^{k-i-1}}{(k-i-1)!} x_k. \end{aligned}$$

A pseudocode for this routine is provided in Algorithm 1. A crucial consideration for the deflation algorithm is the selection of suitable stopping criteria, as once all roots have been found, the algorithm will naturally fail to converge. However, numerical instability can also prevent convergence as the number of roots in the deflation operator increases. Therefore, establishing a bound on the number of real solutions in our polynomial system is fundamental. The next section explores this bound by drawing from real algebraic geometry and utilizing the Hermite quadratic form.

Algorithm 1 Deflated Newton

Data: function $\mathcal{F}$ and its derivative $\mathcal{F}'$, initial guess, expected number of roots n_{roots} , deflation parameters p, ξ , convergence tolerance $\varepsilon \ll 1$;

$\text{max_iteration} = 10^3$, $\text{solutions} \leftarrow \{\}$; initializing

$\eta_0(*) \leftarrow 1, \eta'_0(*) \leftarrow 0, \tau(*) \leftarrow 1, \tau'(*) \leftarrow 0$; deflation operators

$m \leftarrow 0$; count of identified roots

while $m < n_{\text{roots}}$ **do**

$t \leftarrow \text{guess}$; initial guess

$\text{update} \gg 1, \text{iteration} \leftarrow 0$; reset convergence flags

while $\text{update} > \varepsilon$ and $\text{iteration} < \text{max_iteration}$ **do**

$\text{iteration} += 1$;

$f \leftarrow \mathcal{F}(t), df \leftarrow \mathcal{F}'(t)$;

$\mathcal{G} \leftarrow (8), \mathcal{G}' \leftarrow (9)$; compute deflated system

$t_{\text{new}} \leftarrow t - \text{pinv}(\mathcal{G}') \cdot \mathcal{G}$; update root candidate

$\text{update} \leftarrow \|t_{\text{new}} - t\|$;

$t \leftarrow t_{\text{new}}$;

if $\text{iteration} = \text{max_iteration}$ **then**

break

else

$\text{solutions} \leftarrow \text{solutions} \cup \{t\}$; store solution

$m += 1$;

$\tau(*) \leftarrow \|* - t\|^p$; update deflation operators

$\tau'(*) \leftarrow p \cdot \|* - t\|^{p-2} \cdot (* - t)$;

$\eta_m(*) = \eta_{m-1}(*)\tau(*)$;

$\eta'_m(*) = \eta'_{m-1}(*)\tau(*) + \eta_{m-1}(*)\tau'(*)$.

Result: list solutions , containing the roots of $\mathcal{F}$.

4 The Hermite Quadratic Form

We present the Hermite quadratic form method, as introduced in [3], as a tool to determine the number of roots in the polynomial system (5). We denote by $R = \mathbb{R}[t_1, \dots, t_n]$ the ring of polynomials in t with real coefficients. Let

$$J = \langle f_i : i = 1, \dots, n \rangle = \left\{ \sum_{i=1}^n g_i f_i : g_i \in R \right\} \subseteq R \quad (10)$$

be the ideal of R generated by the polynomials $f_i \in R$ in (6). The ideal J is said to be *zero-dimensional* if the set $R/J = \{g + J : g \in R\}$ of polynomials module J forms a finite dimensional real vector space, where $g + J = \{g + f \in R : f \in J\}$, for any $g \in R$.

To apply the Hermite quadratic form method, it is necessary to ensure that R/J is a finite-dimensional vector space, which is equivalent to J being zero-dimensional. This property can be verified using a monomial ordering, which can be used either to establish certain properties of the monomials which are powers of the variables t_i or to compute a Gröbner basis for the ideal J . Alternatively, if we can verify that the polynomials f_i generating J have only a finite number of solutions in $\mathbb{C}$, this also implies that J is zero-dimensional. For further details on these methods, see, for example, [13, Sect. 3, Chapter 5].

Definition 1 (Hermite quadratic form) If a polynomial system $\{f_i\}_{i=1}^n$ in R has a finite number of solutions in $\mathbb{C}$, and $\mathcal{B} = \{\beta_1, \dots, \beta_r\}$ is a basis of R/J , for J as in (10), let $\mathcal{L}_{ij}(J)$ be the map on R/J defined by

$$f + J \mapsto \beta_i \beta_j f + J,$$

for $1 \leq i, j \leq r$. The *Hermite quadratic form* of J is the map on R/J defined by the matrix $(\mathcal{H}(J))_{ij} = \text{trace}(\mathcal{L}_{ij}(J))$, where $\mathcal{L}_{ij}(J)$ is the matrix of the linear map $\mathcal{L}_{ij}(J)$. In particular, $\mathcal{H}(J)$ is a real symmetric matrix.

Example 2 Taking $n = 2$ and $u(0) = 1$ in (5) leads to

$$f_1 = t_1 - t_2 + x_2, \quad (11)$$

$$f_2 = \frac{t_1^2}{2} + t_1 t_2 - \frac{t_2^2}{2} + x_1 + (t_1 + t_2)x_2. \quad (12)$$

These are polynomials in $R = \mathbb{R}[t_1, t_2]$, and we take $J = \langle f_1, f_2 \rangle$. This system has a finite number of solutions, because if $t_2 = t_1 + x_2$ by (11), then (12) leads to a quadratic equation in t_1 , which has at most two solutions. We see that $\mathcal{B} = \{1, t_2\}$ is a basis for the vector space R/J . Taking $\beta_1 = 1$, and $\beta_2 = t_2$, then the map $\mathcal{L}_{12}(J)$ is given by $1 + J \mapsto t_2 + J$, and $t_2 + J \mapsto t_2^2 + J = -x_1 + \frac{1}{2}x_2^2 + J$. Thus,

$$L_{12}(J) = \begin{pmatrix} 0 & -x_1 + \frac{1}{2}x_2^2 \\ 1 & 0 \end{pmatrix},$$

and so $\text{trace}(L_{12}(J)) = 0$. Similarly, we compute $L_{ij}(J)$ for $i, j = 1, 2$, and we get the Hermite quadratic form of J , which is given by the matrix $\mathcal{H}(J) = \begin{pmatrix} 2 & 0 \\ 0 & x_2^2 - 2x_1 \end{pmatrix}$.

In practice, for a given value of x , we use Macaulay2 [18] to compute the dimension of the ideal J and verify if it is zero. The Hermite quadratic form for parametric systems of polynomial equations has been studied in [29]. Here, it is shown that for a given parameter,

it is possible to obtain the Hermite quadratic of the associated polynomial system, as long as the denominator of the parameter-augmented Hermite quadratic form does not vanish.

If the ideal J generated by $\{f_i\}_{i=1}^n$ in (6) is zero-dimensional, the Hermite quadratic form $\mathcal{H}(J)$ can be used to count the number of real roots of $\mathcal{F}$ as follows. First, recall that the rank of the matrix $\mathcal{H}(J)$, denoted $\text{rank}(\mathcal{H}(J))$, is equal to the number of non-zero eigenvalues of $\mathcal{H}(J)$ counted with multiplicity, and the *signature*, denoted $\text{sign}(\mathcal{H}(J))$, is the difference between the number of its positive and negative eigenvalues. In addition, we denote by $\mathbb{V}_{\mathbb{R}}(J)$ and $\mathbb{V}_{\mathbb{C}}(J)$ the set of real and complex solutions of $\mathcal{F}$, respectively. The following result, which is a special case of the one presented in [3, Theorem 4.100], establishes a link between the Hermite quadratic form $\mathcal{H}(J)$ and the number of roots of $\mathcal{F}$.

Proposition 2 Suppose the system of polynomial equations $\{f_i\}_{i=1}^n$ in R has finitely many solutions. Let $J = \langle f_i : i = 1, \dots, n \rangle$, and $\mathcal{H}(J)$ be the Hermite quadratic form of J . Then,

$$\text{rank}(\mathcal{H}(J)) = \#\mathbb{V}_{\mathbb{C}}(J), \text{ and } \text{sign}(\mathcal{H}(J)) = \#\mathbb{V}_{\mathbb{R}}(J).$$

Remark 2 By Proposition 2, the number of real roots of a polynomial system can be computed by counting the positive and negative real roots of the characteristic polynomial of the matrix $\mathcal{H}(J)$.

Example 3 In Example 2, the signature of $\mathcal{H}(J)$ is determined by the sign of the eigenvalue $x_2^2 - 2x_1$. If $x_2^2 - 2x_1 > 0$, then both eigenvalues of $\mathcal{H}(J)$ are positive, meaning that the system of polynomial equations has two real solutions. If $x_2^2 - 2x_1 = 0$, then the system has only one real solution. If $x_2^2 - 2x_1 < 0$, then $\text{sign}(\mathcal{H}(J)) = 0$, and the system has no real solutions. These three cases are illustrated in Fig. 1.

The Hermite quadratic form feeds into Algorithm 1 by providing n_{roots} , the expected number of real roots. We set $n_{\text{roots}} = \text{sign}(\mathcal{H}(J))$. This quantity is valid for fixed x and $u(0)$, and once n_{roots} has been reached without finding a non-negative solution, we are guaranteed the existence of a non-negative solution by taking $-u(0)$.

All of our computations are completed in Macaulay2 using the RealRoots package [23]. The command “traceCount(J)” returns the signature of the Hermite quadratic form of J . Within this computation, a Gröbner basis of J is computed when finding a basis of R/J . As shown in [15], the algorithms to compute the Gröbner basis of the ideal J will generate polynomials with total degree bounded above by $2(n^2/2 + n)^{2^{n-1}}$. As a result, as we increase the number of variables n , the Gröbner basis computation will increase double-exponentially. In Sect. 6, we report results, where the Hermite quadratic form has been computed for polynomial systems up to dimension 5, although computations have been successfully performed up to dimension 7.

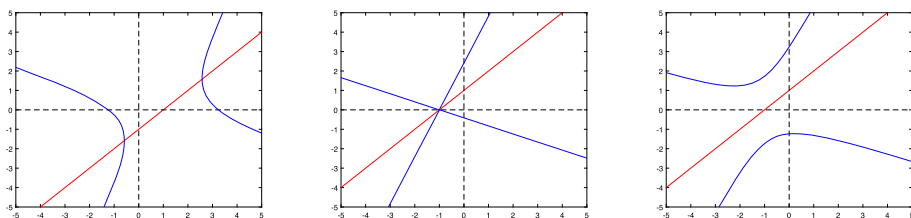


Fig. 1 Graph of the two-dimensional polynomial system from Example 2. The number of real solutions of the system depends on the sign of the diagonal elements in the matrix $\mathcal{H}(J)$. We have three cases: $x_2^2 - 2x_1 > 0$ (left), $x_2^2 - 2x_1 = 0$ (middle), and $x_2^2 - 2x_1 < 0$ (right)

5 Constructing a Neural Feedback Law as a Binary Classifier

The methodology developed so far generates open-loop controls, which are effective in an ideal setting, where both system dynamics and initial conditions are exactly known. However, in real-world applications, model uncertainties and disturbances will lead to deviations from the optimal trajectory, requiring a re-computation of the optimal action. Such a computationally intensive task limits the applicability of such a synthesis procedure for real-time control. These issues are well-understood in the control literature, particularly in the context of stabilizing control, and are typically mitigated by adopting a feedback control approach, which is inherently more robust. Time-optimal feedback controls are static maps that depend solely on the current state of the system and are evaluated as the system evolves. This adaptability allows them to correct deviations from the optimal trajectory, improving their robustness to uncertainties and disturbances.

In deterministic optimal control, the construction of an optimal feedback law is obtained via dynamic programming, leading to a static, first-order Hamilton-Jacobi-Bellman PDE for the value function of the control problem. The optimal feedback law is obtained as a by-product. However, this method requires an accurate tracking of the discontinuous switching law, which can only be achieved using local, grid-based numerical methods, and hence it is limited to low-dimensional problems. In this section, we present an alternative synthesis method that circumvents the solution of the Hamilton-Jacobi-Bellman PDE by resorting to supervised learning. Here, we train an NN using synthetic data from sampling open-loop, optimal trajectories obtained with the methodology presented in previous sections.

5.1 Feedforward Neural Network Classifier

Given the bang-bang (binary) nature of the control in the problem under consideration, we frame the feedback approximation as a classification task rather than relying on regression. This means that the output of our model will be of a discrete nature in $\{-1, 1\}$.

We consider a class of models within the family of feedforward NNs characterized by a sequential composition of nonlinear activation functions applied componentwise to affine transformations of the inputs. If we define the m th layer as $l_m(\mathbf{z}_m) = \sigma_m(A_m \mathbf{z}_m + \mathbf{b}_m)$, the parametric model reads

$$u_\theta(\mathbf{z}) = l_M \circ \cdots \circ l_2 \circ l_1(\mathbf{z}),$$

with trainable parameters $\{A_m, \mathbf{b}_m\}_{m=1}^{M-1}$, $A_m \in \mathbb{R}^{n_{m-1} \times n_m}$ and $\mathbf{b}_m \in \mathbb{R}^{n_m}$ being the weight matrix and bias vector of the m th layer, respectively, which has n_m neurons. Here, the nonlinear activation function σ_m , $m = 1, \dots, M-1$, is applied componentwise to the layer input variable $\mathbf{z}_m \in \mathbb{R}^{n_{m-1}}$. We fix the final layer to be sigmoid

$$l_M(\mathbf{z}_M) = \frac{1}{1 + e^{-\mathbf{z}_M}},$$

where, as before, the function is applied componentwise to the entries of the layer input $\mathbf{z}_M$. This function maps any real-valued input into the range $(0, 1)$, making it particularly suitable for binary classification tasks, as the output can be interpreted as a probability:

$$\mathbb{P}[u(\mathbf{z}) = 1] = u_\theta(\mathbf{z}), \text{ and } \mathbb{P}[u(\mathbf{z}) = -1] = 1 - u_\theta(\mathbf{z}). \quad (13)$$

By denoting the approximation of the feedback control as $\tilde{u}(z) \approx u(z)$, we define it as the most likely event under the probability u_θ :

$$\tilde{u}(z) = \begin{cases} 1, & \text{if } u_\theta(z) \geq 0.5, \\ -1, & \text{otherwise.} \end{cases}$$

This formulation not only provides a classification decision but also quantifies the confidence of the model in the forecast, in the sense that the farther $u_\theta(z)$ is from the decision threshold 0.5, the higher the model's confidence in its classification.

5.2 Synthetic Data Generation

For the generation of the training and testing datasets, we utilize the deflated Newton's method discussed in Sect. 3 and solve the polynomial system in (5). We begin by generating N_s sample points in the state space, denoted by $\{\mathbf{x}^{(i)}\}_{i=1}^{N_s}$. For each initial condition, we evaluate the solvability of the polynomial system under the two possible initial control values, $u_0 = 1$ and $u_0 = -1$.

To ensure the accurate detection of switching surfaces, we further store the state and control information for a sequence of 100 points along the resulting optimal trajectory generated from each initial condition. This process results in a dataset consisting of $N = 100N_s$ entries:

$$\mathcal{T} = \{\mathbf{x}^{(i)}, u_0^{(i)}\}_i \cup \{\mathbf{y}^{(i)}(\tau_k), u^{(i)}(\tau_k)\}_{i,k}, \quad \text{for } i = 1, \dots, N_s, \text{ and } k = 1, \dots, 100,$$

where $\mathbf{y}^{(i)}(t)$ is the optimal trajectory departing from $\mathbf{x}^{(i)}$ and τ_k are uniformly spaced discrete times in $(0, T]$. For convenience, we will denote the input $u_\theta(\cdot)$ as z , and rearrange the training set as

$$\mathcal{T} = \{\mathbf{z}^{(j)}, u^{(j)}\}_{j=1}^N,$$

where $N = 100N_s$.

5.3 Training of the Model

The network is trained to minimize a cross-entropy loss function on data, ensuring the output probabilities align with the provided labels. The training is performed using the Adam optimizer [20]. The dataset is divided into two subsets: 90% for training, 10% for testing. The number of layers and neurons is selected to balance model complexity and computational efficiency, and hyperparameters such as learning rate and batch size are tuned using a grid search with the aim of maximizing the approximation performance. These choices are problem-specific and will be clarified for each numerical test in Sect. 6.

In this case, the binary cross-entropy loss function measures how well the NN's predicted probabilities align with the true binary labels ($u(z) \in \{-1, 1\}$). The binary cross-entropy loss for a single data point z is given by

$$L(z, u(z)) = -[\lambda \log(u_\theta(z)) + (1 - \lambda) \log(1 - u_\theta(z))],$$

where

- $\lambda \in \{0, 1\}$ is the true label, with $\lambda = 0$ corresponding to $u(z) = -1$ and $\lambda = 1$ corresponding to $u(z) = 1$,
- $u_\theta(z)$ is the network's predicted likelihood that $u(z) = 1$, as in (13).

For a dataset of N samples, the total binary cross-entropy loss is

$$\mathcal{L} = \frac{1}{N} \sum_{j=1}^N [-\lambda_j \log(u_\theta(z^{(j)})) - (1 - \lambda_j) \log(1 - u_\theta(z^{(j)}))]. \quad (14)$$

This loss function encourages the network to predict probabilities $u_\theta(z^{(j)})$ close to 1 when $\lambda_j = 1$ and close to 0 when $\lambda_j = 0$, effectively learning the mapping from states $z^{(j)}$ to control labels $u^{(j)}$.

The model's performance is assessed by measuring the proportion of correctly predicted labels out of the total number of samples:

$$\text{Accuracy} = \frac{1}{N} \sum_{j=1}^N \mathbf{1}_{(\tilde{u}(z^{(j)})=u^{(j)})},$$

where $\mathbf{1}$ denotes the indicator function.

6 Numerical Tests

In this section, we evaluate the proposed methodology on an n th order integrator, where n ranges from 2 to 5. The discussion includes the process of data generation, the architecture of the model, and the choice of hyperparameters. In particular, the network architecture was selected via a grid search. We explored NNs with between 1 and 3 hidden layers, and a number of neurons drawn from the set $\{10, 20, 30, 40, 50, 60, 80, 100\}$. Across this search space, the tanh activation function consistently yielded the most stable and accurate results. For completeness, we also evaluated ReLU and sigmoid activations, but these did not improve performance in our setting. The final architectures reported in the following tests correspond to the best-performing models identified from the above grid.

We compare the trajectories controlled by the trained model against the optimal trajectories obtained using Algorithm 1 to solve (5). Across the numerical examples, we demonstrate that interpreting u_θ as a binary classifier with confidence information improves its approximation performance. We evaluate the robustness of the feedback approximation under noisy trajectories and highlight the computational advantages of using Hermite quadratic forms to determine a bound for the number of roots to be identified through deflation.

6.1 The Double Integrator

We consider the time-optimal control problem

$$\min_{u(t)} T \quad \text{such that} \quad \begin{cases} \dot{y}_1 = y_2, & y_1(0) = x_1, & y_1(T) = 0, \\ \dot{y}_2 = u, & y_2(0) = x_2, & y_2(T) = 0, \end{cases}$$

whose solution is given by a piecewise constant control, starting from an initial control $u_0 \in \{-1, 1\}$ and a switching sequence $\mathbf{t} = (t_1, t_2)^\top$, computed by solving the associated polynomial system (5), which for $n = 2$ reads

$$\begin{cases} t_1 - t_2 + x_2 = 0, & \frac{t_1^2}{2} + t_1 t_2 - \frac{t_2^2}{2} + x_1 + x_2(t_1 + t_2) = 0 & \text{for } u_0 = 1, \\ t_1 - t_2 - x_2 = 0, & \frac{t_1^2}{2} + t_1 t_2 - \frac{t_2^2}{2} - x_1 - x_2(t_1 + t_2) = 0 & \text{for } u_0 = -1. \end{cases} \quad (15)$$

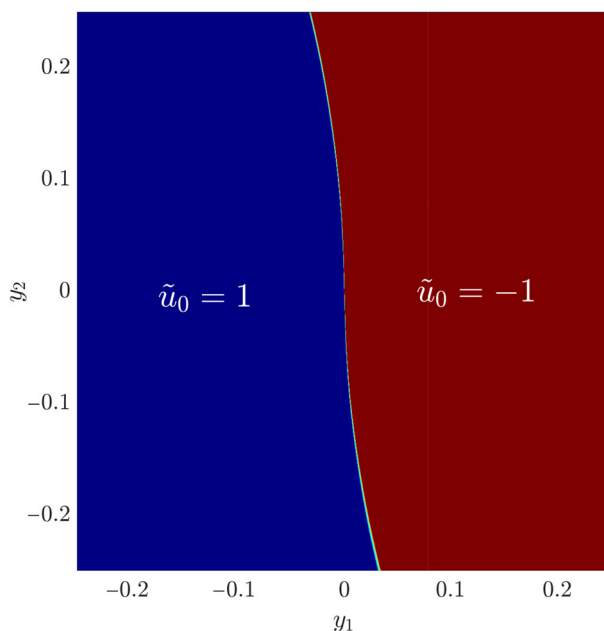


Fig. 2 Trained classifier identifies the switching surface as a low-confidence region

Admissible solutions of (15) specify the optimal trajectory, starting from $\mathbf{x} = (x_1, x_2)^\top$ with control u_0 and switching sequence² $\mathbf{t}$. We generate a dataset by sampling $N_s = 50$ uniformly distributed initial conditions $\{\mathbf{x}^{(i)}\}_{i=1}^{N_s}$ in the numerical domain $[-1, 1]^2$, generating couples of optimal state-actions $\{\mathbf{y}^{(i)}(\tau_k), u^{(i)}(\tau_k)\}$ for $0 = \tau_0 < \dots < \tau_k < \dots < \tau_{100} = T$.

We rely on the $N = 100 \cdot N_s$ generated data pairs

$$\mathcal{T} = \{\mathbf{z}^{(j)}, u^j\}_{j=1}^N$$

to train an NN classifier mapping states $\mathbf{z}$ into the associated feedback $u_\theta(\mathbf{z})$. After a grid search over a number of possible feedforward architectures, we select $u_\theta(\cdot)$ to have a single hidden layer of width $n_2 = 100$ neurons and $\sigma_2 = \tanh(\cdot)$ as an activation function. We train the model to minimize the loss function (14) over the training set via Adam.

After training, the model achieves a test set accuracy of 99.38% with a corresponding loss $\mathcal{L} = 0.0156$. In Fig. 2, we show how the trained classifier labels points in $[-0.25, 0.25]^2$. The model assigns controls with high probability to points on either side of the switching surface, while points lying directly on the surface are associated with lower confidence, as indicated by the lighter colors in the figure.

All optimal trajectories eventually intersect the switching hyperplane and travel along it until reaching the origin. Therefore, the model's accuracy on the switching surface is crucial for achieving a reliable feedback approximation. At this point, we have two options:

- we rely on the classification provided by the NN u_θ , disregarding its low confidence,

² For the sake of clarity, we recall that this switching sequence is to be interpreted in the following way: there is an initial control value u_0 in $[0, t_1)$, which then switches to $-u_0$ in $[t_1, t_1 + t_2]$, being the minimum time $T = t_1 + t_2$.

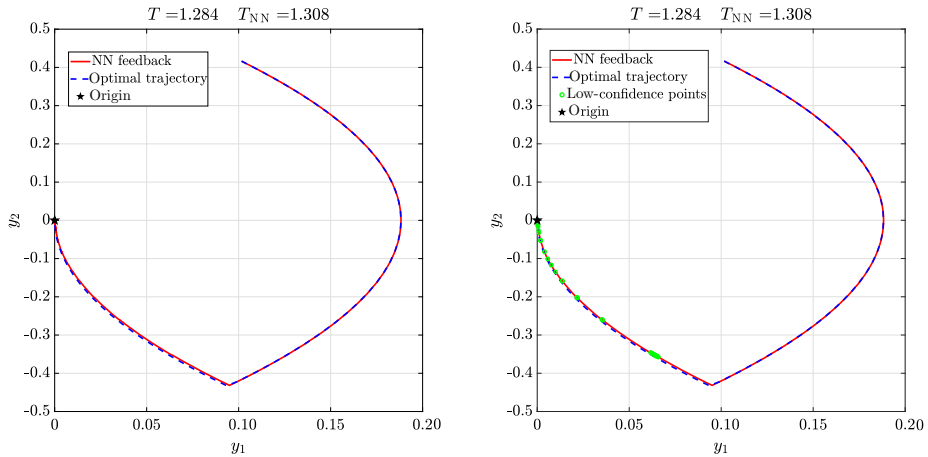


Fig. 3 Comparison of the optimal trajectory computed via deflation with the one controlled with the classifier u_θ . On the left, the NN feedback is solely determined via u_θ , while on the right we rely on the identification of the solution of the polynomial system (15) to identify the feedback at low-confidence points. T denotes the exact minimum time

- or we define a confidence threshold, and if the confidence falls below this threshold, we invoke the open-loop solver to determine the feedback control at the low-confidence point.

We compare these two procedures in Fig. 3. The trajectory controlled through the neural feedback approximation is integrated via explicit Euler, with time-step 5^{-4} . The enhanced approximation—which is based on the polynomial system solution when the classifier confidence is below $\varepsilon = 0.01$ —is triggered on 1% of the trajectory points and does not significantly improve the approximated optimal horizon T_{NN} . We conclude that, in this first example, even low-confidence points are classified correctly via u_θ .

6.2 Triple Integrator

Similarly, as in the first test, we consider the time-optimal control problem subject to

$$\begin{cases} \dot{y}_1 = y_2, \\ \dot{y}_2 = y_3, \\ \dot{y}_3 = u \end{cases}$$

with the state variable $\mathbf{y} = (y_1, y_2, y_3)^\top$ and boundary conditions $\mathbf{y}(0) = \mathbf{x} = (x_1, x_2, x_3)^\top$, $\mathbf{y}(T) = \mathbf{0}$. The polynomial system (5) features three equations for the unknown switching sequence $\mathbf{t} = (t_1, t_2, t_3)$. We sample state and controls along the optimal trajectories departing from $N_s = 5000$ uniformly distributed initial conditions in $[-1, 1]^3$. The resulting dataset is used to train a model u_θ characterized by a single hidden layer with width $n_2 = 80$ and activation function $\sigma_2 = \tanh(\cdot)$. After training, the model reaches 99.12% accuracy in the test set, associated with a loss $\mathcal{L} = 0.0334$. In this example, the enhanced feedback approximation, calling the polynomial solver for identifying the control at points with confidence below $\varepsilon = 0.005$, occurs on 4.08% of the trajectory points, and improves the approximated horizon when compared to the trajectory controlled solely via u_θ . We show the results in Fig.

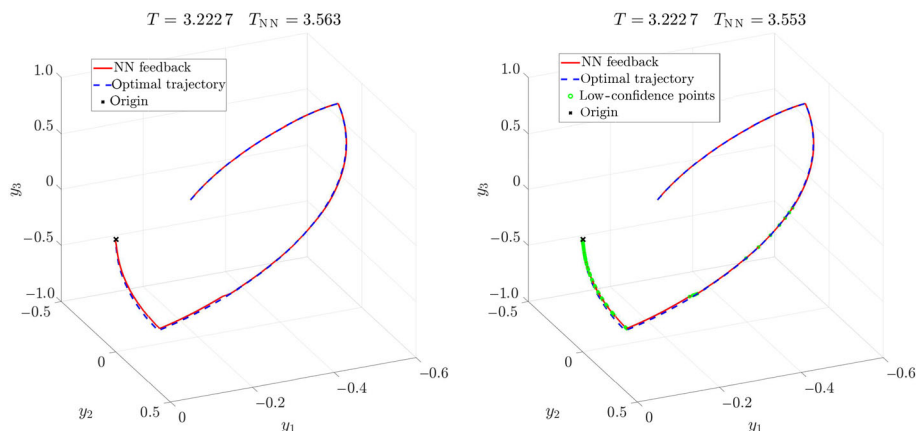


Fig. 4 Comparison of the optimal trajectory with the u_θ -controlled (left) and with the confidence-enhanced approximation (right). Relying on the polynomial solver for low-confidence points improves the approximation, as the time T_{NN} needed to reach the origin decreases

4, where the NN-controlled trajectory is computed via a forward Euler scheme with stepsize 10^{-3} , while the optimal trajectory is computed through exact integration.

We motivated the introduction of the feedback NN approximation as a way to achieve robustness against noise. To evaluate this, we introduce white noise into the last component of the system. Specifically, at every discrete integration time t , we add $(0, 0, \zeta(t))$, where $\zeta(t) \sim \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = 0.02$. Starting from the same initial condition, we perform a Monte Carlo simulation for both the open-loop and the approximated feedback control approaches. Figure 5 illustrates the mean and variance obtained from 1 000 noisy controlled trajectories. As expected, while the open-loop solution diverges on average from the target final state, the approximated feedback law successfully drives the system toward the origin.

6.3 Fourth-Order Tests

We consider the time-optimal control problem subject to the fourth-order integrator for the state variable $\mathbf{y} = (y_1, \dots, y_4)^\top$ and boundary conditions $\mathbf{y}(0) = \mathbf{x} = (x_1, \dots, x_4)^\top$, $\mathbf{y}(T) = \mathbf{0}$. In this higher-dimensional example, we examine the upper bound on the number of solutions of the associated system (5) for the switching sequence $t_1, \dots, t_4$. The theoretical bound on the total number of real and complex solutions is $4! = 24$. However, as discussed in Sect. 4, the use of Hermite quadratic forms allows us to refine this bound by identifying the number of real solutions more precisely.

We sample initial conditions $\mathbf{x}^{(i)} \in [-1, 1] \cap \mathbb{Q}^4$, for $i = 1, \dots, N_s = 10^4$, and for each $\mathbf{x}^{(i)}$ we compute the number of real roots of the polynomial system associated with $u_0 = 1$ and $u_0 = -1$, by computing the Hermite quadratic form, denoted by $\mathcal{H}(J^+)$ and $\mathcal{H}(J^-)$, respectively. Note the transition from $\mathbb{R}$ to $\mathbb{Q}$, which we do this as computation of the Hermite quadratic form over the reals is computationally unreliable, if not impossible. This preliminary step reduces the computational cost of data generation, as the bound for the deflation Algorithm 1 is tightened relative to the theoretical one. The range of real roots given by the Hermite quadratic form varies from 0 to 8, depending on the initial condition. For $\mathbf{x}^{(i)}$'s with 0 real solutions for J^+ or J^- , this directly determines the sign of the initial

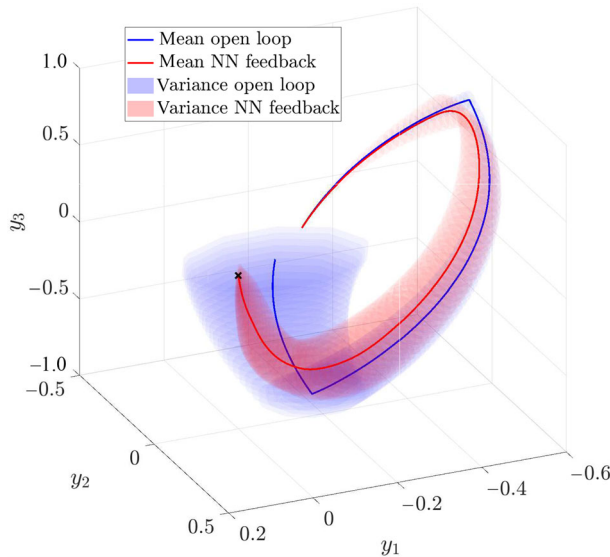


Fig. 5 Mean and variance of a Monte Carlo simulation with 1 000 trajectories perturbed by Gaussian noise. The robustness of the approximated feedback law ensures that the controlled system reaches the target destination, while the open-loop solution, obtained as the solution of the polynomial system (5), diverges from it

control u_0 . These new bounds reduce the CPU time for data generation from 108 to 49 min when compared to the theoretical bound.

We train a model u_θ characterized by 2 hidden layers, with widths $n_2 = n_3 = 100$ and activation functions $\sigma_2 = \sigma_3 = \tanh(\cdot)$. After training, we evaluate the model on the test set, obtaining an accuracy of 96.76% and a loss $\mathcal{L} = 0.0882$. In Fig. 6, we analyze the controlled trajectories comparing the optimal solution against its approximation along the dimensions y_2 , y_3 , and y_4 . We consider a low-confidence threshold $\varepsilon = 0.005$, under which we rely on the deflation routine to determine the control. Low-confidence classification triggers on 1.3% of the trajectory points. Even more prominently than in the lower order examples, the enhanced approximate feedback improves the performance of the control law. As in the previous example, we also test the robustness of the feedback approximation by injecting additive noise of the form $(0, 0, 0, \zeta(t))$, with $\zeta(t) \sim \mathcal{N}(0, \sigma^2)$, where $\sigma = 2^{-2}$. A Monte Carlo simulation of 1 000 controlled noisy trajectories shows how the approximated feedback control successfully steers the system toward the origin, while the open-loop solution diverges.

6.4 Fifth-Order Integrator

Finally, we consider the fifth-order integrator for state $\mathbf{y} = (y_1, \dots, y_5)^\top$ with boundary conditions $\mathbf{y}(0) = \mathbf{x} = (x_1, \dots, x_5)^\top$ and $\mathbf{y}(T) = \mathbf{0}$. The theoretical upper bound for the number of solutions $(t_1, \dots, t_5)$ of the associated polynomial system is $5! = 120$. However, excessively deflating a polynomial system can lead to instability. Heuristically, as the number of identified and deflated roots increases, the root-finding procedure may diverge, failing to locate additional existing solutions. This pathological behaviour naturally worsens as the upper bound increases. While still feasible, the CPU time associated with the computation

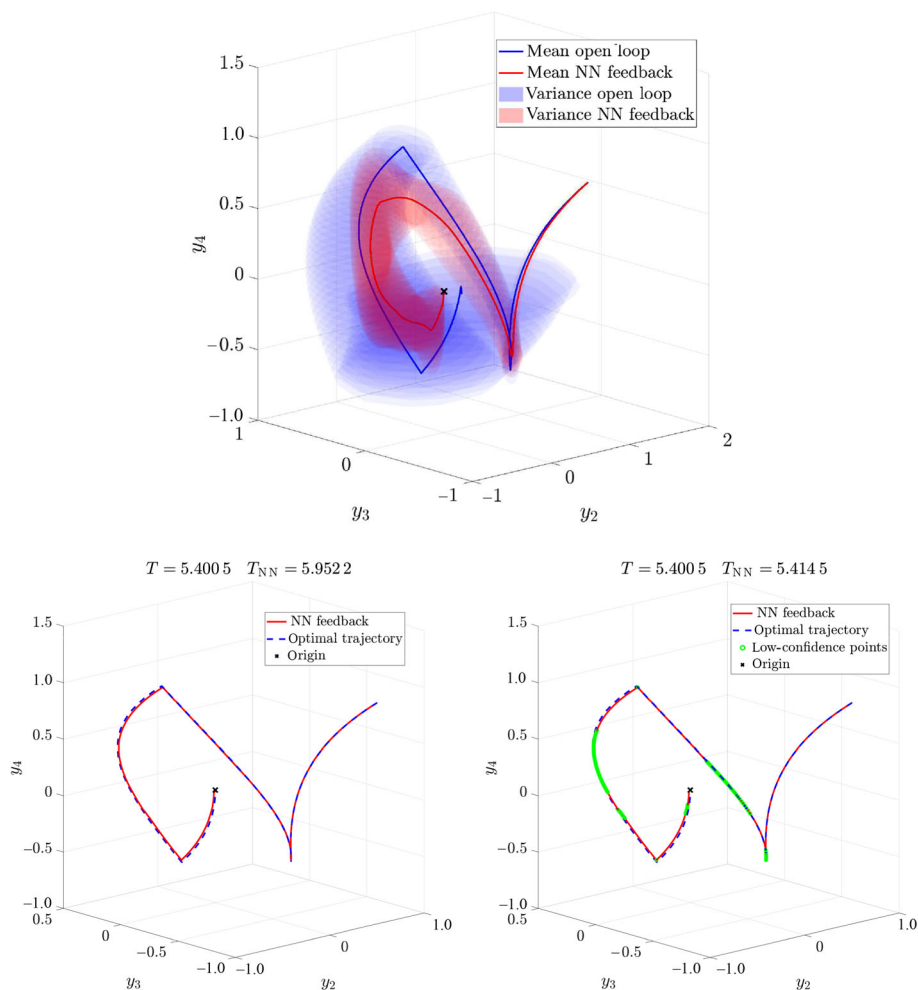


Fig. 6 Monte Carlo simulation of 1000 controlled noisy trajectories via open-loop control signal vs. approximated feedback control (top). Comparison of the optimal trajectory with the one controlled via the approximated feedback law (left). Calling the polynomial solver for the low-confidence points improves the approximation performance in terms of time horizon T_{NN} (right)

of the Hermite quadratic form can be used instead to sample different initial conditions with the uninformed deflation algorithm.

We sample $N_s = 50\,000$ points $\mathbf{x}^{(i)} \in [-1, 1]^5$, for $i = 1, \dots, N_s$, and include in the dataset $\mathcal{T}$ only the optimal trajectories corresponding to initial conditions for which the deflation algorithm successfully identifies an admissible solution. This selection process excludes approximately 20% of the initial conditions. We consider a model u_θ with 2 hidden layers having widths $n_2 = n_3 = 80$ and activation functions $\sigma_2 = \sigma_3 = \tanh(\cdot)$. The trained classifier achieves an accuracy of 99.61% on the test set, associated with a loss $\mathcal{L} = 0.0141$. In Fig. 7, we test the performance of the NN feedback controller against the optimal trajectory. The performance of the approximation is significantly improved by selectively intervening

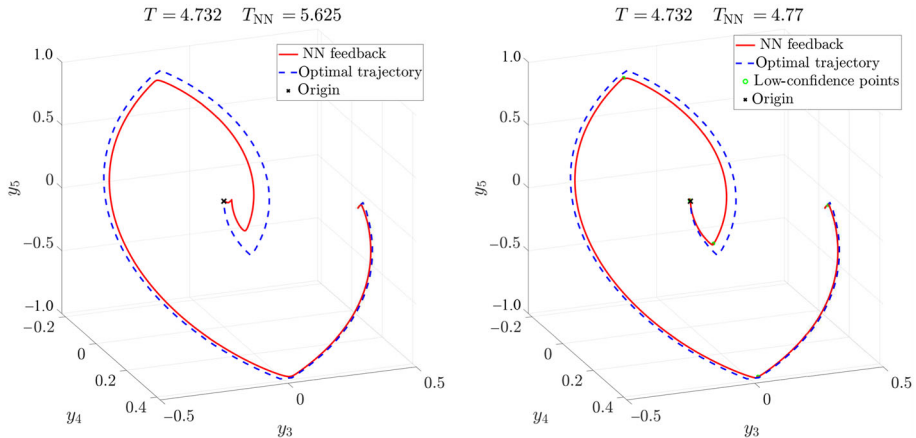


Fig. 7 Comparison of the optimal trajectory with the trajectory controlled using the approximated feedback law (left). Leveraging the polynomial solver for low-confidence points enhances the approximation's performance in terms of the time horizon T_{NN} (right)

with the polynomial system solver whenever the classifier confidence drops below $\varepsilon = 0.1$. This occurs only at 5 low-confidence points, all located at the intersection of switching surfaces.

7 Concluding Remarks

In this paper, we have revisited a classical time-optimal control problem from a perspective combining polynomial systems, numerical analysis, and statistical machine learning. Our original motivation was to explore the scalability of the approach originally developed in [34], where the use of Gröbner bases and techniques from real algebraic geometry were proposed to solve polynomial systems arising in time-optimal control. In this respect, we have reached a negative result: to the best of our experience, the computational cost and complexity of such tools become prohibitively expensive as the dimension of the state space increases beyond 3, rendering them inapplicable for constructing real-time feedback controllers. Hence, we have resorted to the numerical solution of polynomial systems using Newton's method on a deflated variant, which can effectively exhaust all possible roots. Here, the Hermite quadratic form is proposed as a tool to establish a bound on the number of roots of the polynomial system, increasing the efficiency of the deflation algorithm. Overall, this constitutes a consistent computational workflow for the solution of time-optimal trajectories for nilpotent linear systems.

In the second part of this paper, we adopt a statistical machine learning approach to interpret the time-optimal feedback controller as a binary classifier over the state space. This allows us to construct a control law via supervised learning, yielding a bang-bang control with a confidence indicator. This is particularly relevant to developing a robust, chattering-free control in the vicinity of the switching surface. Our numerical tests show that enhancing the feedback law with an open-loop solver whenever the confidence of the classifier is low leads to a fair trade-off between accuracy of the approximate optimal trajectory, robustness, and computational cost.

This work suggests different avenues for future research. The use of the Hermite quadratic form to determine the number of roots of time-optimal polynomial systems is promising and merits a deeper consideration. The interpolation of the bang-bang feedback law as a binary classifier enables the application of machine learning methods for a class of control problems that remains elusive to other types of global approximation techniques due to the non-smoothness of the optimal control field. Finally, the proposed methodology can be applied to other time-optimal and sparse control problems with bang-bang or bang-zero-bang structures.

Acknowledgements We thank two anonymous reviewers for their careful reading of the manuscript and for their comments that helped improve the clarity and presentation of this paper. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA8655-26-1-B011.

Data Availability This paper contains no datasets, not applicable.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Albi, G., Bicego, S., Herty, M., Huang, Y., Kalise, D., Segala, C.: Data/moment-driven approaches for fast predictive control of collective dynamics. In: Daniilidis, A., Grüne, L., Haunschmied, J., Tragler, G. (eds) *Model Predictive Control. Dynamic Modeling and Econometrics in Economics and Finance*, vol. 31, pp. 29–54. Springer, Cham (2025)
2. Azmi, B., Kalise, D., Kunisch, K.: Optimal feedback law recovery by gradient-augmented sparse polynomial regression. *J. Mach. Learn. Res.* **22**(48), 1–32 (2021)
3. Basu, S., Pollack, R., Roy, M.: *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics, 2nd ed. Springer, Berlin, Heidelberg (2006)
4. Bellman, R., Glicksberg, I., Gross, O.: On the “bang-bang” control problem. *Q. Appl. Math.* **14**, 11–18 (1956)
5. Ben-Or, M., Kozen, D., Reif, J.: The complexity of elementary algebra and geometry. *J. Comput. Syst. Sci.* **32**, 251–264 (1986)
6. Bicego, S., Kalise, D., Pavliotis, G.A.: Computation and control of unstable steady states for mean field multiagent systems. *Proc. Roy. Soc. A* **481**, 20240476 (2025)
7. Bonnard, B., Cots, O., Privat, Y., Trélat, E.: Zermelo navigation on the sphere with revolution metrics. In: *Ivan Kupka Legacy: a Tour Through Controlled Dynamics*. Volume 12 of *AIMS on Applied Mathematics*, pp. 35–65. American Institute of Mathematical Sciences, Springfield (2023)
8. Brown, K., Gearhart, W.: Deflation techniques for the calculation of further solutions of a nonlinear system. *Numerische Mathematik* **16**, 334–342 (1970/1971)
9. Charalampidis, E., Boullé, N., Farrell, P., Kevrekidis, P.: Bifurcation analysis of stationary solutions of two-dimensional coupled Gross-Pitaevskii equations using deflated continuation. *Commun. Nonlinear Sci. Numer. Simul.* **87**, 105255 (2020)
10. Chen, C.-T., Desoer, C.: A proof of controllability of Jordan form state equations. *IEEE Trans. Autom. Control* **13**, 195–196 (1968)
11. Cipriani, C., Scagliotti, A., Wöhrer, T.: A minimax optimal control approach for robust neural ODEs. In: *2024 European Control Conference (ECC)*, pp. 58–64. IEEE (2024)

12. Cohen, C.: Formalization of a sign determination algorithm in real algebraic geometry. <https://inria.hal.science/hal-03274013> (2021)
13. Cox, D., Little, J., O'Shea, D.: *Ideals, Varieties, and Algorithms*. Undergraduate Texts in Mathematics, 4th ed. Springer, Cham (2015)
14. Dolgov, S., Kalise, D., Saluzzi, L.: Data-driven tensor train gradient cross approximation for Hamilton-Jacobi-Bellman equations. *SIAM J. Sci. Comput.* **45**(5), A2153–A2184 (2023)
15. Dubé, T.: The structure of polynomial ideals and Gröbner bases. *SIAM J. Comput.* **19**(4), 750–773 (1990)
16. Farrell, P., Birkisson, A., Funke, S.: Deflation techniques for finding distinct solutions of nonlinear partial differential equations. *SIAM J. Sci. Comput.* **37**(4), A2026–A2045 (2015)
17. Gaillard, L., Safey El Din, M.: Solving parameter-dependent semi-algebraic systems. In: *International Symposium on Symbolic and Algebraic Computation*, pp. 447–456 (2024)
18. Grayson, D., Stillman, M.: *Macaulay2*, a software system for research in algebraic geometry. Available at <https://macaulay2.com> (2026)
19. Karg, B., Lucia, S.: Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Trans. Cybern.* **50**(9), 3866–3878 (2020)
20. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds) *International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015* (2015)
21. LaSalle, J.: Time optimal control systems. *Proc. Natl. Acad. Sci. USA* **45**, 573–577 (1959)
22. Li, L., Wang, L., Li, H.: An efficient spectral trust-region deflation method for multiple solutions. *J. Sci. Comput.* **95**(32) (2023)
23. Lopez, J., Maluccio, K., Sottile, F., Yahl, T.: Realroots, version 0.1. *Macaulay2* package. Available at <https://macaulay2.com/doc/Macaulay2/share/doc/Macaulay2/RealRoots/html/index.html> (2026)
24. Nakamura-Zimmerer, T., Gong, Q., Kang, W.: Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations. *SIAM J. Sci. Comput.* **43**(2), A1221–A1247 (2021)
25. Oster, M., Saluzzi, L., Wenzel, T.: A comparison study of supervised learning techniques for the approximation of high dimensional functions and feedback control. *Dyn. Games Appl.* **15**, 454–480 (2025)
26. Park, H., Regensburger, G.: *Gröbner Bases in Control Theory and Signal Processing*. De Gruyter, Berlin, Boston (2007)
27. Patil, D., Mulla, A., Chakraborty, D., Pillai, H.: Computation of feedback control for time optimal state transfer using Groebner basis. *Syst. Control Lett.* **79**, 1–7 (2015)
28. Patil, D.U., Chakraborty, D.: Computation of time optimal feedback control using Groebner basis. *IEEE Trans. Autom. Control* **59**(8), 2271–2276 (2014)
29. Phuoc Le, H., Safey El Din, M.: Solving parametric systems of polynomial equations over the reals through Hermite matrices. *J. Symb. Comput.* **112**, 25–61 (2022)
30. Pontryagin, L., Boltyanskii, V., Gamkrelidze, R., Mishchenko, E.: *The Mathematical Theory of Optimal Processes*. A Pergamon Press Book. The Macmillan Company, New York (1964). **(Translated by D. E. Brown)**
31. Rauscher, F., Sawodny, O.: Efficient online trajectory planning for integrator chain dynamics using polynomial elimination. *IEEE Robot. Autom. Lett.* **6**(3) (2021)
32. Sylvester, J.: On a remarkable modification of Sturm's theorem. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **5**, 446–456 (1853)
33. Vu, D.C., Pham, M.D., Nguyen, T.T.H., Nguyen, T.V.A., Nguyen, T.L.: Time-optimal trajectory generation and observer-based hierarchical sliding mode control for ballbots with system constraints. *Int. J. Robust Nonlinear Control* **34**(11), 7580–7610 (2024)
34. Walther, U., Georgiou, T., Tannenbaum, A.: On the computation of switching surfaces in optimal control: a Gröbner basis approach. *IEEE Trans. Autom. Control* **46**(4), 534–540 (2001)
35. Wang, Y., Hu, C., Li, Z., Lin, S., He, S., Zhu, Y.: Time-optimal control for high-order chain-of-integrators systems with full state constraints and arbitrary terminal states. *IEEE Trans. Autom. Control* **70**(3), 1499–1514 (2025)