

# A Machine Learning Enhanced Algorithm for the Optimal Landing Problem

**Yaohua Zang**

*Zhejiang University*

YHCHUANG@ZJU.EDU.CN

**Jihao Long**

*Princeton University*

JIHAOL@PRINCETON.EDU

**Xuanxi Zhang**

*Peking University*

ZHANGXUANXI@PKU.EDU.CN

**Wei Hu**

*Princeton University*

WH11@PRINCETON.EDU

**Weinan E**

*Peking University and Princeton University*

WEINAN@MATH.PRINCETON.EDU

**Jiequn Han**

*Flatiron Institute*

JIEQUNHAN@GMAIL.COM

**Editors:** Bin Dong, Qianxiao Li, Lei Wang, Zhi-Qin John Xu

## Abstract

We propose a machine learning enhanced algorithm for solving the optimal landing problem. Using Pontryagin's minimum principle, we derive a two-point boundary value problem for the landing problem. The proposed algorithm uses deep learning to predict the optimal landing time and a space-marching technique to provide good initial guesses for the boundary value problem solver. The performance of the proposed method is studied using the quadrotor example, a reasonably high dimensional and strongly nonlinear system. Drastic improvement in reliability and efficiency is observed.

**Keywords:** Optimal control, landing problem, deep neural networks, machine learning-based warm start

## 1. Introduction

The optimal landing problem is concerned with optimally controlling aerial vehicles to land on the target position. Developing effective numerical algorithms for the optimal landing problem has been a challenging task for some time, due to the high dimensionality of the state space and nonlinearity of the dynamics. Traditionally, there are two approaches for solving optimal control and landing problems: the direct method and the indirect method. The direct method (Vanderbei, 1999; Ross and Fahroo, 2002) first translates the optimal control problem to a nonlinear optimization problem by discretizing time and then solves the discretized problem using well-developed optimization solvers. This approach has been applied to lunar landing (Liu and Duan, 2006), rocket landing (Liu, 2019), quadrotor landing (Bouktir et al., 2008; Hu and Mishra, 2017), to name a few. The indirect method (Bock and Plitt, 1984; Bertolazzi et al., 2005; Wang, 2009) is based on the Pontryagin Minimum/Maximum Principle (PMP), which can be interpreted as the first-order optimality conditions of the optimal control problem. One then solves the two-point boundary value problem (TPBVP) derived from the PMP. The indirect method has also been applied to a variety of landing problems

with different terminal constraints (Guo et al., 2011; Assellaou et al., 2016; Hu et al., 2015). Despite the progress, these methods still have serious limitations such as the sub-optimality of the trajectory (Foehn et al., 2021; Romero et al., 2021), reliance on good initial guesses (Geisert and Mansard, 2016), and long computation time.

In recent years, starting from Han and E (2016), deep neural networks (DNN) have been widely used to solve high dimensional optimal control problem (see, e.g., Tang et al. (2018); Tang and Hauser (2019); Zhu et al. (2019); Beppu et al. (2021); Nakamura-Zimmerer et al. (2021a)). Sánchez-Sánchez and Izzo (2018) used the DNNs to approximate the mapping from states to optimal actions and presented applications on several landing problems. Tang et al. (2018) introduced a trajectory optimization approach that achieved real-time performance by combining machine learning to predict optimal trajectories with refinement by quadratic optimization. Zhu et al. (2019) used DNNs to learn the optimal action to improve the computational efficiency for the fuel-optimum lunar landing problem. Shi et al. (2019) presented a deep learning-based robust nonlinear controller to improve the control performance of a quadrotor during landing. You et al. (2020) developed a learning-based optimal control method for the Mars entry and power descent guidance to find optimal guidance laws in real-time. Instead of learning the feedback control directly, Nakamura-Zimmerer et al. (2021a) proposed to learn the value function via DNNs and predict the optimal feedback through the dynamic programming principle. Although they have shown great potential for the optimal control of high dimensional and strongly nonlinear systems, current DNN-based algorithms are still not robust enough (Chen et al., 2018; Nakamura-Zimmerer et al., 2021b). One major obstacle is the sensitive dependence on a good initialization at the various stages of the algorithms.

This paper proposes a new numerical method that combines the traditional indirect method and DNNs to solve the optimal landing problems with much improved reliability and efficiency. Overall we will use DNN-based prediction of the terminal time and space-marching technique to warm start the solving process and accelerate convergence. We will take the quadrotor unmanned aerial vehicles (UAVs) as an example to demonstrate the methodology. Rotary-wing UAVs have received widespread attention in recent years due to their wide range of application scenarios, including package delivery, film photography, agricultural inspections, and search and rescue missions. Among the many types of UAVs, the quadrotor is the most prominent aerial system. Due to its simplicity and versatility, the quadrotor has become the most flexible and maneuverable drone Ackermann (2020); Verbeke and Schutter (2018). We consider the full quadrotor dynamic model and aim to achieve an optimal landing path with minimum time and control effort. We start from the indirect method, utilizing the PMP to transform the original optimal landing problem into a TPBVP. One critical issue of the TPBVP solver is to find good initial guesses (Tsiotras et al., 2011; Nakamura-Zimmerer et al., 2021a). To overcome this difficulty, we mainly make two contributions to enhance the performance of TPBVP solver. First, we propose to use a solution to the fixed terminal time problem as an initial guess to the free time problem (Algorithm 2) and use neural networks to predict the optimal terminal time to obtain the best performance. Second, we introduce a space-marching scheme (Algorithm 3) to solve the fixed terminal time problem, which to the best of our knowledge, has not been applied in optimal control problems before. Compared to the baseline TPBVP solver, the proposed algorithm obtains the optimal landing trajectory with a much higher success rate and less computation time.

The paper is organized as follows. Section 2 presents the general form of control problems that our algorithm will be applied to and the specification of the optimal landing problem for the quadrotor. Our machine learning enhanced algorithm will be introduced in Section 3, with numerical results demonstrating its efficiency. Section 4 presents the difference between the obtained opti-

mal solution and the suboptimal solution provided by the model predictive control (MPC) method, another popular method for solving optimal control problems. Section 5 concludes the paper with some discussions on future work.

## 2. Formulation of the optimal control problem

We consider a deterministic system defined by the following ordinary differential equation (ODE)

$$\begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), & t \in [0, t_f] \\ \mathbf{x}(0) = \mathbf{x}_0, \mathbf{g}(\mathbf{x}(t_f)) = \mathbf{0}, \end{cases} \quad (1)$$

where  $\mathbf{x}(t) \in \mathbb{R}^n$  indicates the states,  $\mathbf{u}(t) \in \mathcal{U} \subset \mathbb{R}^m$  represents the controls with  $\mathcal{U}$  being the admissible set of the controls,  $f : \mathbb{R}^n \times \mathcal{U} \mapsto \mathbb{R}^n$  and  $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^k$  are smooth functions describing the dynamics and terminal condition. We call  $\{\mathbf{x}, \mathbf{u}, t_f\}$  a feasible path if (1) is satisfied and use  $\mathcal{P}$  to denote the set of all feasible paths. The total cost is defined as

$$\mathcal{C}[\mathbf{x}, \mathbf{u}, t_f] = \int_0^{t_f} L(\mathbf{x}(t), \mathbf{u}(t)) dt, \quad (2)$$

where  $L : \mathbb{R}^n \times \mathcal{U} \mapsto \mathbb{R}$  is the running cost, which is assumed to be smooth and non-negative.

We will consider two different but closely related problems. In the first problem,  $t_f$  is a given positive constant and we aim to minimize the performance function over all feasible paths with a fixed terminal time  $t_f$ :

$$\min_{(\mathbf{x}, \mathbf{u}) : \{\mathbf{x}, \mathbf{u}, t_f\} \in \mathcal{P}} \mathcal{C}[\mathbf{x}, \mathbf{u}, t_f]. \quad (3)$$

We call this problem a *fixed terminal time problem*. We are also interested in the *free terminal time problem*, where we aim to minimize the performance function over all feasible paths:

$$\min_{\{\mathbf{x}, \mathbf{u}, t_f\} \in \mathcal{P}} \mathcal{C}[\mathbf{x}, \mathbf{u}, t_f]. \quad (4)$$

Our ultimate goal is to solve this free terminal time problem to find the optimal landing trajectory and corresponding control with minimum time and control effort.

We remark that, in more complicated cases, there can be path constraints such as  $c(\mathbf{x}(t), \mathbf{u}(t)) \geq 0$  in the dynamical system. Here we only consider problems free of path constraints to highlight the main features of the techniques we introduce. These techniques can be naturally extended to problems with path constraints and, we intend to study this in future work.

### 2.1. Pontryagin's Minimum Principle

Pontryagin's Minimum Principle (PMP) establishes a set of necessary conditions for optimality, which converts the optimal control problem (3) or (4) to two-point boundary value problems (TPBVP). Algorithms based on the PMP are usually called indirect methods [Bock and Plitt \(1984\)](#); [Bertolazzi et al. \(2005\)](#); [Wang \(2009\)](#). To simplify the discussion, we assume that both the fixed terminal time problem (3) and the free terminal time problem (4) admit a unique minimizer and the solutions of corresponding TPBVPs are unique. This assumption ensures the optimality of the solutions of TPBVPs. In the examples considered here, we have not found multiple solutions to the TPBVPs.

To state the PMP, we introduce a costate variable  $\boldsymbol{\lambda} \in \mathbb{R}^n$  and define the Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) = L(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \quad (5)$$

The PMP reduces the fixed terminal time problem (3) to a system of equations in the form of

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \partial_{\boldsymbol{\lambda}}^T H(\boldsymbol{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{u}^*(t)), \\ \dot{\boldsymbol{\lambda}}(t) = -\partial_{\boldsymbol{x}}^T H(\boldsymbol{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{u}^*(t)), \end{cases} \quad (6)$$

together with the boundary conditions given by the original one augmented with the transversality conditions:

$$\begin{cases} \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{g}(\boldsymbol{x}(t_f)) = \mathbf{0}, \\ \nabla \boldsymbol{g}(\boldsymbol{x}(t_f)) \boldsymbol{\alpha} = \boldsymbol{\lambda}(t_f). \end{cases} \quad (7)$$

Here  $\nabla \boldsymbol{g}(\boldsymbol{x}(t_f)) \in \mathbb{R}^{n \times k}$ , and  $\boldsymbol{\alpha} \in \mathbb{R}^k$  is a multiplier. In addition, the optimal control  $\boldsymbol{u}^*(t)$  should satisfy the minimization of the Hamiltonian at each  $t$ :

$$\boldsymbol{u}^*(t) = \arg \min_{\boldsymbol{u} \in \mathcal{U}} H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}). \quad (8)$$

Equations (6), (7) and (8) together complete the PMP for the fixed terminal time problem (3). For the free terminal time problem (4), besides (6), (7) and (8), we need the extra condition for the optimal terminal time  $t_f$ :

$$H(\boldsymbol{x}(t_f), \boldsymbol{\lambda}(t_f), \boldsymbol{u}^*(t_f)) = 0. \quad (9)$$

See [Hartl et al. \(1995\)](#) for the proof of the above PMP and detailed discussions.

## 2.2. The optimal landing problem

### 2.2.1. THE FULL DYNAMIC MODEL OF QUADROTOR

To introduce the dynamics of the quadrotor, we let  $\{\mathcal{O}_E, \mathcal{X}_E, \mathcal{Y}_E, \mathcal{Z}_E\}$  denote the Earth-fixed coordinate system and  $\{\mathcal{O}_b, \mathcal{X}_b, \mathcal{Y}_b, \mathcal{Z}_b\}$  the body-fixed coordinate system, whose origin  $\mathcal{O}_b$  is at the center of mass (CoM) of the quadrotor. Then the dynamics of the quadrotor can be modeled as follows

$$\begin{cases} \dot{\boldsymbol{p}} = \boldsymbol{R}^T(\boldsymbol{\eta}) \boldsymbol{v}_b \\ \dot{\boldsymbol{v}}_b = -\boldsymbol{w}_b \times \boldsymbol{v}_b - \boldsymbol{R}(\boldsymbol{\eta}) \boldsymbol{g} + \frac{1}{m} \boldsymbol{f}_u \\ \dot{\boldsymbol{\eta}} = \boldsymbol{K}(\boldsymbol{\eta}) \boldsymbol{w}_b \\ \dot{\boldsymbol{w}}_b = -\boldsymbol{J}^{-1} \boldsymbol{w}_b \times \boldsymbol{J} \boldsymbol{w}_b + \boldsymbol{J}^{-1} \boldsymbol{\tau}_u, \end{cases} \quad (10)$$

where  $\boldsymbol{p} = (x, y, z)^T$  is the inertial position of the CoM in the Earth-fixed coordinates and  $\boldsymbol{v}_b = (v_x, v_y, v_z)^T$  is the linear velocity of the quadrotor expressed in the body-fixed coordinates.  $\boldsymbol{\eta} = (\phi, \theta, \psi)$  is the attitude of the quadrotor in the Earth-fixed coordinates defined by the Euler angles: roll( $\phi$ ), pitch( $\theta$ ) and yaw( $\psi$ ).  $\boldsymbol{w}_b = (p, q, r)^T$  denotes the angular velocity in the body-fixed coordinates. In total  $\boldsymbol{x} = (\boldsymbol{p}^T, \boldsymbol{v}_b^T, \boldsymbol{\eta}^T, \boldsymbol{w}_b^T)^T \in \mathbb{R}^{12}$  denotes the state variable.  $\boldsymbol{f}_u = (0, 0, T)^T$  and  $\boldsymbol{\tau}_u = (\tau_x, \tau_y, \tau_z)^T$  are the total thrust and body torques from the four rotors, which are forces applied by the control variables to adjust the quadrotor's dynamics. The constants  $m$  and  $\boldsymbol{g} = (0, 0, g)^T$

Table 1: List of notations in the quadrotor model.

Notation	Meaning
$\mathbf{p} = (x, y, z)^T$	the inertial position in the Earth-fixed coordinates
$\mathbf{v}_b = (v_x, v_y, v_z)^T$	the linear velocity in the body-fixed coordinates
$\boldsymbol{\eta} = (\phi, \theta, \psi)^T$	the Euler angles in the Earth-fixed coordinates
$\boldsymbol{\omega}_b = (p, q, r)^T$	the angular velocity in the body-fixed coordinates
$\mathbf{f}_u = (0, 0, T)^T$	the total thrust vector in the body-fixed coordinates
$\boldsymbol{\tau}_u = (\tau_x, \tau_y, \tau_z)^T$	the body torques from the four rotors in the body-fixed coordinates
$m$	the mass of quadrotor
$\mathbf{g} = (0, 0, g)^T$	the gravity vector in the Earth-fixed coordinates, where $g = 9.81m/s^2$
$\mathbf{R}^T(\boldsymbol{\eta}) \in \mathbb{R}^3$	the transformation from the Earth-fixed to the body-fixed coordinates
$\mathbf{K}(\boldsymbol{\eta}) \in \mathbb{R}^3$	the attitude kinematic matrix
$\mathbf{J} \in \mathbb{R}^3$	the constant inertia matrix
$\mathbf{S}(\boldsymbol{\omega}_b) \in \mathbb{R}^3$	the skew-symmetric matrix.
$\mathbf{F} = (F_1, F_2, F_3, F_4)$	the individual rotor thrusts vector

denote the mass and the gravity vector ( $g = 9.81m/s^2$  denotes the acceleration of gravity on Earth), respectively. The direction cosine matrix  $\mathbf{R}(\boldsymbol{\eta}) \in SO(3)$  (representing the transformation from the Earth-fixed coordinates to the body-fixed coordinates), attitude kinematic matrix  $\mathbf{K}(\boldsymbol{\eta})$  (relating the time derivative of the attitude representation with the associated angular rate) and constant inertia matrix  $\mathbf{J}$  are defined as follows

$$\mathbf{R}(\boldsymbol{\eta}) = \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \sin \theta \cos \psi \sin \phi - \sin \psi \cos \phi & \sin \theta \sin \psi \sin \phi + \cos \psi \cos \phi & \cos \theta \sin \phi \\ \sin \theta \cos \psi \cos \phi + \sin \psi \sin \phi & \sin \theta \sin \psi \cos \phi - \cos \psi \sin \phi & \cos \theta \cos \phi \end{bmatrix},$$

$$\mathbf{K}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{bmatrix}, \quad \mathbf{J} = \text{diag}(J_x, J_y, J_z),$$

where  $J_x, J_y,$  and  $J_z$  are the moments of inertia of the quadrotor in the  $x$ -axis,  $y$ -axis, and  $z$ -axis, respectively.

To ease the notation, we denote the controls as  $\mathbf{u} = (T, \tau_x, \tau_y, \tau_z)^T$ . Then we have  $\mathbf{f}_u = A\mathbf{u}$  and  $\boldsymbol{\tau}_u = B\mathbf{u}$  with  $A$  and  $B$  defined as

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that in practice the quadrotor is directly controlled by the the individual rotor thrusts  $\mathbf{F} = (F_1, F_2, F_3, F_4)^T$ , and we have the relation  $\mathbf{u} = E\mathbf{F}$  with

$$E = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ c & -c & c & -c \end{bmatrix},$$

where  $l$  is the distance from the rotor to the UAV's center of gravity and  $c$  is a constant that relates the rotor angular momentum to the rotor thrust (normal force). So once we obtain the optimal control  $\mathbf{u}^*$ , we are able to get the optimal  $\mathbf{F}^*$  immediately by the relation  $\mathbf{F}^* = E^{-1}\mathbf{u}^*$ . We also introduce the skew-symmetric matrix  $\mathbf{S}(\mathbf{w}_b)$

$$\mathbf{S}(\mathbf{w}_b) = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix},$$

which has the property

$$-\mathbf{w}_b \times \mathbf{v}_b = \mathbf{S}(\mathbf{w}_b)\mathbf{v}_b \quad \text{and} \quad -\mathbf{J}^{-1}\mathbf{w}_b \times \mathbf{J}\mathbf{w}_b = \mathbf{J}^{-1}\mathbf{S}(\mathbf{w}_b)\mathbf{J}\mathbf{w}_b.$$

Then equation (10) can be finally rewritten as

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{R}^T(\boldsymbol{\eta})\mathbf{v}_b \\ \dot{\mathbf{v}}_b = \mathbf{S}(\mathbf{w}_b)\mathbf{v}_b - \mathbf{R}(\boldsymbol{\eta})\mathbf{g} + \frac{1}{m}A\mathbf{u} \\ \dot{\boldsymbol{\eta}} = \mathbf{K}(\boldsymbol{\eta})\mathbf{w}_b \\ \dot{\mathbf{w}}_b = \mathbf{J}^{-1}\mathbf{S}(\mathbf{w}_b)\mathbf{J}\mathbf{w}_b + \mathbf{J}^{-1}B\mathbf{u}. \end{cases} \quad (11)$$

Table 1 summarizes all the notations used in full dynamic model of the quadrotor.

### 2.2.2. THE OPTIMAL CONTROL PROBLEM

We aim to solve the landing problem with the minimum control effort and shortest landing time under the dynamics described in (11). That is, to find the optimal controls to steer the quadrotor from some initial states  $\mathbf{x}_0 \in \mathcal{S}_0$  to a target state  $\mathbf{x}_{t_f} \in \mathcal{S}_T := \{\mathbf{x} : \mathbf{g}(\mathbf{x}) = \mathbf{0}\}$ .

For the landing problem, the terminal set is defined as

$$\mathcal{S}_T = \{\mathbf{x} = (\mathbf{p}^T, \mathbf{v}_b^T, \boldsymbol{\eta}^T, \mathbf{w}_b^T)^T \mid \mathbf{p} = \mathbf{v}_b = \mathbf{w}_b = \mathbf{0}, \phi = \theta = 0\}.$$

The running cost  $L$  in (2) is given by

$$L(\mathbf{x}, \mathbf{u}) = 1 + (\mathbf{u} - \mathbf{u}_d)^T Q_u (\mathbf{u} - \mathbf{u}_d),$$

where  $\mathbf{u}_d = (mg, 0, 0, 0)$  represents the reference control that balances with gravity and  $Q_u = \text{diag}(1, 1, 1, 1)$  represents the weight matrix characterizing the cost of deviating from the reference control. As the total cost  $\mathcal{C}[\mathbf{x}, \mathbf{u}, t_f]$  is the integration of the function  $L$  over  $[0, t_f]$ , the integration of the first term of  $L$  is exactly the landing time  $t_f$  that we hope to minimize and the integration of

the second term of  $L$  is the accumulation of deviations between the applied control and the reference control that enables the quadrotor to hover in the air. Therefore, the definition of function  $L$  is to minimize the landing time  $t_f$ , supplemented by a regularization term to make the control close to the reference one. Given the running cost, the optimal control problem with fixed terminal time can be written as

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{u}} \quad & \mathcal{C}[\mathbf{x}, \mathbf{u}, t_f] \\ \text{subject to} \quad & \begin{cases} \dot{\mathbf{p}} = \mathbf{R}^T(\boldsymbol{\eta})\mathbf{v}_b \\ \dot{\mathbf{v}}_b = \mathbf{S}(\mathbf{w}_b)\mathbf{v}_b - \mathbf{R}(\boldsymbol{\eta})\mathbf{g} + \frac{1}{m}\mathbf{A}\mathbf{u} \\ \dot{\boldsymbol{\eta}} = \mathbf{K}(\boldsymbol{\eta})\mathbf{w}_b \\ \dot{\mathbf{w}}_b = \mathbf{J}^{-1}\mathbf{S}(\mathbf{w}_b)\mathbf{J}\mathbf{w}_b + \mathbf{J}^{-1}\mathbf{B}\mathbf{u} \\ \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{p}(t_f) = \mathbf{v}(t_f) = \mathbf{w}(t_f) = \mathbf{0}, \phi(t_f) = \theta(t_f) = 0. \end{cases} \end{aligned} \quad (12)$$

The optimal control problem with free terminal time needs to further find the optimal terminal time  $t_f$  to minimize the total cost in (12).

To derive the TPBVP for the problem (12), we denote  $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_p^T, \boldsymbol{\lambda}_v^T, \boldsymbol{\lambda}_\eta^T, \boldsymbol{\lambda}_w^T)^T \in \mathbb{R}^{12}$  as the costate of  $\mathbf{x}$  and write down the Hamiltonian

$$\begin{aligned} H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}) = & L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}_p \cdot (\mathbf{R}^T(\boldsymbol{\eta})\mathbf{v}_b) + \boldsymbol{\lambda}_v \cdot (\mathbf{S}(\mathbf{w}_b)\mathbf{v}_b - \mathbf{R}(\boldsymbol{\eta})\mathbf{g} + \frac{1}{m}\mathbf{A}\mathbf{u}) \\ & + \boldsymbol{\lambda}_\eta \cdot (\mathbf{K}(\boldsymbol{\eta})\mathbf{w}_b) + \boldsymbol{\lambda}_w \cdot (\mathbf{J}^{-1}\mathbf{S}(\mathbf{w}_b)\mathbf{J}\mathbf{w}_b + \mathbf{J}^{-1}\mathbf{B}\mathbf{u}). \end{aligned}$$

Therefore, the TPBVP for the optimal control problem (12) is

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{R}^T(\boldsymbol{\eta})\mathbf{v}_b \\ \dot{\mathbf{v}}_b = \mathbf{S}(\mathbf{w}_b)\mathbf{v}_b - \mathbf{R}(\boldsymbol{\eta})\mathbf{g} + \frac{1}{m}\mathbf{A}\mathbf{u}^* \\ \dot{\boldsymbol{\eta}} = \mathbf{K}(\boldsymbol{\eta})\mathbf{w}_b \\ \dot{\mathbf{w}}_b = \mathbf{J}^{-1}\mathbf{S}(\mathbf{w}_b)\mathbf{J}\mathbf{w}_b + \mathbf{J}^{-1}\mathbf{B}\mathbf{u}^* \\ \dot{\boldsymbol{\lambda}}_p = \mathbf{0} \\ \dot{\boldsymbol{\lambda}}_v = -\mathbf{R}(\boldsymbol{\eta})\boldsymbol{\lambda}_p - \mathbf{S}^T(\mathbf{w}_b)\boldsymbol{\lambda}_v \\ \dot{\boldsymbol{\lambda}}_\eta = -\frac{\partial(\boldsymbol{\lambda}_p \cdot \mathbf{R}^T(\boldsymbol{\eta})\mathbf{v}_b)}{\partial \boldsymbol{\eta}} + \frac{\partial(\boldsymbol{\lambda}_v \cdot \mathbf{R}(\boldsymbol{\eta})\mathbf{g})}{\partial \boldsymbol{\eta}} - \frac{\partial(\boldsymbol{\lambda}_\eta \cdot \mathbf{K}(\boldsymbol{\eta})\mathbf{w}_b)}{\partial \boldsymbol{\eta}} \\ \dot{\boldsymbol{\lambda}}_w = -\mathbf{K}^T(\boldsymbol{\eta})\boldsymbol{\lambda}_\eta - \frac{\partial(\boldsymbol{\lambda}_v \cdot \mathbf{S}(\mathbf{w}_b)\mathbf{v}_b)}{\partial \mathbf{w}_b} - \frac{\partial(\boldsymbol{\lambda}_w \cdot \mathbf{J}^{-1}\mathbf{S}(\mathbf{w}_b)\mathbf{J}\mathbf{w}_b)}{\partial \mathbf{w}_b}, \end{cases} \quad (13)$$

with boundary conditions

$$\begin{cases} \mathbf{x}(0) = \mathbf{x}_0, \\ \mathbf{p}(t_f) = \mathbf{v}(t_f) = \mathbf{w}(t_f) = \mathbf{0}, \\ \phi(t_f) = \theta(t_f) = \lambda_\psi(t_f) = 0. \end{cases} \quad (14)$$

For the optimal control problem with free terminal time, the optimal final time  $t_f$  is further determined by

$$H(\mathbf{x}(t_f), \boldsymbol{\lambda}(t_f), \mathbf{u}^*(t_f)) = 0. \quad (15)$$

In both cases, the optimal feedback control  $\mathbf{u}^*$  at state  $\mathbf{x}$  and time  $t$  is

$$\mathbf{u}^* = \arg \min_{\mathbf{u}} H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}) = \mathbf{u}_d - (Q_u^T + Q_u)^{-1} \left( \frac{1}{m} A^T \boldsymbol{\lambda}_v + B^T (\mathbf{J}^T)^{-1} \boldsymbol{\lambda}_w \right).$$

In the free terminal time problem, the undetermined  $t_f$  brings difficulty to the time integrator in the TPBVP solver since the time interval keeps changing during the solving process. To circumvent this difficulty, we consider  $t_f$  as a new state  $x_{n+1} = t_f$  and introduce a new variable  $\tau = t/t_f = t/x_{n+1}$ . We then have

$$\dot{x}_{n+1} = 0 \quad (16)$$

and

$$\begin{cases} \dot{\mathbf{p}} = x_{n+1} (\mathbf{R}^T(\boldsymbol{\eta}) \mathbf{v}_b) \\ \dot{\mathbf{v}}_b = x_{n+1} (\mathbf{S}(\mathbf{w}_b) \mathbf{v}_b - \mathbf{R}(\boldsymbol{\eta}) \mathbf{g} + \frac{1}{m} A \mathbf{u}^*) \\ \dot{\boldsymbol{\eta}} = x_{n+1} (\mathbf{K}(\boldsymbol{\eta}) \mathbf{w}_b) \\ \dot{\mathbf{w}}_b = x_{n+1} (\mathbf{J}^{-1} \mathbf{S}(\mathbf{w}_b) \mathbf{J} \mathbf{w}_b + \mathbf{J}^{-1} B \mathbf{u}^*). \end{cases} \quad (17)$$

The cost function becomes

$$\mathcal{C} = x_{n+1} \int_0^1 L(\mathbf{x}(\tau), \mathbf{u}(\tau)) d\tau,$$

and the Hamiltonian becomes

$$\begin{aligned} H(\mathbf{x}, x_{n+1}, \boldsymbol{\lambda}, \mathbf{u}) = x_{n+1} & \left( L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}_p \cdot (\mathbf{R}^T(\boldsymbol{\eta}) \mathbf{v}_b) + \boldsymbol{\lambda}_v \cdot (\mathbf{S}(\mathbf{w}_b) \mathbf{v}_b - \mathbf{R}(\boldsymbol{\eta}) \mathbf{g} + \frac{1}{m} A \mathbf{u}) \right. \\ & \left. + \boldsymbol{\lambda}_\eta \cdot (\mathbf{K}(\boldsymbol{\eta}) \mathbf{w}_b) + \boldsymbol{\lambda}_w \cdot (\mathbf{J}^{-1} \mathbf{S}(\mathbf{w}_b) \mathbf{J} \mathbf{w}_b + \mathbf{J}^{-1} B \mathbf{u}) \right). \end{aligned}$$

Similarly, the costate  $\boldsymbol{\lambda}$  satisfies the following equation

$$\begin{cases} \dot{\boldsymbol{\lambda}}_p = \mathbf{0} \\ \dot{\boldsymbol{\lambda}}_v = -x_{n+1} (\mathbf{R}(\boldsymbol{\eta}) \boldsymbol{\lambda}_p + \mathbf{S}^T(\mathbf{w}_b) \boldsymbol{\lambda}_v) \\ \dot{\boldsymbol{\lambda}}_\eta = -x_{n+1} (Q_\eta^T + Q_\eta) (\boldsymbol{\eta} - \boldsymbol{\eta}_d) - x_{n+1} \left( \frac{\partial(\boldsymbol{\lambda}_p \cdot \mathbf{R}^T(\boldsymbol{\eta}) \mathbf{v}_b)}{\partial \boldsymbol{\eta}} - \frac{\partial(\boldsymbol{\lambda}_v \cdot \mathbf{R}(\boldsymbol{\eta}) \mathbf{g})}{\partial \boldsymbol{\eta}} + \frac{\partial(\boldsymbol{\lambda}_\eta \cdot \mathbf{K}(\boldsymbol{\eta}) \mathbf{w}_b)}{\partial \boldsymbol{\eta}} \right) \\ \dot{\boldsymbol{\lambda}}_w = -x_{n+1} \left( \frac{\partial(\boldsymbol{\lambda}_v \cdot \mathbf{S}(\mathbf{w}_b) \mathbf{v}_b)}{\partial \mathbf{w}_b} + \mathbf{K}^T(\boldsymbol{\eta}) \boldsymbol{\lambda}_\eta + \frac{\partial(\boldsymbol{\lambda}_w \cdot \mathbf{J}^{-1} \mathbf{S}(\mathbf{w}_b) \mathbf{J} \mathbf{w}_b)}{\partial \mathbf{w}_b} \right). \end{cases} \quad (18)$$

The transversality condition (15) then becomes

$$H(\mathbf{x}(1), \boldsymbol{\lambda}(1), \mathbf{u}^*(1)) = 0. \quad (19)$$

Therefore, we transform the TPBVP (13)-(15) with undetermined time-horizon to a TPBVP with a fixed time-horizon  $[0, 1]$ , which is easier to solve for a TPBVP solver.

### 3. Machine learning enhanced algorithm

In this section, we present our algorithm for solving the optimal landing problem and show numerical results for the quadrotor model. We use the same system parameters as in [Madani and Benallegue \(2006\)](#). We take the mass  $m = 2\text{kg}$ , the gravity  $g = 9.81\text{m/s}^2$ , the moment of inertia  $J_x = J_y = J_z/2 = 1.2416\text{kg} \cdot \text{m}^2$ . In the following experiments, we specify the domain of initial



state as  $\mathcal{S}_0 = \{x, y \in [-10, 10], z \in [5, 100], v_x, v_y, v_z \in [-0.5, 0.5], \theta, \phi \in [-\pi/4, \pi/4], \psi \in [-\pi, \pi]; \mathbf{w} = \mathbf{0}\}$ . We always uniformly sample 100 samples of  $\mathbf{x}_0$  from  $\mathcal{S}_0$  to estimate the success rate and average computation time of the algorithm. In Figure 1, we present an example solution of the optimal landing problem that manipulates the quadrotor from the starting position  $\mathbf{x}_0$  to the origin. Figure 1(a) shows the optimal trajectory in the Earth-fixed coordinates obtained by the proposed method. Figure 1(b) and Figure 1(c) show the optimal position  $\mathbf{p}$  and attitude  $\boldsymbol{\eta}$ , respectively.

In Section 3.1 below, we first demonstrate that the baseline TPBVP solver for the free terminal time TPBVP (17)-(19) hardly works. Then in Section 3.2, we show that the solution to the fixed terminal time TPBVP (13)-(14) can provide better initialization for the free terminal time problem. Finally, in Sections 3.3 and 3.4, we propose the space-marching technique and DNN-based prediction of the optimal terminal time to further improve the success rate and efficiency of the solver to the fixed time problem.

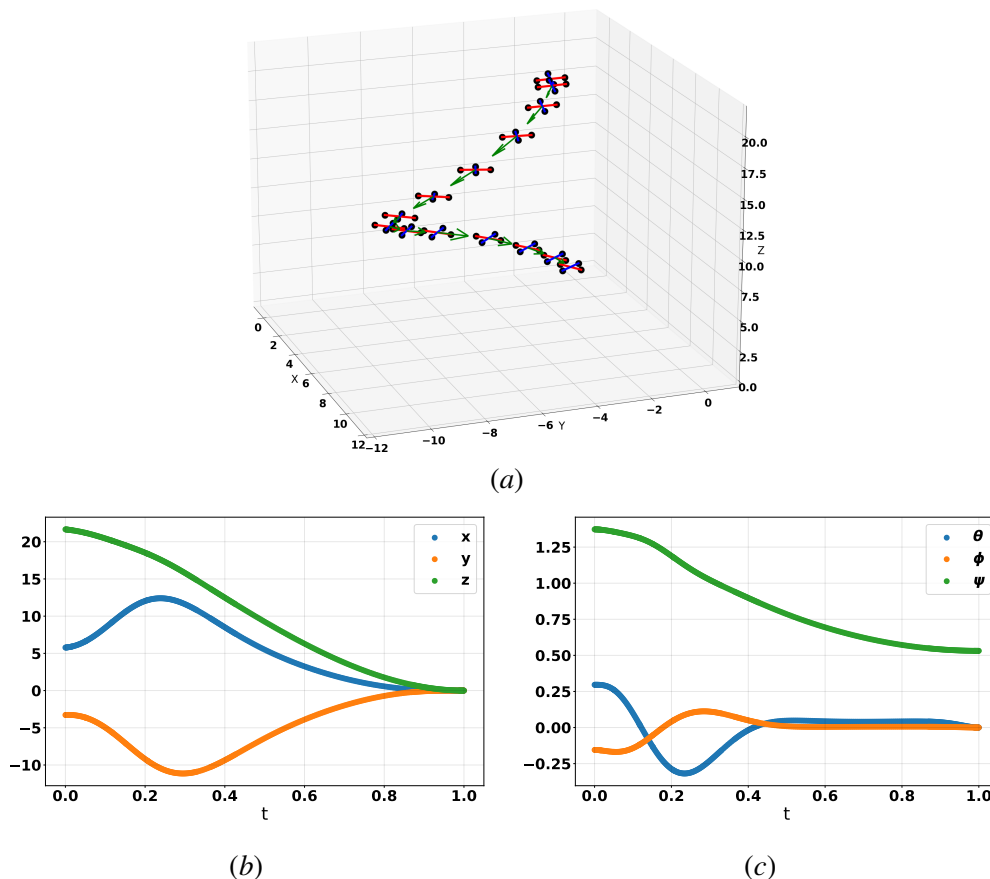


Figure 1: (a) An example of the optimal trajectory for the landing problem of quadrotor; (b) The optimal position  $\mathbf{p}$  vs. time; (c) The optimal attitude  $\boldsymbol{\eta}$  vs. time.

### 3.1. Baseline method for the TPBVP

Throughout the paper, we will use the classical *bvp4c* method (Kierzenka and Shampine, 2001) as the TPBVP solver. It is a collocation method based on piecewise cubic polynomials. The solution at all collocation points and unknown parameters are solved from a system of algebraic equations by the quasi-Newton method. Then the residual defined in terms of the interpolant is used to estimate the error and refine the mesh. By the nature of the quasi-Newton method, the quality of the initial guess of the solution is critical for the performance.

In our TPBVP, without any prior knowledge, the simplest choice of the initial guess is to set  $t_f$  to a reasonable scalar and  $\mathbf{x}(t), \boldsymbol{\lambda}(t)$  to the constant zero, as summarized in Algorithm 1. However, the TPBVP solver hardly converges with this choice. Table 2 reports the success rate of Algorithm 1 with a few different initial guesses of  $t_f$ . We can see that, with zero initialization of the path, the success rate of solving the free terminal time TPBVP is always very low, regardless of the initial guess of  $t_f$ .

---

**Algorithm 1** Free terminal time problem with zero initialization

---

- 1: **Input:** The initial state  $\mathbf{x}_0$ ; the guess value of the optimal terminal time  $\tilde{t}_f^*$ .
  - 2: Solve the TPBVP corresponding to the free time problem with the zero as the initial guess of the path and  $\tilde{t}_f^*$  as the initial guess of the terminal time.
  - 3: **Output:** The solution of the free time problem.
- 

Table 2: Solving TPBVP corresponding to the free terminal time problems with zero initialization

initial guess of $t_f$	4	8	12	16	20	24
success rate	3%	4%	0%	0%	1%	1%

### 3.2. Using solution to the fixed terminal time problem as initial guess

Table 2 suggests that such simple initial guess of  $\mathbf{x}(t)$  and  $\boldsymbol{\lambda}(t)$  leads to the TPBVP solver’s poor performance when solving the free terminal time problem. To address this issue, we notice that the solution of the free terminal time problem is also the solution of a corresponding fixed terminal time problem if the fixed terminal time  $t_f$  equals the optimal terminal time  $t_f^*$ . In other words, if we have a reasonable guess of  $t_f^*$ , the solution of the fixed terminal time problem can provide us a good initial guess to the free terminal time problem. Moreover, the fixed terminal time problem is easier to solve with many efficient techniques, such as the marching method introduced in the following subsection. Therefore, we can first guess a value of the optimal terminal time  $\tilde{t}_f^*$  and solve the fixed terminal time problem with  $t_f = \tilde{t}_f^*$ . Then we use its solution as the initial guess to solve the free terminal time problem. This approach can be viewed as a warm start method for solving the optimization problem. The corresponding algorithm is summarized in Algorithm 2 (*warm start with fixed terminal time solution*), in which the fixed time problem is solved using zero initialization.

The numerical results of Algorithm 2 with different choices of the initial guess  $t_f = \tilde{t}_f^*$  are presented in Figure 2. Comparing Figure 2 with Table 2, we can see that a warm start with the solution to the fixed terminal time problem significantly improves the success rate, although the rate is still not high enough for practical applications.

---

**Algorithm 2** Warm start with fixed terminal time solution

---

- 1: **Input:** The initial state  $\mathbf{x}_0$ ; the guess value of the optimal terminal time  $\tilde{t}_f^*$ .
  - 2: Solve the fixed time problem with  $t_f = \tilde{t}_f^*$  using zero initialization.
  - 3: Solve the free time problem by using the solution from step 2 as initial guess.
  - 4: **Output:** The solution of the free terminal time problem.
- 

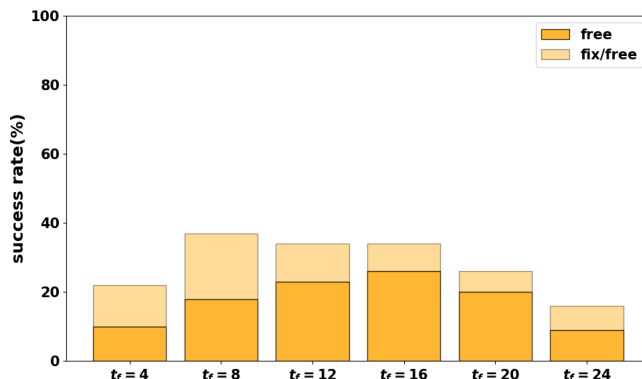


Figure 2: Warm start with fixed terminal time solution: the label “free” denotes the success rate for solving the free terminal time problems; the label “fix/free” (with a lighter color) denotes the rate that the fixed terminal time problems are solved successfully but the corresponding free terminal time problem is not solved successfully. The sum of the two parts is the success rate for solving the fixed terminal time problem.

### 3.3. Space-marching method for solving the fixed terminal time problem

To further increase the success rate of solving the fixed terminal time TPBVP, we propose a homotopy method in the spirit of space-marching (Ascher et al., 1995). Space-marching is primarily developed to solve differential equations. Here we tailor a similar idea to help solve our optimal landing problems. Its intuition is as follows. Solving the fixed time problem is still difficult since the initial state  $\mathbf{x}_0$  is far away from the terminal set  $\mathcal{S}_T$ . We can solve a simpler fixed time problem whose initial state is closer to the terminal state and the corresponding solution is not far from that of the original fixed time problem. After the simpler fixed time problem is solved, we can use its solution as the initial guess to solve the original harder one. In other words, we seek another level of a warm start to help solve the fixed terminal time problem, and this process can be performed repeatedly.

To present this method in a systematic way, we say  $\mathbf{x}_{end}$  is a terminal state if there exists  $\mathbf{u} \in \mathcal{U}$  such that for any  $t_f \geq 0$ , the path

$$\mathbf{x}(t) \equiv \mathbf{x}_{end}, \mathbf{u}(t) \equiv \mathbf{u}, 0 \leq t \leq t_f$$

is the optimal path for the fixed terminal time problem with  $t_f$  as terminal time and  $\mathbf{x}_0 = \mathbf{x}_{end}$ . We always assume such a terminal state exists for the optimal landing problem. In this paper, we choose the origin (of the 12-dimensional state space) as the terminal state. In order to solve the problem

with a given initial state  $\mathbf{x}_0$ , we evenly select  $K$  points in the line segment from  $\mathbf{x}_{end}$  to  $\mathbf{x}_0$ , and denote them as  $\{\mathbf{x}_0^1, \mathbf{x}_0^2, \dots, \mathbf{x}_0^K\}$  according to their increasing distances to  $\mathbf{x}_{end}$  ( $\mathbf{x}_0^K = \mathbf{x}_0$ ). We use  $\mathbf{x}_{aug}^k$  to denote the solution  $(\mathbf{x}(t), \boldsymbol{\lambda}(t))$ , ( $0 \leq t \leq \tilde{t}_f^*$ ) to the fixed time problem with the initial state  $\mathbf{x}_0^k$  and terminal time  $\tilde{t}_f^*$ ,  $k = 1, \dots, K$ . We assume  $\mathbf{x}_{aug}^0$  constant zero. The space-marching starts with  $k = 1$ . In the  $k$ -th step of marching, we solve the fixed time problem with the initial state  $\mathbf{x}_0^k$  by using the solution  $\mathbf{x}_{aug}^{k-1}$  obtained from the previous step as the initial guess. The process repeats until  $k = K$ . We call this algorithm *warm start with fixed terminal time solution through space-marching*, and it is summarized in Algorithm 3.

---

**Algorithm 3** Warm start with fixed terminal time solution through space-marching

---

- 1: **Input:** The initial state  $\mathbf{x}_0$ ; the guess value of the optimal terminal time  $\tilde{t}_f^*$ ; the the number of marching steps  $K$ .
  - 2: Evenly select  $K$  points in the line segment from  $\mathbf{x}_{end}$  to  $\mathbf{x}_0$ , and denotes them as  $\{\mathbf{x}_0^1, \mathbf{x}_0^2, \dots, \mathbf{x}_0^K\}$ .
  - 3: Initialize  $\mathbf{x}_{aug}^0$  with constant zero.
  - 4: **for**  $k = 1, 2, \dots, K$  **do**
  - 5:     Solve the fixed terminal time problem with initial state  $\mathbf{x}_0^k$  and terminal time  $\tilde{t}_f^*$  by using  $\mathbf{x}_{aug}^{k-1}$  as the initial guess. Denote the solution as  $\mathbf{x}_{aug}^k$ .
  - 6: **end for**  
       Solving the free time problem with  $\mathbf{x}_{aug}^K$  as the initial guess of the path and  $\tilde{t}_f^*$  as the initial guess of the terminal time.
  - 7: **Output:** The solution of the free terminal time problem with initial state  $\mathbf{x}_0$ .
- 

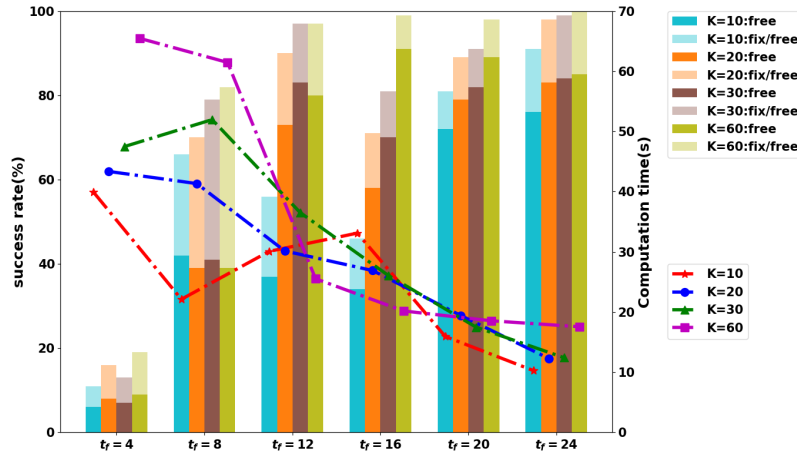


Figure 3: Warm start with fixed terminal time solution through space-marching: the histograms denote the success rates (left  $y$ -axis), whose labels share the same meaning as those in Figure 2; the dash-dotted lines denote the average computation time (in seconds) per path (right  $y$ -axis).

Figure 3 shows the success rate and computation time of Algorithm 3 with different choices of the initial guess  $t_f = \tilde{t}_f^*$  and marching steps  $K$ . We can see that most of the fixed terminal time problems can be solved with a high success rate if the guessed terminal time  $\tilde{t}_f^*$  is not too small and the number of marching steps  $K$  is large enough (greater than 20). However, the free terminal time problem might not be solved successfully if the guessed terminal time is not close enough to the optimal terminal time. To see this more clearly, we use Algorithm 3 with  $K = 60, \tilde{t}_f^* = 24$  to collect optimal time  $t_f^*$  associated with 300 randomly sampled initial states  $x_0$ . We plot the distribution of  $t_f^*$  in Figure 4, from which we can see that the values of  $t_f^*$  are distributed over a wide range. Hence, using a constant initial guess  $\tilde{t}_f^*$  cannot achieve the best performance. To improve the success rate of the free terminal time problem, we need a more accurate prediction of the optimal terminal time. We will investigate this issue in the next subsection.

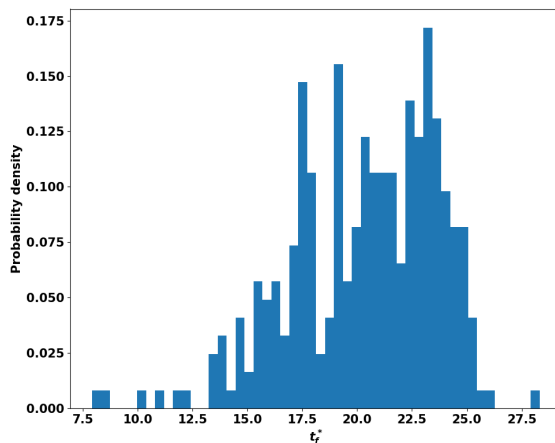


Figure 4: The distribution of optimal terminal time  $t_f^*$  obtained by solving 300 optimal landing problems with randomly selected initial states.

### 3.4. Predicting the optimal terminal time

In this section, we consider empowering Algorithm 3 by predicting the optimal terminal time as a function of the initial state  $x_0$  through a linear model or a neural network. To do so, we need to prepare a dataset for supervised learning. As described in the previous section, we first randomly select 300 initial states  $x_0$  and then use Algorithm 3 with  $K = 60, \tilde{t}_f^* = 24$  to collect 300 optimal landing paths (the specified terminal time 24 is not necessarily the optimal terminal time). We select 100 states (uniformly in time) on each optimal landing path and store the corresponding optimal landing time to obtain the training data. We have 30000 pairs of starting positions and optimal ending times for training in total. We then use this dataset to optimize a linear model and a neural network model (3 three hidden layers and 64 neurons in each layer) based on the objective being the squared difference between the predicted  $\tilde{t}_f^*$  and the truth optimal terminal time. The Adam optimizer (Kingma and Ba, 2015) is used to train the neural network model with 500 epochs, batch size 128, and learning rate 0.002. Afterwards, when we need to solve a free terminal time problem

with a new initial state  $x_0$ , we first use the linear model or the neural network to predict the optimal terminal time  $\tilde{t}_f^*$  associated with  $x_0$  and then use Algorithm 3 to solve the problem.

The success rates of using a constant ( $\tilde{t}_f^* = 24$ ), linear model, or a neural network to predict the optimal terminal time with the different space-marching steps  $K$  are presented in Figure 5. Comparing these results, together with those using other guessed constant optimal terminal time in Figure 3, we can see that both the linear model and neural network model achieve much higher success rates. Using a neural network attains higher success rates and takes less computation time because it can predict the optimal terminal time more accurately. With the help of neural networks and the space-marching with  $K = 60$ , we achieve a 99% successful rate, and the average computation time is about 17 seconds, which is the best performance among the methods considered in this paper.<sup>1</sup>

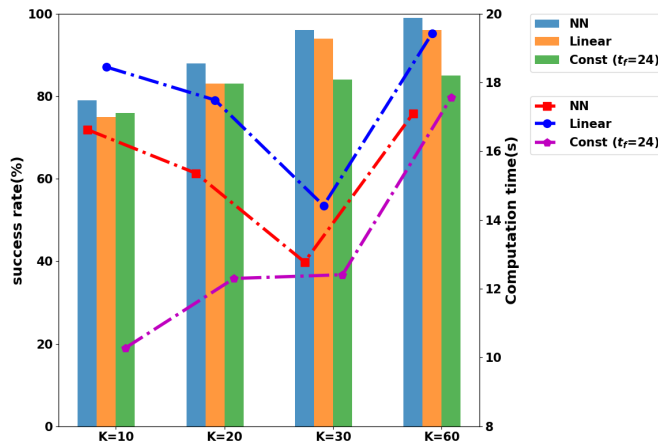


Figure 5: Comparison between the constant, linear model, and neural network model when they are used to predict  $t_f^*$  as the initial guess in the TPBVP solver. The histograms denote the success rates (left  $y$ -axis) of solving the free terminal time problem. Dash-dotted lines denote the average computation time (in seconds) per path (right  $y$ -axis).

The comparison among different methods proposed in this section with best hyper-parameters is listed in Table 3. Notice that Algorithm 1 refers to directly solving the free terminal problem with zero initialization, Algorithm 2 refers to a warm start with the fixed terminal time solution, and Algorithm 3 refers to a warm start with the fixed terminal time solution through space-marching.

#### 4. Suboptimality of Model Predictive Control

In this section, we study the performance of the model predictive control (MPC) method (Camacho and Alba, 2013), a widely used algorithm in optimal control, on the optimal landing problem. At each discretized time step, the algorithm computes a cost-minimizing control strategy starting from the current state for a relatively short time horizon (called the prediction horizon). The obtained control strategy is implemented for a very small time duration, then the states are updated and

1. We have also tested the performance of using 50 instead of 300 trajectories to optimize the neural networks. The success rate achieved with  $K = 60$  is 95%.

Table 3: Summary of success rate and computation time of different methods

	Algorithm 1 ( $t_f = 8$ )	Algorithm 2 ( $t_f = 16$ )	Algorithm 3 ( $t_f = 16,$ $K = 60$ )	Algorithm 3 ( $t_f$ : linear, $K = 60$ )	Algorithm 3 ( $t_f$ : NN, $K = 60$ )
success rate	4%	26%	89%	96%	99%
computation time (s)	8.24	9.78	19.46	19.43	17.10

the calculations are repeated. There have been various studies using the MPC method to control quadrotors (Bangura and Mahony, 2014; Ru and Subbarao, 2017; Eren et al., 2017; Romero et al., 2021). For application to the landing problem (Eren et al., 2017), a landing trajectory is required as the input. MPC can then use that to define the running cost in the prediction horizon in order to track that trajectory to land on the target position.

We pick an initial state and compute its optimal trajectory with our machine learning enhanced algorithm. The overall trajectory and trajectories of position and attitude are shown in Figure 6(a) and 6(b) with red color. The corresponding optimal terminal time  $t_f^*$  is 14.58(s), and the optimal cost obtained by the proposed method is 18.63. We then use the Matlab Model Predictive Control Toolbox (Bemporad et al., 2004) to track the optimal trajectory with the MPC method. We set the prediction horizon as 1.8(s) with a time difference  $\Delta t = 0.1$ (s) to strike a reasonable balance between computation accuracy and efficiency. We also modify the cost parameter in the prediction horizon to achieve the best tracking performance. The blue curves in Figure 6(b) show the trajectories of the position and attitude obtained by the MPC method. The corresponding total cost according to (2) is 32.31. From this result, we can clearly see that the MPC method can only find controls with suboptimality even provided the optimal trajectory. It is determined by the nature of the short horizon approximation scheme in the MPC method.

## 5. Conclusion and discussion

This paper proposes a machine learning enhanced algorithm for solving the optimal landing problem. The proposed algorithm is based on providing a good initialization for the corresponding TPBVP derived from PMP. It has two main components: predicting the terminal time using DNN and solving the resulting fixed time problem using a space-marching method. Through a series of experiments, the effectiveness of the proposed method has been verified. The techniques introduced in this paper are mainly designed for the indirect methods based on the TPBVP solver. It is of interest to study in the future work how machine learning techniques can enhance direct methods in some similar spirit. Another interesting direction for future work is to include the parameters in the dynamical systems as the input of the DNN such that the resulting DNN can deal with various dynamical systems.

So far we have mainly used neural networks to predict the optimal terminal time. An obvious alternative is to directly learn the mapping from the state to control or value function using DNN. Our effort along this line has not been successful so far. Here we present such examples. We consider the same problem in Section 3. Similarly as above, we randomly sample 1000, 1500, and 2000 samples of  $x_0$  from  $S_0$  and use them to generate 1000, 1500, and 2000 optimal trajectories, respectively. We then collect 20 (uniformly in time) states on each path and the corresponding value, resulting in

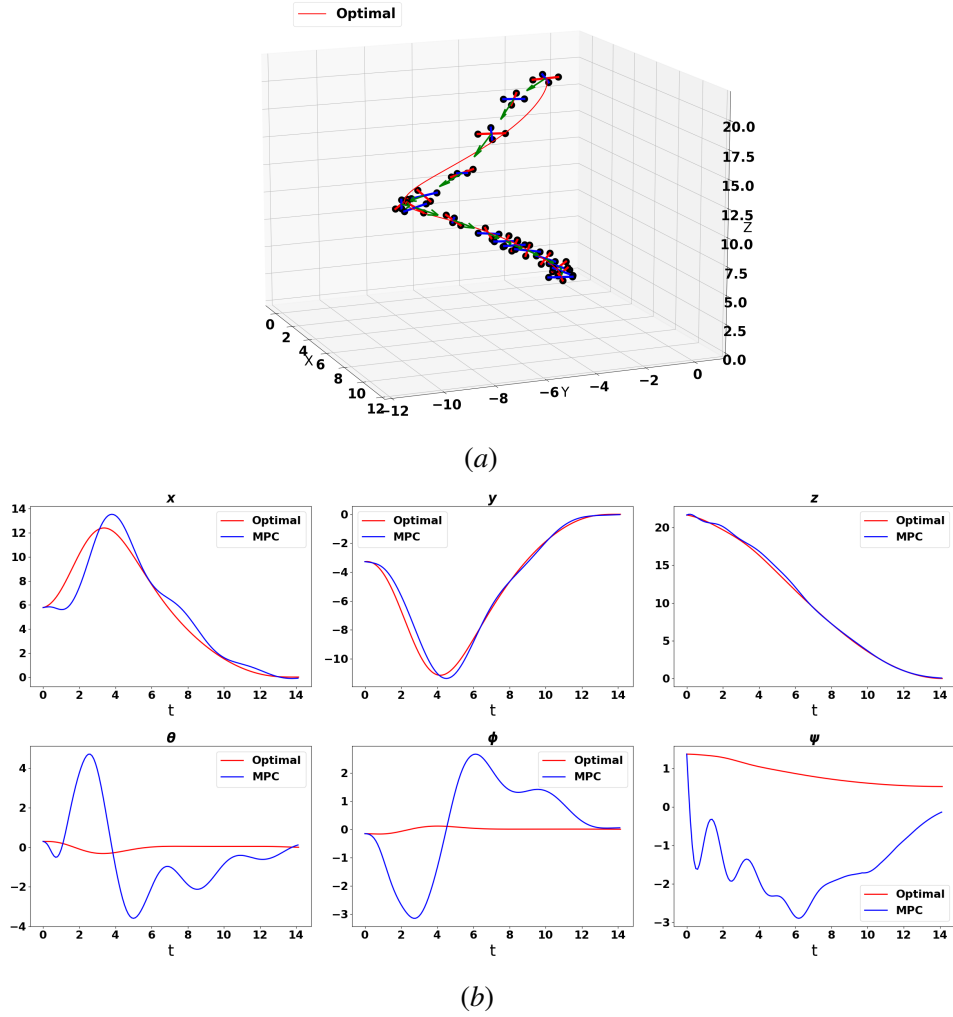


Figure 6: (a): The trajectory obtained by the MPC method (cost=32.31) following the optimal trajectory obtained by the proposed method (cost=18.63); (b): The optimal position  $(x, y, z)$  and attitude  $(\phi, \theta, \psi)$  (red) vs. those obtained by the MPC method (blue).

three training sets with 20000, 30000, and 40000 pairs of data, respectively. Finally, we use these training sets to train three different neural networks, denoted by  $NN_{1000}$ ,  $NN_{1500}$ , and  $NN_{2000}$ , respectively. According to the PMP, the gradients of the approximate value function should provide the optimal feedback control (Nakamura-Zimmerer et al., 2021a). Figures 7(a), 7(b) and 7(c) show trajectories on the time interval  $[0, t_f^*]$ , starting from the same initial state and following such control strategies provided by three neural networks. Compared to the optimal trajectory obtained by our algorithm, the neural network-based feedback control performs reasonably in the early stages of the landing but gradually deteriorates. Even when we use these trajectories as the initial guess for the



TPBVP solver, the solution still does not converge. This phenomenon is perhaps due to the high dimensionality and strong nonlinearity of the problem, as well as the wide range of scales spanned by the training data. It is an important future direction to explore techniques that can leverage neural networks to directly provide feedback control robustly for challenging problems like the optimal landing problem considered in this paper.

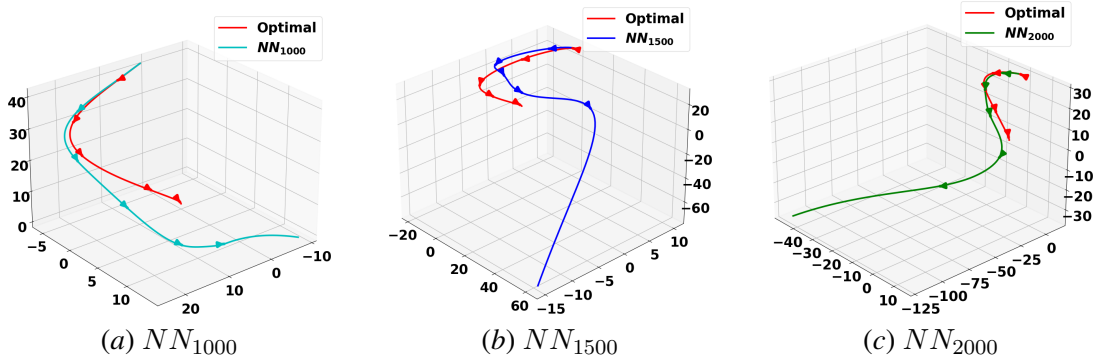


Figure 7: Trajectories on the time interval  $[0, t_f^*]$ , starting from the same initial state and following controls provided by neural networks  $NN_{1000}$ ,  $NN_{1500}$ , and  $NN_{2000}$ , respectively. The neural network-based feedback control performs reasonably well at the beginning and then deteriorates.

## References

- Evan Ackermann. AI-powered drone learns extreme acrobatics. *IEEE Spectrum*, 2020.
- Uri M Ascher, Robert MM Mattheij, and Robert D Russell. *Numerical solution of boundary value problems for ordinary differential equations*. SIAM, 1995.
- Mohamed Assellaou, Olivier Bokanowski, Anya Desilles, and Hasnaa Zidani. A Hamilton-Jacobi-Bellman approach for the optimal control of an abort landing problem. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3630–3635. IEEE, 2016.
- Moses Bangura and Robert Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.
- Alberto Bemporad, Manfred Morari, and N Lawrence Ricker. *Model predictive control toolbox. User’s Guide, Version, 2*, 2004.
- Hirofumi Beppu, Ichiro Maruta, and Kenji Fujimoto. Value iteration with deep neural networks for optimal control of input-affine nonlinear systems. *SICE Journal of Control, Measurement, and System Integration*, 14(1):140–149, 2021.
- Enrico Bertolazzi, Francesco Biral, and Mauro Da Lio. Symbolic–numeric indirect method for solving optimal control problems for large multibody systems. *Multibody System Dynamics*, 13(2):233–252, 2005.

- Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.
- Yasser Bouktir, Moussa Haddad, and Taha Chettibi. Trajectory planning for a quadrotor helicopter. In *2008 16th mediterranean conference on control and automation*, pages 1258–1263. Ieee, 2008.
- Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.
- Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835*, 2018.
- Utku Eren, Anna Prach, Başaran Bahadır Koçer, Saša V Raković, Erdal Kayacan, and Behçet Açıkmese. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017.
- Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor way-point flight. *Science Robotics*, 6(56):eabh1221, 2021.
- Mathieu Geisert and Nicolas Mansard. Trajectory generation for quadrotor based systems using numerical optimal control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 2958–2964. IEEE, 2016.
- Yanning Guo, Matt Hawkins, and Bong Wie. Optimal feedback guidance algorithms for planetary landing and asteroid intercept. In *AAS/AIAA astrodynamics specialist conference*, pages 2011–588. AAS, 2011.
- Jiequn Han and Weinan E. Deep learning approximation for stochastic control problems. *arXiv preprint arXiv:1611.07422*, 2016.
- Richard F Hartl, Suresh P Sethi, and Raymond G Vickson. A survey of the maximum principles for optimal control problems with state constraints. *SIAM review*, 37(2):181–218, 1995.
- Botao Hu and Sandipan Mishra. A time-optimal trajectory generation algorithm for quadrotor landing onto a moving platform. In *2017 American Control Conference (ACC)*, pages 4183–4188. IEEE, 2017.
- Botao Hu, Lu Lu, and Sandipan Mishra. Fast, safe and precise landing of a quadrotor on an oscillating platform. In *2015 American Control Conference (ACC)*, pages 3836–3841. IEEE, 2015.
- Jacek Kierzenka and Lawrence F Shampine. A BVP solver based on residual control and the Matlab PSE. *ACM Transactions on Mathematical Software (TOMS)*, 27(3):299–316, 2001.
- Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.
- Xinfu Liu. Fuel-optimal rocket landing with aerodynamic controls. *Journal of Guidance, Control, and Dynamics*, 42(1):65–77, 2019.

- Xinglong Liu and Guangren Duan. Nonlinear optimal control for the soft landing of lunar lander. In *2006 1st International Symposium on Systems and Control in Aerospace and Astronautics*, pages 6 pp.–1387. IEEE, 2006.
- Tarek Madani and Abdelaziz Benallegue. Backstepping control for a quadrotor helicopter. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260. IEEE, 2006.
- Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021a.
- Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Neural network optimal feedback control with enhanced closed loop stability. *arXiv preprint arXiv:2109.07466*, 2021b.
- Angel Romero, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. Model predictive contouring control for near-time-optimal quadrotor flight. *arXiv preprint arXiv:2108.13205*, 2021.
- I Michael Ross and Fariba Fahroo. A direct method for solving nonsmooth optimal control problems. *IFAC Proceedings Volumes*, 35(1):479–484, 2002.
- Pengkai Ru and Kamesh Subbarao. Nonlinear model predictive control for unmanned aerial vehicles. *Aerospace*, 4(2):31, 2017.
- Carlos Sánchez-Sánchez and Dario Izzo. Real-time optimal control via deep neural networks: study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018.
- Guanya Shi, Xichen Shi, Michael O’Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790. IEEE, 2019.
- Gao Tang and Kris Hauser. A data-driven indirect method for nonlinear optimal control. *Astrodynamics*, 3(4):345–359, 2019.
- Gao Tang, Weidong Sun, and Kris Hauser. Learning trajectories for real-time optimal control of quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3620–3625. IEEE, 2018.
- Panagiotis Tsiotras, Efsthathios Bakolas, and Yiming Zhao. Initial guess generation for aircraft landing trajectory optimization. In *AIAA Guidance, Navigation, and Control Conference*, page 6689, 2011.
- Robert J Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization methods and software*, 11(1-4):451–484, 1999.
- Jon Verbeke and Joris De Schutter. Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle. *International Journal of Micro Air Vehicles*, 10(1):3–11, 2018.
- Xuezhong Wang. Solving optimal control problems with MATLAB: Indirect methods. Technical report, 2009.

Sixiong You, Changuang Wan, Ran Dai, Ping Lu, and Jeremy R Rea. Learning-based optimal control for planetary entry, powered descent and landing guidance. In *AIAA Scitech 2020 Forum*, page 0849, 2020.

Lingchao Zhu, Jian Ma, and Shuquan Wang. Deep neural networks based real-time optimal control for lunar landing. In *IOP Conference Series: Materials Science and Engineering*, volume 608, page 012045. IOP Publishing, 2019.