




Nonlinear Control of Quadcopters via Approximate Dynamic Programming

Conference Paper**Author(s):**

Romero Aguilar, Ángel; [Beuchat, Paul N.](#) ; Stürz, Yvonne R.; [Smith, Roy](#) ; [Lygeros, John](#) 

Publication date:

2019

Permanent link:

<https://doi.org/10.3929/ethz-b-000363979>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

<https://doi.org/10.23919/ECC.2019.8796289>

Nonlinear Control of Quadcopters via Approximate Dynamic Programming

Angel Romero, Paul N. Beuchat, Yvonne R. Stürz, Roy S. Smith, and John Lygeros

Abstract—While Approximate Dynamic Programming has successfully been used in many applications involving discrete states and inputs such as playing the games of Tetris or chess, it has not been used in many continuous state and input space applications. In this paper, we combine Approximate Dynamic Programming techniques and apply them to the continuous, nonlinear and high dimensional dynamics of a quadcopter vehicle. We use a polynomial approximation of the dynamics and sum-of-squares programming techniques to compute a family of polynomial value function approximations for different tuning parameters. The resulting approximations to the optimal value function are combined in a point-wise maximum approach, which is used to compute the online policy. The success of the method is demonstrated in both simulations and experiments on a quadcopter. The control performance is compared to a linear time-varying Model Predictive Controller (MPC). The two methods are then combined to keep the computational benefits of a short horizon MPC and the long term performance benefits of the Approximate Dynamic Programming value function as the terminal cost.

I. INTRODUCTION

Dynamic Programming (DP), [1], is an important method for optimal decision making, relevant in a broad range of applications including finance, health sciences and engineering. While DP has been successfully applied to many specific problems [2], it suffers from severe limitations when dealing with high dimensional systems. Approximate Dynamic Programming (ADP) aims to alleviate these limitations by finding tractable approximations to the optimal policies that result from DP. ADP methods have proven to work well for certain practical applications such as the game of Tetris [3], or the game of chess [4]. However, despite considerable progress in the theory and algorithms of ADP, [5], [6], [7], there have not been many successful applications of ADP to systems with continuous state and input spaces.

In [8], the authors suggest the so-called *iterated Bellman inequality* as a novel way of obtaining approximations to the value function by solving convex optimization problems. The approach was validated by applying it to the linear model of a simple mechanical system. In [9] the iterated Bellman inequality method is used to approximate the tail cost of a Model Predictive Control (MPC) problem of a continuous state space and finite input space power inverter. This effectively allows for a shorter time horizon, and thus a reduced computational burden. In [10], the authors generalize the problem formulation of [8] by considering polynomials

This research was partially funded by the European Research Council under the project OCAL, grant number 787845. All authors are with the Automatic Control Laboratory at ETH Zürich, Switzerland, email: {beuchatp, stuerzy, rsmith, jlygeros}@ethz.ch

as the chosen function space, and then use sum-of-squares (SOS) techniques to relax the problem to a semi-definite program (SDP). This method is implemented and applied to control a miniature-helicopter using a simplified linear model around hover.

The combination of ADP methods we apply to quadcopters is most closely connected with the method proposed in [11]. There the authors introduce a novel way of computing successively tighter under-estimators to the optimal value function in an iterative fashion. This method has proven to be computationally tractable for the nonlinear quadcopter model considered in this paper, in contrast to [8]. In [12], a family of objective functions is chosen instead of only one, with a value function approximation computed for each and then put together in a point-wise maximum (PWM) approach. This reduces the dependency of the result on the choice of the individual objectives. Parts of these techniques are used in [13] and demonstrated in simulation on a nonlinear energy storage system with a low dimensional state-by-input space. In that application, the slow time scale allows for computationally demanding online policies.

Due to their inherent instability and ease of design, unmanned aerial vehicles (UAV) have widely been used as a test bench for control and robotics research, [14]. In [15], nonlinear MPC is used to control the fast attitude dynamics of a hexacopter vehicle, and a Linear Quadratic Regulator (LQR) is used for the slower position dynamics. A nonlinear constraint on the tilt angle is considered. The maneuverability and tracking performance of the system are shown in simulation and experiment while tracking setpoints on a square. Similarly, in [16], a real-time iteration linear MPC for the control of the position dynamics is run on-board a quadcopter.

In this paper, we combine multiple ADP methods to design a controller for the continuous, nonlinear, high dimensional model of a quadcopter and demonstrate the success of the methods in simulations and experiments. The key contributions of this work are:

- We combine the ADP techniques from [10], [11] and [12], leveraging the strengths of each.
- We develop the necessary approximation steps for synthesizing an ADP controller by this combined method for a high-dimensional quadcopter model.
- We demonstrate the benefits of this method compared to other linear and nonlinear control techniques in simulations and experiments (see video [17]).

The paper is organized as follows: In Section II, we present the deterministic control problem and introduce the ADP

formulation. In Section III, we derive the nonlinear model of the quadcopter, transform it to fit in our ADP formulation, and use it then to derive the approximation to the optimal value function. Section IV presents the results obtained from simulations and experiments.

II. APPROXIMATE DYNAMIC PROGRAMMING FORMULATION

In this section, we first state the optimal control problem to be solved, and then we present the techniques used for computing approximate solutions, which will be applied to a quadcopter in the next section.

A. Optimal Control Problem

We consider the discrete time dynamical system of a quadcopter with continuous state and input spaces and aim to minimize a discounted cost objective. The specific states, inputs and dynamics of the quadcopter will be given in Section III. For now, let us denote the state of the system at time k as $x_k \in \mathbb{R}^{n_x}$, and the control input as $u_k \in \mathbb{R}^{n_u}$. The evolution of the system is described by the dynamics function $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ such that

$$x_{k+1} = f(x_k, u_k). \quad (1)$$

The stage cost function is the non-negative cost of taking decision u_k when being in state x_k , denoted as $l : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}_+$. Given a *discount factor* $\xi \in [0, 1)$ and a stationary control policy of the form $\pi : \mathcal{X} \rightarrow \mathcal{U}$, the discounted infinite horizon cost is thus,

$$J_\pi(x) = \sum_{k=0}^{\infty} \xi^k l(x_k, \pi(x_k)), \quad \text{with } x_0 = x. \quad (2)$$

The aim is to find the policy that minimizes (2) for all $x \in \mathcal{X}$. The solution is characterized by the optimal value function $V^* : \mathcal{X} \rightarrow \mathbb{R}$ that satisfies the Bellman equation,

$$V^*(x) = \underbrace{\min_{u \in \mathcal{U}} l(x, u) + \xi V^*(f(x, u))}_{\mathcal{T}V^*}, \quad \forall x \in \mathcal{X}, \quad (3)$$

where \mathcal{T} is the Bellman operator. Given a solution of (3) the optimal policy is the so-called *greedy policy*, [5],

$$\pi^*(x) = \arg \min_{u \in \mathcal{U}} l(x, u) + \xi V^*(f(x, u)). \quad (4)$$

which minimizes the cost of taking the decision u now, plus the cost-to-go from the next time period onwards. We require the standard assumptions on the stage cost, dynamics, and existence of feasible policies to ensure that V^* and π^* exist, are time-invariant, and attain the minimum, for example [18, Assumption 4.2.1, 4.2.2] or [19, §1.2]. For a general problem instance, solving (3) and implementing (4) is intractable.

B. Linear Programming Approach to ADP with Polynomials

The monotone and contractive properties of the Bellman operator mean that any function $\hat{V} : \mathcal{X} \rightarrow \mathbb{R}$ satisfying the so-called *Bellman inequality* is a point-wise under-estimator of V^* , i.e.,

$$\hat{V}(x) \leq \mathcal{T}\hat{V}(x), \quad \forall x \in \mathcal{X} \Rightarrow \hat{V}(x) \leq V^*(x), \quad \forall x \in \mathcal{X}, \quad (5)$$

where \hat{V} is referred to as an *approximate value function*. This is the key property that motivates the various LP approaches to ADP proposed in [5], [8], [10], [11], [13]. Letting $\mathcal{F}(\mathcal{X})$ denote a space of functions mapping $\mathcal{X} \rightarrow \mathbb{R}$, and using the Bellman inequality, a point-wise under-estimator is found by solving the following optimization problem,

$$\underset{\hat{V}(x)}{\text{maximize}} \quad \int \hat{V}(x) c(x) dx \quad (6a)$$

$$\text{subject to} \quad \hat{V} \in \mathcal{F}(\mathcal{X}) \quad (6b)$$

$$\hat{V}(x) \leq \mathcal{T}\hat{V}(x), \quad \forall x \in \mathcal{X}, \quad (6c)$$

where the *state relevance weighting* function, $c(x)$, is a finite measure on the state space and it allows the practitioner to select the region where a better approximation is desired. When $\mathcal{F}(\mathcal{X})$ is the space of real-valued measurable functions on \mathcal{X} it is proven in [18] that the solution to (6) solves the Bellman equation for c -almost all $x \in \mathcal{X}$, however the infinite dimensional decision variable makes the problem intractable.

In this paper we use the space of polynomials for the function space $\mathcal{F}(\mathcal{X})$. This choice is motivated by [10] where the authors show that problem (6) can be reformulated as a finite SDP when the dynamics, stage costs, and spaces are described by polynomials. Letting $\mathcal{P}_d(\mathcal{X})$ denote the space of polynomials up to degree d , we replace (6b) by $\hat{V} \in \mathcal{P}_d(\mathcal{X})$ and hence the decision variables are the coefficients of the monomials up to degree d . Considering the minimization in the Bellman operator \mathcal{T} , the infinite constraints (6c) can be equivalently written as,

$$0 \leq \underbrace{-\hat{V}(x) + l(x, u) + \xi \hat{V}(f(x, u))}_{b(x, u)}, \quad \forall x \in \mathcal{X}, u \in \mathcal{U}, \quad (7)$$

where $b(x, u)$ is a polynomial when l and f are polynomials.

We now use the SOS S-Procedure [20] to reformulate (7) as a single Linear Matrix Inequality (LMI) constraint on the decision variable $\hat{V} \in \mathcal{P}_d(\mathcal{X})$. Letting *SOS* denote the set of polynomials that are representable as a sum of polynomials squared, then from [20, Theorem 3.3] we have the following equivalence for certifying that a polynomial $v \in \mathcal{P}_{2d}(\mathcal{X})$ is an SOS polynomial,

$$v(x) \in \text{SOS} \Leftrightarrow \exists M \succeq 0, \text{ s.t. } v(x) = z(x)^T M z(x), \quad (8)$$

where $z(x)$ is the vector of monomials of $x \in \mathbb{R}^{n_x}$ up to degree d , and M is a square matrix of size $\binom{n_x+d}{d}$. Letting $g_i(x, u) \geq 0$ denote the polynomials that describe the state and input spaces (\mathcal{X} and \mathcal{U}), then the following set of equations,

$$b(x, u) - \lambda(x, u) \sum_i g_i(x, u) \in \text{SOS}, \quad (9a)$$

$$\lambda(x, u) \in \text{SOS}, \quad (9b)$$

is a sufficient reformulation of (7) in the sense that (9) \Rightarrow (7). Replacing (6c) with (9), the multiplier $\lambda \in \mathcal{P}_d(\mathcal{X} \times \mathcal{U})$ is an additional decision variable with the polynomial degree of the multiplier as a choice. It is reasonable to choose

the largest multiplier degree for which the LMI constraints stemming from (9) are computationally tractable.

Finally, the objective (6a) is linear in the coefficients of $\hat{V} \in \mathcal{P}_d(\mathcal{X})$ and requires knowledge of the moments of the state relevance weighting function $c(x)$ up to order d . Thus, a point-wise under-estimator of V^* can be found using a commercial solver for the SDP relaxation of (6).

C. Point-wise Maximum Approach to ADP

To improve the quality of the approximate value function, we use the approach proposed in [11] that solves a sequence of optimization problems, each with constraints of the same size as (9).

First, we solve the SDP relaxation of (6) for a particular choice of the state relevance weighting, and denote the solution as \hat{V}_1^* . Then we solve the SDP relaxation of the following optimization problem with $j = 2$,

$$\text{maximize}_{\hat{V}_j} \int \hat{V}_j(x) c(x) dx \quad (10a)$$

$$\text{subject to } \hat{V}_j \in \mathcal{P}_d(\mathcal{X}), \quad (10b)$$

$$\hat{V}_j(x) \leq \left(\mathcal{T}_{k=1, \dots, j-1} \max \hat{V}_k^* \right) (x), \quad \forall x \in \mathcal{X} \quad (10c)$$

where \hat{V}_1^* is fixed problem data, and we denote the solution as \hat{V}_2^* . We then solve the SDP relaxation of (10) iteratively for $j \geq 3$, each time storing the solution \hat{V}_j^* as fixed data to be used in the next iteration. As the cost function (10a) is non-decreasing with the iterations of (10), we terminate when the improvement in the cost function becomes less than a pre-specified threshold. The approximate value function for a particular choice of $c(x)$ is set as the solution of the final iteration.

The steps given in [11] show how to reformulate (10c) as a polynomial inequality constraint similar to (7). The SOS S-Procedure is then applied and the resulting relaxation involves one LMI constraint with the same size as (9a), and $j - 1$ LMI constraints identical to (9b).

D. Online Policy

To construct an online policy, we first construct our best approximation to the value function, and we use this as a surrogate for V^* in the greedy policy (4). For different choices of the state relevance weighting, denoted $c_i(x)$ for $i = 1, \dots, N_c$, we perform the iteration described in Section II-C and denote $\hat{V}_{c_i}^*$ as the value function estimate returned by the final iteration. We take the PWM of these under-estimators as our best estimate for V^* , i.e.,

$$\max_{i=1, \dots, N_c} \hat{V}_{c_i}^*(x) \leq V^*(x), \quad \forall x \in \mathcal{X}. \quad (11)$$

The online policy, also called the *approximate greedy policy*, is thus,

$$\hat{\pi}(x) = \arg \min_{u \in \mathcal{U}} l(x, u) + \xi \max_{i=1, \dots, N_c} \hat{V}_{c_i}^*(f(x, u)), \quad (12)$$

where the maximum over the $\hat{V}_{c_i}^*$ is readily reformulated using an epigraph variable. Note that even though (10) is

a convex problem over the value function coefficients, the policy (12) might be a nonconvex problem over u .

To construct the surrogate for V^* , it remains to choose the c_i weightings and in Section IV we provide the weightings used for nonlinear control of a quadcopter. The surrogate for V^* can be considered as a sufficiently accurate estimate if the online performance of (12) achieves a low cost.

III. CONTROL OF A QUADCOPTER

In the following, the theory from the previous section will be used for the control of a quadcopter vehicle. The dynamics are brought into a suitable form by applying a reduction in the state space and a polynomial approximation.

A. Dynamical Model of the Quadcopter

To describe the dynamics of the quadcopter, two frames of reference are used, the *inertial (world) frame* and the *body frame* (attached to the quadcopter). The location of the body frame with respect to the inertial frame is denoted as $\vec{p} = [p_x, p_y, p_z]^\top$ and shown in Figure 1. The angular rates about the body frame axes $x^{(B)}$, $y^{(B)}$ and $z^{(B)}$ are denoted $\vec{\omega} = [w_x, w_y, w_z]^\top$. The orientation of the body frame relative to the inertial frame is given by $\vec{\psi} = [\gamma, \beta, \alpha]^\top$, the roll, pitch, and yaw intrinsic Euler angles, respectively. Using the ZYX convention for the Euler angles, and with $s_{(\cdot)} := \sin(\cdot)$, $c_{(\cdot)} := \cos(\cdot)$, the equations of motion for a quadcopter of mass m and with moment of inertia J , are readily derived as,

$$\ddot{\vec{p}}^{(I)} = \frac{1}{m} \sum_{i=1}^4 f_i \begin{bmatrix} c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma \\ s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ c_\beta c_\gamma \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \quad (13a)$$

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = J^{-1} \left(\begin{bmatrix} \sum_{i=1}^4 f_i d_{y_i} \\ \sum_{i=1}^4 -f_i d_{x_i} \\ \sum_{i=1}^4 f_i d_{\tau_i} \end{bmatrix} - \vec{\omega} \times J \vec{\omega} \right), \quad (13b)$$

where f_i is the thrust produced by propeller i , d_{y_i} and d_{x_i} are the distances from the axis of propeller i to the center of gravity of the quadcopter, d_{τ_i} is a constant of proportionality for how much torque is produced by thrust f_i , and g is the acceleration due to gravity. For more details on the derivation of the dynamics refer to [21]. The system thus has 12 states, given by $[\vec{p}^\top, \dot{\vec{p}}^\top, \vec{\psi}^\top, \vec{\omega}^\top]^\top$, and four inputs, given by the motor thrusts f_i , $i = 1, \dots, 4$.

B. System Model Approximation for ADP

In order to apply the ADP method from Section II, we approximate the full dynamics of (13) in three steps. First, we exploit time scale separation to reduce the state dimension of the dynamics by using the cascaded control structure shown in Figure 2. The inner controller considers the states $[\vec{\psi}^\top, \vec{\omega}^\top]^\top$ and uses the thrust inputs f_1, \dots, f_4 to track a reference attitude $\vec{\psi}_{\text{ref}}$ and total thrust $f_{T, \text{ref}}$. The outer controller considers the states $x = [\vec{p}^\top, \dot{\vec{p}}^\top]^\top$ and uses the inputs $u = [\gamma, \beta, \alpha, f_T]^\top$ to stabilize the quadcopter to a specified position and yaw setpoint. These inputs are given as references to the inner controller. This cascaded structure is common practice for quadcopter control [15], and the

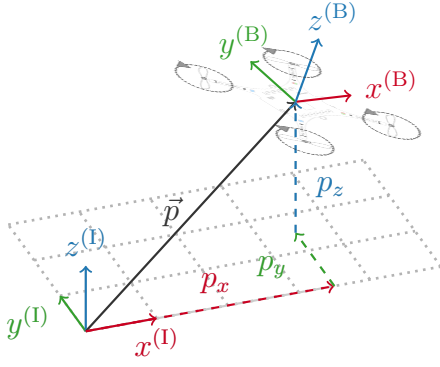


Fig. 1: Coordinate frames for the quadcopter: Inertial frame (I) and body frame (B).

approximation step assumes a sufficient time-scale separation so that the transients of the inner loop controller can be neglected when designing the outer loop controller. The goal is to design the outer controller using the combination of ADP methods described in Section II.

In the second approximation step, we use a Taylor expansion of (13a) to replace the trigonometric functions with a polynomial approximation that fits the SOS framework described in Section II-B. We use a third order Taylor expansion as a compromise between the accuracy of the approximated nonlinear dynamics and the size of the optimization problem (10) for fitting approximate value functions. The translational dynamics (13a) are thus approximated as

$$\begin{bmatrix} \dot{\vec{p}} \\ \frac{1}{m}(\beta f_T - \frac{1}{2}mg\alpha^2\beta + f_T\alpha\gamma - \frac{1}{6}mg\beta^3 - \frac{1}{2}mg\beta\gamma^2) \\ \frac{1}{m}(-\gamma f_T + f_T\alpha\beta + \frac{1}{2}mg\alpha^2\gamma + \frac{1}{6}mg\gamma^3) \\ \frac{1}{m}((f_T - mg) - \frac{1}{2}f_T\beta^2 - \frac{1}{2}f_T\gamma^2) \end{bmatrix}, \quad (14)$$

where we have introduced $f_T = \sum_i f_i$ to represent the total thrust.

Finally, to bring the dynamics of (14) to discrete time, we use a forward Euler scheme. The discrete time dynamics are required for applying the ADP methods to the design of the outer-loop controller. We choose a forward Euler approximation in order not to increase the polynomial order of the dynamics, and assume that the sampling time is fast enough to maintain stability properties of the plant. For the stage cost we chose the quadratic form, $l(x, u) = x^\top Qx + u^\top Ru$.

C. Input Constraints

We chose to constrain the tilt angle that the $z^{(I)}$ axis forms with the $z^{(B)}$ axis, denoted by θ and shown in Figure 3. The motivation behind this choice is linked to the nonlinearities in the dynamics. We choose θ large enough so that the nonlinearities are relevant for the solution of the optimal control problem, but small enough so that the third order Taylor expansion (14) is an adequate approximation of the true dynamics (13a). The angle constraint is given by

$$\cos(\theta) \leq \cos(\beta) \cos(\gamma), \quad (15)$$

which is nonlinear in the inputs γ and β , similar to the constraint introduced in [15].

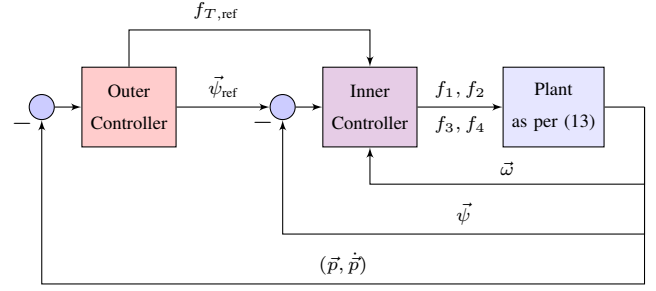


Fig. 2: Cascaded control structure. The references $f_{T,ref}$ and $\vec{\psi}_{ref}$ are the inputs u used in the model for synthesizing the outer controller. The inner loop consists of PID controllers that track these attitude and thrust references.

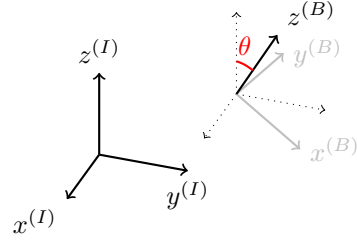


Fig. 3: Visualization of the constraint on the tilt angle θ .

To represent the physical actuator limits of the propellers, we impose a lower and upper bound on the total thrust, f_T . Denoting the lower and upper bound as $f_{T,lb}$ and $f_{T,ub}$, respectively, the constraint is thus,

$$f_{T,lb} \leq f_T \leq f_{T,ub}. \quad (16)$$

In addition to the dynamics, the constraints also need to be in polynomial form in order to apply the SOS techniques from Section II-B. We chose quadratic forms for the constraints in (15) and (16). The constraint on the angle in (15) is approximated via least-squares to the closest ellipse with radii a_1 and a_2 , and the constraint on the total thrust in (16) can be transformed into a single second order polynomial to reduce the number of SOS multipliers. The constraint functions are thus given by

$$1 - \frac{\gamma^2}{a_1^2} - \frac{\beta^2}{a_2^2} =: g_\theta(u) \geq 0, \quad (17a)$$

$$f_T(f_{T,lb} + f_{T,ub}) - f_{T,lb}f_{T,ub} - f_T^2 =: g_f(u) \geq 0. \quad (17b)$$

Note that $g_f(u)$ and $g_\theta(u)$ correspond to $g_i(x, u)$ in (9).

D. Value Function Fitting

Heuristics have been developed for how to choose the state relevance weighting, $c(x)$, which is the main tuning parameter. As presented in [12], to alleviate the sensitivity of this choice, we select a family of $c_i(x) \sim \mathcal{N}(0, \Sigma_i)$ as Gaussians with zero mean and different covariance matrices. Let $\text{diag}(v)$ define a diagonal matrix with the entries of vector v on its diagonal. Then, with $\Sigma_0 = \text{diag}([0.1, 0.1, 0.1, 1, 1, 1])$,

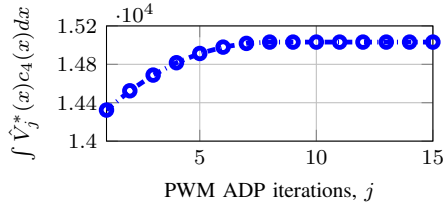


Fig. 4: Convergence of the objective function value of (10) over iterations j , for state relevance weighting c_4 in (18).

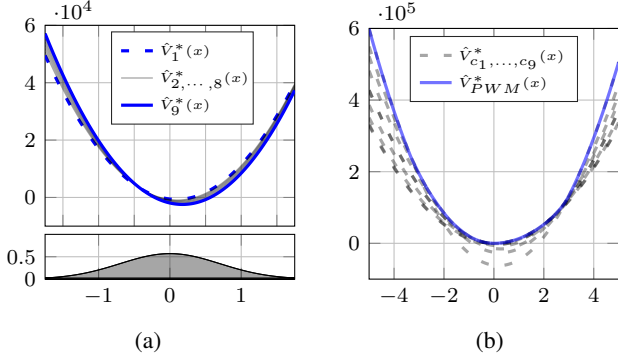


Fig. 5: Cuts around \hat{p}_z (a): of the approximate value functions $\hat{V}_j^*(x)$ for different iterations $j = 1, \dots, 9$ of (10) for $c_4(x)$ as shown in the lower plot, and (b): of the value function approximations $\hat{V}_{c_i}^*(x)$ for the $c_i(x)$ in (18) and their PWM $\hat{V}_{PWM}^*(x)$.

the set of covariance matrices Σ_i used is obtained as,

$$\begin{aligned} \Sigma_i &= K_i \Sigma_0, \quad i = \{1, \dots, 9\}, \quad \text{with} \\ K_i &\in \{0.01, 0.1, 0.3, 0.5, 0.7, 1, 1.3, 1.5, 2\}. \end{aligned} \quad (18)$$

For each different $c_i(x)$ in (18), we perform the iterative procedure described in Section II-C. As shown in Figure 4, the relative cost improvement converges to under a threshold of 10^{-5} within 9 iterations. In Figure 5(a), a cut of the value function approximation for different iterations for the same $c_i(x)$ is shown. The cut is obtained by setting all states to zero except \hat{p}_z . Solving this iterative problem for the different $c_i(x)$ results in a family of value function approximations, $\hat{V}_{c_i}^*(x)$, $i = 1, \dots, N_c$. As presented in Section II-D, the point-wise maximum (PWM) of this family, $\hat{V}_{PWM}^*(x) := \max_{i=1, \dots, N_c} \hat{V}_{c_i}^*(x)$, is the value function approximation that is used for the online greedy policy (12). Figure 5(b) shows a cut in the \hat{p}_z state of the value function approximations $\hat{V}_{c_i}^*(x)$ and of $\hat{V}_{PWM}^*(x)$.

IV. NUMERICAL RESULTS

In this section, in order to evaluate the control performance of the presented ADP approach, we compare different control schemes, which will be explained in detail in the following, with an overview given in Table I. The names are comprised of three parts, the first part encodes the approximation of the system dynamics used in the online policy, the middle part refers to the type of policy, and the last part indicates the terminal cost or value function used. The control task is a

TABLE I: Control policies compared in Section IV.

Name	dynamics approximation	policy	terminal cost
NL-GP-ADP	3rd order Taylor as in (14)	Greedy policy (12)	\hat{V}_{PWM}^* , §III-D
LTV-MPC-LQR	LTV as in (20)	MPC	P from Riccati
LTI-MPC-LQR	linearized around hover	MPC	P from Riccati
LTV-MPC-ADP	LTV as in (21)	MPC	\hat{V}_{PWM}^* , §III-D

setpoint change of 2 meters in p_x -direction, which shows the performance benefits of the control methods using the ADP value function compared to the other control schemes. First, results from simulations are given and then, experimental results on the quadcopter are presented.

A. Simulation Results

We consider the model of a small-sized quadcopter, the *Crazyflie 2.0*, [22], with a mass of 27 g. Details about the model parameters such as its inertia and dimensions can be found in [23], [24]. For the simulation the full state dynamics of the quadcopter as in (13) are used, together with the control structure shown in Figure 2. The inner loop tracks the commanded reference Euler angles ψ_{ref} with a PID controller running at 500 Hz, with the controller parameters and implementation details found in [24]. The outer control loop, and hence all the controllers listed in Table I, runs at 50 Hz.

In the ADP approach, the matrices for the stage cost, $l(x, u) = x^\top Qx + u^\top Ru$, have been chosen as,

$$\begin{aligned} Q &= \text{diag}([50 \quad 50 \quad 50 \quad 5 \quad 5 \quad 5]), \\ R &= \text{diag}([20 \quad 20 \quad 100 \quad 1500]), \end{aligned} \quad (19)$$

and the discount factor is set to $\xi = 0.98$. The total thrust is constrained to $f_T \in [0N, 0.56N]$. For the angle constraint in (15) we have chosen $\theta = 45^\circ$, which is large enough to demonstrate that the controller takes into account the nonlinearities of the system, but small enough for the constraint to become active.

At each time step, the online policy in (12) is solved and the optimal input is applied to the system. As the term $\hat{V}_{c_i}^*(f(x, u))$ is a composition of a quadratic and a third order polynomial, the resulting optimization problem is not convex. We refer to this as NL-GP-ADP, see Table I, and we solve it in simulation with an interior point method.

In the following, we compare NL-GP-ADP to a linear time-varying (LTV) MPC, [25], which also accounts for the nonlinear dynamics in its predictions. At each time step, the dynamics (13a) are linearized around the predicted trajectory, $x_{\text{traj},k}$, $u_{\text{traj},k}$, and denoted by A_k, B_k, g_k . Then,

the following problem is solved,

$$\begin{aligned} & \underset{\{u_k\}_{k=0}^{N-1}}{\text{minimize}} && \sum_{k=0}^{N-1} \|x_k - x_{\text{traj},k}\|_Q^2 + \|u_k - u_{\text{traj},k}\|_R^2 \\ & \text{subject to} && x_{k+1} = A_k x_k + B_k u_k + g_k, \\ & && u_k \in \mathcal{U}, \end{aligned} \quad (20)$$

where the set \mathcal{U} is defined as in (17), making this problem a convex Quadratically Constrained Quadratic Program (QCQP). Note that the x_k and u_k variables are deviations from the predicted trajectory. The cost matrices Q and R are the same as in (19), and P comes from the solution to the Riccati equation using the linearization of dynamics (13a) around the hover state $[\vec{p}, \dot{\vec{p}}]^\top = 0$. The first input, u_0^* , is applied, and the system evolves by one time step. The predicted trajectory for the next iteration, i.e., $x_{\text{traj},k}$, $u_{\text{traj},k}$, is based on the state measurement and the inputs u_1^*, \dots, u_{N-1}^* . We refer to this as LTV-MPC-LQR of horizon length N , see Table I.

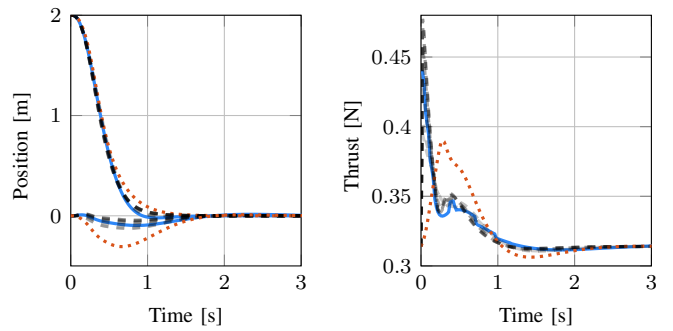
Furthermore, a comparison of the NL-GP-ADP policy to a controller that does not account for the nonlinear dynamics is given. For this we use a linear time-invariant (LTI) MPC, i.e., for the dynamics we use the linearized model around hover and the same Q , R , and P cost and Riccati solution matrices as in (20). Consistent with the naming convention, this policy is referred to as LTI-MPC-LQR of horizon $N = 1$, and the policy is also a convex QCQP. This policy is similar to the ADP approach demonstrated on miniature helicopters in [10], and for the control task we consider the performance was almost identical. Thus, for the sake of clarity, we do not include a policy based on [10] in the comparison here.

A comparison between the three policies is shown in Figure 6. The main difference between the approaches is the drop in p_z due to the fact that NL-GP-ADP and LTV-MPC-LQR have more information about the nonlinearities of the system than LTI-MPC-LQR which uses the linearized dynamics around hover. The nonlinearity of interest for this test is the change in vertical thrust when the quadcopter tilts, and as seen in Figure 6(b), LTI-MPC-LQR corrects for the drop in p_z after it happens. However, NL-GP-ADP and LTV-MPC-LQR pre-compensate for this drop by increasing the thrust in advance.

B. Experimental Results

For the experiments, the Crazyflie quadcopter is used together with a set of infrared cameras that provide the system with position and attitude measurements, [26]. Body frame angular rate measurements come from the on-board IMU. The control loops are the same as described in Section IV-A for the simulations.

Implementation of the NL-GP-ADP in real-time (i.e., at 50 Hz) was computationally intractable. To overcome this difficulty, the ADP approach is combined with the linear time-varying MPC scheme by replacing the terminal cost in (20) with the ADP value function approximation, leading to



(a) Evolution of p_x (starting from 2 meters) and p_z

(b) Evolution of f_T

Fig. 6: Comparison of the behavior of the system in simulation for: — NL-GP-ADP, ··· LTI-MPC-LQR $N=1$, and - - - LTV-MPC-LQR of different horizon lengths (from lighter to darker, $N = 1, 2, 5, 10, 15, 20$).

$$\begin{aligned} & \underset{\{u_k\}_{k=0}^{N-1}}{\text{minimize}} && \sum_{k=0}^{N-1} \|x_k - x_{\text{traj},k}\|_Q^2 + \|u_k - u_{\text{traj},k}\|_R^2 \\ & \text{subject to} && x_{k+1} = A_k x_k + B_k u_k + g_k, \\ & && u_k \in \mathcal{U}. \end{aligned} \quad (21)$$

We call this the LTV-MPC-ADP policy of horizon N . If we introduce the epigraph of all quadratics $\hat{V}_{c_i}^*(x)$ in \hat{V}_{PWM}^* as a new optimization variable, as in [12], the resulting optimization problem is a QCQP, making problem (21) implementable in real-time. Particularly, if in (21) we choose the horizon length to be 1, we have the *linearized greedy policy*, which differs from problem (12) in that the prediction of the next state is done using the linearized dynamics around the current state, instead of the nonlinear dynamics.

In Figure 7, we show a comparison of the experimental results when using LTV-MPC-ADP, LTV-MPC-LQR and LTI-MPC-LQR, all with a horizon length of $N = 1$. As can be seen in Figure 7, the initial thrust is higher for LTV-MPC-ADP, which suggests that it is able to predict the behavior of the system better than LTV-MPC-LQR. It shows that for the same horizon length, LTV-MPC-ADP reacts faster in the correction of the drop in p_z , since the initial thrust input is larger. This suggests that using the ADP value function approximation for the final cost yields better predictions and therefore, performance. The experimental comparison of LTV-MPC-ADP and LTI-MPC-LQR can be seen in the video at [17]. For details on the implementation, we refer to [24].

Figure 8 shows a series of simulations using the LTV-MPC-ADP, the LTV-MPC-LQR, and the LTI-MPC-LQR policies for horizon lengths from 1 up to 11. For every horizon length, the sum of stage costs, $\sum_{k=0}^N l(x_k, u_k)$ is computed, until all states and inputs have converged, and then plotted as a measurement of the performance of the policy. As the aim is to minimize (2), the smaller this cost is, the better the controller is performing. Figure 8 supports the results of the performance comparison given in Figure 6, as

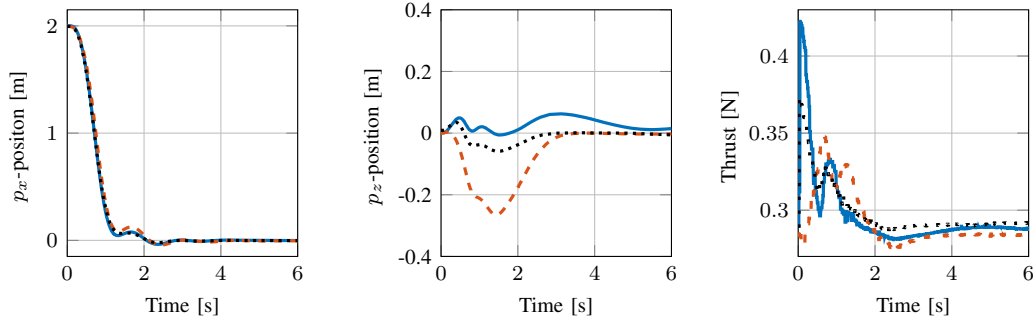


Fig. 7: Experimental comparison of different policies: — LTV-MPC-ADP, LTV-MPC-LQR, - - - LTI-MPC-LQR, all with horizon length $N = 1$.

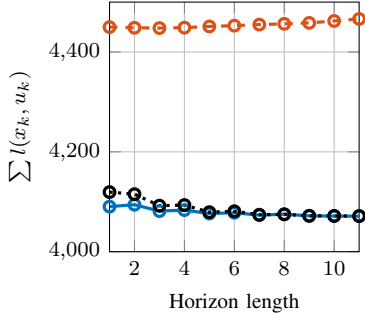


Fig. 8: Costs for different horizon lengths, for policies: — LTV-MPC-ADP, .. LTV-MPC-LQR and - - LTI-MPC-LQR.

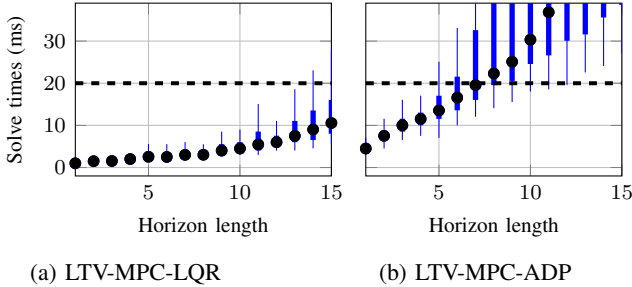


Fig. 9: Comparison of the solve times reported by the Gurobi solver [27] when playing the respective policy online. The black dotted line show the 50 Hz real-time limit.

for short horizon lengths the cost of LTV-MPC-ADP is lower than the one of LTV-MPC-LQR. It also shows that the quality of the approximation of the final cost loses importance when the horizon length increases.

Figure 9 shows that the maximum implementable horizon length for the LTV-MPC-ADP policy at 50 Hz is of 4-5 steps, while the LTV-MPC-LQR policy is tractable for up to 15 steps. The online computation time of the LTV-MPC-ADP policy can be reduced by using fewer approximate value functions in V_{PWM}^* , which introduces fewer quadratic constraints in (21). This generally introduces a trade-off with the online performance because the approximation quality of the cost-to-go may get reduced. Moreover, reducing

the computation time allows for a higher frequency which might improve the forward Euler approximation used for the discrete time dynamics.

The offline computation time required to compute the policies LTV-MPC-LQR and LTI-MPC-LQR is solving the Riccati equation, which takes less than one second for this model. The offline computation required to prepare \hat{V}_{PWM}^* , i.e., solving (6) and (10), for the LTV-MPC-ADP policy is less than 10 minutes on a standard desktop computer, using Yalmip [28] and Mosek [29] for building and solving the necessary SDPs. Note that the iterated Bellman inequality method from [30] would require solving a problem larger than (6) and (10), which is not tractable to solve in real-time.

V. CONCLUSION AND FUTURE WORK

We applied Approximate Dynamic Programming methods to a nonlinear, high dimensional quadcopter system with continuous state and input spaces. Using polynomial approximations of the dynamics, the constraints and the value function, leads to an online policy capturing the relevant nonlinearities of the system. For computational tractability, the ADP method was combined with an MPC scheme, which leverages the computational benefits of a short-term MPC scheme and the long term advantages of capturing the effects of nonlinearities by the ADP value function. The latter was used as the terminal cost in a linear time-varying MPC. For short time horizons this control method outperformed the same MPC with the terminal cost being the solution to a Riccati equation. The performance of the resulting controller was successfully demonstrated both in simulations and in experiments on a nano-quadcopter.

In future work, the control performance could further be improved by allowing for higher order polynomial approximations of the dynamics, or of relevant parts of the dynamics while keeping the complexity of other parts low, in order to ensure computational tractability. Furthermore, formulating the dynamics in terms of quaternions instead of Euler angles is promising as quaternion dynamics are naturally polynomial. However, more investigation is needed on how to effectively approximate the value function that only takes values on the quaternion manifold.

REFERENCES

- [1] R. E. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences*, vol. 38, no. 8, pp. 716–719, 1952.
- [2] S. Yakowitz, "Dynamic programming applications in water resources," *Water Resources Research*, vol. 18, no. 4, pp. 673–696, 1982.
- [3] V. Gabillon, M. Ghavamzadeh, and B. Scherrer, "Approximate dynamic programming finally performs well in the game of tetris," in *Neural Information Processing Systems (NIPS)*, pp. 1–9, 2013.
- [4] R. E. Bellman, "On the application of dynamic programming to the determination of optimal play in chess and checkers," *Proceedings of the National Academy of Sciences*, vol. 53, no. 2, pp. 244–247, 1965.
- [5] D. P. De Fariás and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. pages 850–865, 2003.
- [6] W. B. Powell and J. Ma, "A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multidimensional continuous applications," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 336–352, 2011.
- [7] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, *Handbook of learning and approximate dynamic programming*, vol. 2. John Wiley & Sons, 2004.
- [8] Y. Wang, B. O'Donoghue, and S. Boyd, "Approximate dynamic programming via iterated Bellman inequalities," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 10, pp. 1472–1496, 2015.
- [9] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 32, no. 5, pp. 4007 – 4020, 2017.
- [10] T. Summers, K. Kunz, N. Kariotoglou, M. Kamgarpour, S. Summers, and J. Lygeros, "Approximate dynamic programming via sum of squares programming," in *IEEE European Control Conference (ECC)*, pp. 191–197, 2013.
- [11] P. N. Beuchat, J. Warrington, and J. Lygeros, "Point-wise maximum approach to approximate dynamic programming," in *Conference on Decision and Control (CDC)*, pp. 3694–3701, 2017.
- [12] P. Beuchat, A. Georghiou, and J. Lygeros, "Alleviating tuning sensitivity in approximate dynamic programming," in *European Control Conference (ECC)*, pp. 1616–1622, 2016.
- [13] M. Hohmann, J. Warrington, and J. Lygeros, "A moment and sum-of-squares extension of dual dynamic programming with application to nonlinear energy storage problems," *preprint: arXiv:1807.05947*, 2018.
- [14] C. F. Liew, D. DeLatta, N. Takeishi, and T. Yairi, "Recent developments in aerial robotics: A survey and prototypes overview," *preprint arXiv:1711.10085*, 2017.
- [15] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on SOS(3)," in *Conference on Control Applications (CCA)*, pp. 1160–1166, 2015.
- [16] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *19th IFAC World Congress*, vol. 47, no. 3, pp. 11773 – 11780, 2014.
- [17] A. Romero and P. Beuchat, "Experimental results for ECC2019: Nonlinear control of quadcopters via ADP," 2019. <https://youtu.be/RBFuEdvPeDk>.
- [18] O. Hernández-Lerma and J. B. Lasserre, *Discrete-time Markov control processes: basic optimality criteria*, vol. 30. Springer Science and Business Media, 2012.
- [19] D. Bertsekas, *Dynamic programming and optimal control*, vol. 2. Athena Scientific, 4 ed., 2012.
- [20] P. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, no. 2, pp. pages 293–320, 2003.
- [21] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *Robotics Automation Magazine, IEEE*, vol. 19, pp. 20 –32, 09 2012.
- [22] "Crazyflie 2.0," 2018. <https://www.bitcraze.io/crazyflie-2/>.
- [23] A. Romero Aguilar, "Non-linear control of quad-copters via approximate dynamic programming," 2018. <https://doi.org/10.3929/ethz-b-000272622>.
- [24] Distributed Flying and Localization Lab, "Open-source code D-FaLL-System," 2018. <https://gitlab.ethz.ch/D-FaLL/PandS-System/D-FaLL-System>.
- [25] M. Diehl, H. Bock, and J. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.
- [26] "Motion capture system Vicon," 2018. <https://www.vicon.com>.
- [27] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2018. <http://www.gurobi.com>.
- [28] J. Löfberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *International Symposium on Computer Aided Control Systems Design*, pp. 284–289, IEEE, 2004.
- [29] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1*, 2017. <http://docs.mosek.com/8.1/toolbox/index.html>.
- [30] B. O'Donoghue, Y. Wang, and S. Boyd, "Iterated approximate value functions," in *IEEE European Control Conference (ECC)*, pp. 3882–3888, 2013.